Bruce Christianson   Bruno Crispo
James A. Malcolm   Michael Roe (Eds.)

Security Protocols

11th International Workshop
Cambridge, UK, April 2-4, 2003
Revised Selected Papers

Springer

Volume Editors

Bruce Christianson
James A. Malcolm
University of Hertfordshire
Computer Science Department
Hatfield AL10 9AB, UK
E-mail: {b.christianson,J.A.Malcolm}@herts.ac.uk

Bruno Crispo
Vrije Universiteit
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
E-mail: crispo@cs.vu.nl

Michael Roe
Microsoft Research Ltd
7 J.J. Thomson Avenue, Cambridge CB3 0FB, UK
E-mail: mroe@mircosoft.com

Library of Congress Control Number: 2005931713

CR Subject Classification (1998): E.3, F.2.1-2, C.2, K.6.5, J.1, K.4.1, D.4.6

ISSN 0302-9743
ISBN-10 3-540-28389-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-28389-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11542322 06/3142 5 4 3 2 1 0

Preface

Greetings. These are the proceedings of the 11th in our series of International Workshops on Security Protocols. Our theme this time was "Where have all the Protocols gone?" Once upon a time security protocols lived mainly in the network and transport layers. Now they increasingly hide in applications, or in specialised hardware. Does this trend lead to better security architectures, or is it an indication that we are addressing the wrong problems?

The intention of the workshops is to provide a forum where incompletely worked out ideas can stimulate discussion, open up new lines of investigation, and suggest more problems. The position papers published here have been revised by the authors in the light of their participation in the workshop. In addition, we publish edited transcripts of some of the discussions, to give our readers access to some of the roads ahead not (yet) taken. We hope that these revised position papers and edited transcripts will give you at least one interesting idea of your own to explore. Please do write and tell us what it was.

Our purpose in publishing these proceedings is to produce a conceptual map which will be of enduring interest, rather than to be merely topical. This is perhaps just as well, given the delay in production. This year we moved to new computer-based recording technology, and of course it failed completely. Fortunately various domestic recorders had been smuggled into the venue, but picking the signal bits out of the noise has taken a long time, and we have had to insert more than the usual number of epicycles to make the discussions come out right.

Our thanks to Sidney Sussex College Cambridge for the use of their facilities, to Lori Klimaszewska of the University of Cambridge Computing Service for the even worse than usual task of transcribing the audio tapes (in which the reverse use of "two rings" provided a test for creative intelligence) and to Johanna Hunt at the University of Hertfordshire for helping us with the Ptolemeic editing of the results.

Lent 2005

Bruce Christianson
Bruno Crispo
James Malcolm
Michael Roe

Previous Proceedings in This Series

The proceedings of previous International Workshops on Security Protocols have also been published by Springer as Lecture Notes in Computer Science, and are occasionally referred to in the text:

10th Workshop (2002), LNCS 2845, ISBN 3-540-20830-5
9th Workshop (2001), LNCS 2467, ISBN 3-540-44263-4
8th Workshop (2000), LNCS 2133, ISBN 3-540-42566-7
7th Workshop (1999), LNCS 1796, ISBN 3-540-67381-4
6th Workshop (1998), LNCS 1550, ISBN 3-540-65663-4
5th Workshop (1997), LNCS 1361, ISBN 3-540-64040-1
4th Workshop (1996), LNCS 1189, ISBN 3-540-63494-5

Table of Contents

[*] Speakers.

Where Have All the Protocols Gone?

Bruce Christianson

University of Hertfordshire, UK

There was a time when security protocols lived mainly in the network and transport layers. Where are they now?

Some have moved downstairs, towards the physical layer. What used to be a wide-area authentication or session establishment protocol is now a very local interaction with a trusted device, such as a tamper-evident smartcard, or a biometric token.

Indeed, in some cases a piece of mobile hardware has actually replaced altogether the security protocol that we used to find. Now in the strict sense, there is still a security protocol here: we use a set of rules to construct an artefact which will then be moved into a different context and interpreted in accordance with a shared set of conventions. But the individual protocol run no longer involves the same kind of electronic message-passing that we used to see or rather, as Marshall McLuhan would have said, the medium *is* now the message.

Other security protocols have moved upwards, to the application layer. The increasing deployment of Trusted Third Parties or TTPs is actually encouraging this trend. The part of the protocol that is visible from the point of view of the network, or of the system infrastructure, is now just the first, not so interesting, half of the protocol.

The quantum computing people regard entanglement as a resource. They have protocols for creating entanglement, but that's not what is interesting. The interesting question is what is done with the entanglement once it's created. How does it get used up?

The security protocol that we see in action across the network is often being used to create a cryptographic entanglement between a number of geographically separated entities. But again, the interesting thing is not how this entanglement got there, it is what happens afterwards[1]. What are these entities going to do with that entangled cryptographic resource? What happens next to the fresh session key after it gets handed off to the application? How does it get used up?

We don't see that backend to the protocol anymore. It's usually happening after the point where the analysis of the protocol has stopped, and it now tends to happen where we can't get at it to analyse it anyway. In many cases the really interesting part of the security protocol has moved all the way upstairs to the penthouse, above the API, where it is effectively hidden.

Still other security protocols have moved to a completely different level of abstraction. This is particularly the case for the protocols which used to be

[1] For more on this point, see Roger Needham's keynote address "Mobile Computing vs Immobile Security", LNCS 2467, 1–3.

used to enforce a security policy in a foreign domain. Many of these have now been replaced with service level agreeements, or SLAs[2]. Now an SLA is also a protocol, but at a much more abstract level. Just as with the biometric tokens, the security protocol itself now no longer involves the traditional type of explicit instance-based message passing. So nor do the threats.

Even the protocols that still seem to look much the way they always did are frequently not doing what they used to do. Partly this is because of the natural tendency to use security protocols for things other than those for which they were originally designed[3]. It's so hard to get a security protocol right, it always seems a shame not to reuse it. Although this isn't often a sensible thing to do.

But this is only part of the reason. The context in which security protocols are operating has also changed dramatically[4]. For example, twenty years ago it was an axiom that the primary purpose of security was to protect the system from the user. Most interesting failure stories now involve some kind of failure to protect the user from the system. I think this is actually the emerging security requirement which is least well met today.

To take another example, a lot of early security protocols were deliberately designed so as not to assume the existence of persistent shared state. Twenty years ago, constructing global state was a terribly difficult technical problem[5]. It was actually less bother to go to extreme lengths to construct protocols in such a way that they didn't rely upon any stable kind of global state. Nowadays the ease with which we can construct reliable, global state is very frightening. It makes various types of privacy invasion a real threat at the purely technical level, leaving aside any personal or political security dimensions[6].

And so finally, a lot of old security protocols have gone over to the dark side of the force, and are now effectively working for the bad guys. Many emerging security requirements are therefore, largely direct consequences of the success with which we've addressed the security problems of ten or twenty years ago.

Modern attacks tend to be built upon, and to exploit the security features of, the very infrastructure with which we ourselves have provided the attacker. Consequently we now need to turn a lot of our threat models inside out. Many countermeasures will become threats, and hopefully vice versa.

Is this trend for security protocols to migrate (and mutate) an opportunity which could lead to better security architectures, or is it an indication that we are addressing the wrong problems?

[2] Federated Identity Management is an example of this process, see B. Pfitzmann, these proceedings.

[3] See Roger's keynote address "Security Protocols and the Swiss Army Knife", LNCS 2133, 1–4.

[4] See Roger's keynote address "The Changing Environment for Security Protocols", LNCS 1796, 1–5.

[5] Conversely, constructing a global trust infrastructure was believed to be a problem which would soon be solved. Now we have learned to do without: see Roger's keynote address "Discerning the Protocol Participants", LNCS 2845, 1–3.

[6] For examples see the concluding discussion in LNCS 2467, 229–238.

A Protocol's Life After Attacks...

Giampaolo Bella[1,2], Stefano Bistarelli[3,4], and Fabio Massacci[5]

[1] Computer Laboratory, University of Cambridge, UK
[2] Dip. di Matematica e Informatica, Università di Catania, Italy
giamp@dmi.unict.it
[3] Dip. di Scienze, Università "G. D'annunzio" di Chieti-Pescara, Italy
bista@sci.unich.it
[4] Istituto di Informatica e Telematica, CNR, Pisa, Italy
stefano.bistarelli@iit.cnr.it
[5] Dip. di Informatica e TLC, Università di Trento, Italy
massacci@ing.unitn.it

Abstract. In the analysis of security protocols, it is customary to stop as soon as we find an attack. Tons of ink can be spilled on whether an "attack" is really an attack, but it goes without saying that there is no life after that, hence no interest in continuing the analysis. If the protocol is broken, then we ought to fix it.

Yet, fixing things is expensive and other measures may be more effective. In the physical world, most ATM safes would not resist heavy shelling with anti-tank bazookas, but banks don't worry about that. The attack will be noisy enough that cops will come within seconds from its start. To secure ourselves, we rely on a mixture of measures including the protection from attacks but also countermeasures after detection.

In the light of these considerations, the following question becomes of interest: *what can happen after an attack?* Does the villain leave enough traces that we can retaliate it on-the-fly? Or, if we can't or won't, does a subsequent forensic analysis allow us to discover who did it (and send the cops behind him)? If even this is impossible, can we discover that we have been hacked by looking at the logs?

To address these issues, we introduce the notions of *retaliation*, *detection*, and *suspicion*, which can be applied after an attack. These properties introduce more sophisticated formal relations between traces of actions, which go beyond the simple existentials that formal methods have made us used to.

These concepts should allow for a more comprehensive evaluation of security protocols. A protocol may well be vulnerable to an attack, but if we can retaliate afterwards, maybe fixing it isn't that necessary: the concrete possibilities of retaliation or detection may be enough to convince potential hackers to refrain from mounting the attack.

1 Introduction and Motivations

What is a security protocol, if we set technology aside? It is just a social behavior that principals of a distributed system must follow to obtain some important collective benefits. For the good guys, we just set up clear, understandable, and acceptable rules describing this behaviour: execute the security protocol correctly, namely by the book. Because they are good guys, they will conform to the rules, and behave as we wanted. The bad guys, by definition, will not conform to the rules and execute the protocol incorrectly, namely arbitrarily.

Classical research in distributed systems and security starts from the need to counter the disruptive behavior of the nasty ones. In classical distributed algorithms, the main focus has been to design the protocol so that if the good guys outnumber the bad ones, the collective benefits will be achieved, no matter what the bad guys do (alternatively, prove that the good guys are doomed, no matter how many they are and how smart they are [4]). The security standpoint is to find a design such that, no matter what the bad guys do and no matter how many they are, they can prevent the good guys from achieving the desired collective benefits.

Each time an attack is published, the implied corollary is that the security experts have failed: the protocol has flaws, the good guys cannot achieve their ultimate goal, and we should go back to the drawing board. Yet, before abandoning the protocol, it is worth looking at what is left after the attack. This can lead us to more comprehensive evaluation of security protocols even if it requires continuation of a protocol analysis after an attack is found.

1.1 Our Contribution

It has never been considered whether it is at all possible to threaten the bad guys in case they execute the protocol incorrectly. In the real world, we impose a virtuous behavior on people by not letting them sin first (the classical security approach), and by making them repent of their sins ever since (sending them to jail). They would therefore weigh up the benefits of an incorrect execution with the consequent threats, and might choose to execute the protocol correctly if the threats were heavier.

Let's consider Lowe's example in his paper on Needham-Schroeder [6]. After the end of the attack, the bad guy asks for a transfer of money. Would he steal 1.000 Euro if the threat that 2.000 Euro could be stolen to him on the next day were significant? Would he transfer the money if the chances of being caught were significant?

In this paper, we propose a number of notions for the analysis of protocols beyond attacks: we introduce the notions of *retaliation*, *detection*, and *suspicion.* For instance, in a peer-2-peer environment or in network games, *direct retaliation* can be an effective threat to force the bad guys to play by the rules.

In the next section we show how Lowe's attack on Needham-Schroeder can be retaliated [1]. Then, we introduce the notion of retaliation, followed by those of detection and suspicion.

2 Retaliation in Needham-Schroeder

Let's start from a classical case: the (in)famous Needham-Schroeder public key protocol represented in Figure 1. We use the classical notation for security protocols [2]:

- keys are denoted by K, possibly extended with subscripts expressing the principals knowing them; the $^{-1}$ superscript expresses the inverse of a key;
- nonces are denoted by N;
- concatenation is denoted by a comma;
- encryption is denoted by a pair of curly braces with the key as a subscript; the type of the key determines the type of encryption.

$$
\begin{array}{ll}
1. & A \rightarrow B : \{\!|Na, A|\!\}_{Kb} \\
2. & B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka} \\
3. & A \rightarrow B : \{\!|Nb|\!\}_{Kb}
\end{array}
$$

Fig. 1. The asymmetric Needham-Schroeder protocol

The goal of the protocol is *authentication*: at completion of a session initiated by A with B, A should get evidence to have communicated with B and, likewise, B should get evidence to have communicated with A. Assuming that encryption is perfect and that the nonces are truly random, authentication is achieved here by confidentiality of the nonces. Indeed, upon reception of Na inside message 2, A would conclude that she is interacting with B, the only principal who could retrieve Na from message 1. In the same fashion, when B receives Nb inside message 3, he would conclude that A was at the other end of the network because Nb must have been obtained from message 2, and no-one but A could perform this operation.

Lowe discovers [6] that the protocol suffers the "attack" described in Figure 2, whereby a malicious principal C masquerades as a principal A with a principal B, after A initiated a session with C. The attack, which sees C interleave two sessions, indicates failure of the authentication of A with B which follows from failure of the confidentiality of Nb.

$$
\begin{array}{ll}
1. & A \rightarrow C : \{\!|Na, A|\!\}_{Kc} \\
1'. & C \rightarrow B : \{\!|Na, A|\!\}_{Kb} \\
2'. & B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka} \\
2. & C \rightarrow A : \{\!|Na, Nb|\!\}_{Ka} \\
3. & A \rightarrow C : \{\!|Nb|\!\}_{Kc} \\
3'. & C \rightarrow B : \{\!|Nb|\!\}_{Kb}
\end{array}
$$

Fig. 2. Lowe's attack to the Needham-Schroeder Protocol

Let's examine this protocol after the attack took place:

- B is the subject of the attack — we call him the *good* agent;
- C is the *bad* guy;

- A is just playing by the rules — we call him the *ugly* participant;
- the trace T of the protocol as in Figure 2 describes an attack

The predicate $Attack(T, \mathcal{G} := \{B\}, \mathcal{B} := \{C\}, \mathcal{U} := \{A\})$ can represent the previous fact with the meaning:

- B executes a run of the protocol apparently with A (in fact he receives/sends the following messages:$[\{\!|Na, A|\!\}_{Kb}, \{\!|Na, Nb|\!\}_{Ka}, \{\!|Nb|\!\}_{Kb}] \in T$);
- C knows the nonce Nb, which was meant to be known to A and B only by receiving the message $\{\!|Nb|\!\}_{Kc} \in T$ in step 3 (so he can complete the run with B and perform the attack).
- A plays according to the rules and is just a spectator in the battle between B and C.

By using the authentication attack if B is a bank for example, C can steal money from A's account as reported by Lowe [6]:

$$4.\quad C \to B : \{\!|Na, Nb, \text{“Transfer £1000 from } A\text{'s account to } C\text{'s”}|\!\}_{Kb}$$

The bank B would honour the request believing it came from the account holder A.

Notice however, that the same predicate *Attack* also holds if instantiated in a different way [1]. In fact, we have also $Attack(T, \mathcal{G} := \{A\}, \mathcal{B} := \{B\}, \mathcal{U} := \{C\})$:

- A executes a run of the protocol apparently with C (in fact he receives/sends the following messages:$[\{\!|Na, A|\!\}_{Kc}, \{\!|Na, Nb|\!\}_{Ka}, \{\!|Nb|\!\}_{Kc}] \in T$);
- B knows the nonce Na, which was meant to be known to A and C only, by receiving the message $\{\!|Na, A|\!\}_{Kb} \in \mathcal{T}$ in step $1'$ (so he can send in step 2 the message $\{\!|Na, Nb|\!\}_{Ka}$ also without having C sending it to A).
- C is in this run just playing according to the rules.

In this case B is the bad guy. As C did previously, B can equally decide to illegally exploit his knowledge of Na. If also A is a bank, B can rob the robber as follows:

$$2''.\quad B \to A : \{\!|Na, Nb|\!\}_{Ka}$$
$$4''\quad B \to A : \{\!|Na, Nb, \text{“Transfer £2000 from } C\text{'s account to } B\text{'s”}|\!\}_{Ka}$$

The bank A would honour the request believing it came from the account holder C. After we found the attack $Attack(T, \mathcal{G} := \{B\}, \mathcal{B} := \{C\}, \mathcal{U} := \{A\})$ in the trace T, we could have continued the analysis and find out that the bad guy could be easily punished, as in the example given above.

3 Vendetta...

To introduce the notion of retaliation we must identify the role played by each principal in the protocol execution. We are not interested in notions such as responder or initiator; rather, we intend to provide a behavioural characterization.

Good principals are denoted by the set $\mathcal{G}$. They currently obey the protocol, but may change their mind in the future when attacked.

Bad principals are denoted by the set $\mathcal{B}$. They also take part in the protocol, possibly communicating with friendly and neutral agents but, crucially, they can as well collaborate with the intruder who runs the network.

Ugly principals are denoted by the set $\mathcal{U}$. Usually they mind their own business and conform to the protocol. However they can also both collaborate with corrupt agents or help in future the friendly agent to retaliate.

Informally, we assume that

- *a trace* $\mathcal{T}$ is a protocol run, namely a sequence of sending, receiving and memorizing actions,
- *a protocol* P is the set of all traces of the protocol[1] (all possible executions),
- *a projection* $\mathcal{T}/\mathcal{A}$ of a trace $\mathcal{T}$with respect to a set of agents $\mathcal{A}$ is the subsequence of the actions in $\mathcal{T}$ that are performed[2] by an agent in $\mathcal{A}$,
- *an attack* is a trace $\mathcal{T}$ that satisfies some property $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

We take the following definition of vulnerable protocol:

Definition 1. A protocol P is vulnerable to an attack A for the principals in $\mathcal{G}$ *if there exists a trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

The property as stated is suited for model checking, whereas people working with theorem provers will try to prove that there is no such trace.

Definition 2. A protocol P is immune to an attack A for the principals in $\mathcal{G}$ *if there exists no trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

What is the essence of retaliation? Each time you try to cheat, we can hit you back. So, do you still want to cheat?

This makes sense if hitting you back is a meaningful property in the context of the protocol. If the corrupted principals $\mathcal{B}$ are protocol participants, then we can assume that they want to reap the benefits of the protocol (such as authentication), plus any additional benefits they may obtain by misbehaving. These latter benefits should be balanced with the threat of being hit back. So, we simply need to pay an attacker with the same coin.

Definition 3. A protocol P allows the retaliation of an attack A onto the principals in $\mathcal{B}$ *iff for every trace* $\mathcal{T} \in P$ *such that* $A(T, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$*, there exists a trace* $T_r \in P$ *extending* $\mathcal{T}$ *such that* $A(\mathcal{T}_r, \mathcal{B}', \mathcal{G}', \mathcal{U}', Spy)$.

[1] This can be defined in the formal model of choice: CSP [10], inductive definitions [8], strand spaces [12].

[2] Note that the action in which A sends a message to B is included in the projection over the set $\{A\}$, but not in the projection over the set $\{B\}$, as B might never receive that message.

In particular,

- when we have $\mathcal{B}' = \mathcal{G}$ we speak of *direct retaliation*;
- when $\mathcal{B}' \cap \mathcal{G} \neq \emptyset$ we have a *combined retaliation*;
- lastly, when $\mathcal{B}' \subset \mathcal{G} \cup \mathcal{U}$ we have an *arbitrary retaliation.*

This definition follows the intuition that each time there is an attack (a trace $\mathcal{T}$ such that $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ some additional actions (hence $\mathcal{T}_r$ extending $\mathcal{T}$) can be done to retaliate. Who acted initially as friendly is now ready to act as corrupted (possibly against who was corrupted initially).

The property as stated is useful for theorem proving: this is a positive guarantee to make bad guys play by the rules. For model checking, and thus finding that not only are there attacks but you are even powerless against them, we need to turn the quantifiers.

Definition 4. A protocol P is vulnerable to an attack A without chance of retaliation onto $\mathcal{B}$ *if there exists a trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *and no trace* $\mathcal{T}_r \in P$ *extending* $\mathcal{T}$ *is such that* $A(\mathcal{T}_r, \mathcal{B}', \mathcal{G}', \mathcal{U}', Spy)$ *where* $\mathcal{B}' \in \mathcal{G} \cup \mathcal{U}, \mathcal{B} \in \mathcal{G}' \cup \mathcal{U}'$.

The important caveat here is that we assume that the formerly friendly principals will jump side and collaborate with the *Spy*, owner of the network, to frame the formerly corrupted principals. The spy will act as a devil helping the current corrupt agent (whoever he/she is). This view may be subject to debate but the key observation is that we need in any case some collaboration with the network provider, otherwise after the attack she could simply start killing every message that is sent on the network.

4 Detection

The notion of *detection* is more complicated and requires the comparison of alternative traces. Let's see the intuition: you look at your logs and spot some fishy actions (the projection of the full trace with respect to yourself). These actions are enough to bring the bad guys to trial if you can show that there is no other full trace which would have left the same marks on your logs and not included the attack.

In contrast, detection would fail if the bad guys could show at Court one trace, beside yours with the attack, with the same marks on your logs, but without the attack. Indeed, it would have been enough for them to show a trace with an attack from somebody else. This would have been sufficient to get acquittal.

Definition 5. A protocol P allows the detection of an attack A from $\mathcal{B}$ *if for every trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$, *for all traces* $\mathcal{T}_r \in P$ *such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$, *it is* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

The intuition is that each time there is an attack on a trace $\mathcal{T}$, the friendly agents can check their own logs $\mathcal{T}/\mathcal{G}$ and pinpoint who acted corruptly, namely the principals in $\mathcal{B}$. The same logs could not have been created by some other set $\mathcal{B}'$ of corrupted principals.

The model checking version of the same property requires two pieces of evidence: the logs of the good guys and the "plausible" no-attack alternative concocted by the bad guy's counsel for the defence.

Definition 6. A protocol P is vulnerable to an attack A without possibility of detection of $\mathcal{B}$ *if there exists some trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *and there exists a trace* $\mathcal{T}_r \in P$, $\mathcal{T}_r \neq \mathcal{T}$, *such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$ *and it is not* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

5 Suspicion

Suspicion is a weaker version of detection: each time there is an attack (a trace $\mathcal{T}$ such that $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$) we can look at the logs (the projection $\mathcal{T}/\mathcal{G}$) and conclude that something fishy happened but we cannot pin-point the villain.

Also in this case, the model checking version of the same property, requires two pieces of evidence: the logs of the good guys and the "plausible" no-attack alternative concocted by the bad guy's counsel for the defence.

Definition 7. A protocol P allows the suspicion of an attack A *if for every trace* $\mathcal{T} \in P$ *and set* $\mathcal{B}$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *then for all traces* $\mathcal{T}_r \in P$ *such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$ *there are sets* $\mathcal{U}'$ *and* $\mathcal{B}'$, *such that* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}', \mathcal{U}', Spy)$.

In a nutshell: each time the log matches with your logs, the traces include some attack by somebody.

The model checking version is by far more complicated:

Definition 8. A protocol P is vulnerable to an attack A without possibility of suspicion *if there exists some trace* $\mathcal{T} \in P$ *and set* $\mathcal{B}$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *and there exists a trace* $\mathcal{T}_r \in P$ *such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$ *and for all sets* $\mathcal{U}'$ *and* $\mathcal{B}'$ *it is not* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}', \mathcal{U}', Spy)$.

6 Implications for Formal Protocol Verification

We have been used to hair-splitting formal analysis of authentication. Can we formally analyze properties such as retaliation?

From a theoretical standpoint there is not a big difference. We have casted our properties as properties of traces because almost all research in tool-supported security verification is based on defining authentication or confidentiality as a property of traces [5,6,7,8,9,10] or fragments thereof [3,11].

The key observation is that the emphasis in the traditional work on security verification was on finding attacks or showing that no attack existed. This was reflected on formal models by the nature of the checked properties, which were essentially of existential nature: is there a trace $\mathcal{T}$ in the protocol P such that $A(\mathcal{T})$ holds? Here, $\mathcal{T}$, P, and A could be complicated at wish. Indeed, A, as a formally defined property, could be awfully complicated (for instance including arithmetical constraints on the number of actions and arbitrarily many quantifiers). However, the trace as such, was only existentially qualified. Theorem proving fellows wished to prove that no such a trace existed, model checking guys longed for a witness of its existence.

Here, the properties are more complex: we run at least two quantifiers over traces. In some cases we have alternation of quantifiers (such as retaliation). Lifting the theory is simple. Lifting the automatic tool support will be the real challenge.

References

1. G. Bella and S. Bistarelli. Confidentiality levels and deliberate/indeliberate protocol attacks. *Springer, LNCS, Proc. Security Protocols 10th International Workshop, Cambridge, UK, April, 2002, Revised Papers*, to appear.
2. M. Burrows, M. Abadi, and R.M. Needham. A logic for authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.
3. Luigia Carlucci Aiello and Fabio Massacci. Verifying security protocols as planning in logic programming. *ACM Transactions on Computational Logic*, 2(4):542–580, 2001.
4. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge.* The MIT Press, 1995.
5. R. Kemmerer, C. Meadows, and J. Millen. Three system for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.
6. G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.
7. J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murphi. In *Proceedings of the 16th IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society Press, 1997.
8. Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
9. Focardi R. and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.
10. Steve Schneider. Security properties and CSP. In *Proceedings of the 15th IEEE Symposium on Security and Privacy*, pages 174–187. IEEE Computer Society Press, 1996.
11. Dawn Song. Athena: An automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop.* IEEE Computer Society Press, 1999.
12. F. Thayer Fabrega, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop.* IEEE Computer Society Press, 1998.

A Protocol's Life After Attacks...
(Transcript of Discussion)

Giampaolo Bella

Computer Laboratory, University of Cambridge, UK

I am going to be speaking about protocol verification again; I'm going to take a rather different perspective from the one we normally take, and I'll be talking about what happens after an attack takes place. Is there a life for a protocol beyond the attacks?

We all know about verification. On the one hand we have the model checking community trying to find a witness of an attack, trying to find if something went wrong and why the specific property of interest failed. On the other hand we have the opposite approach, assuring that there's no such witness therefore the specific property holds. But the question here is, is this the whole story? It appears that everything is about finding the attack: is there an attack, is there no attack against confidentiality or authentication? It appears kind of weird. Is it only the attack we are really interested in? Is this really all we should look at? I'll try and convince you that there's something more.

So, let's suppose for a minute we own a jewellers, and one day we find that the main window has been completely smashed by someone. In the worst case there is no-one around and basically all we can do is suspect anyone, any passer-by, because there's really no evidence against anyone. If we're luckier, we could find the people there while they're still at work carrying away the stuff. Basically we detect who actually mounted the attack and we're kind of happy with that, as we're sure who the attacker is because we saw them. But we can even do more than that, maybe we have time to call the police, and the attackers will be caught, punished, and sent to jail, so we basically retaliate against them. If you move this to a different context perhaps retaliating means that I go up to the attacker's window and smash the window. Anyway, this is just a general notion of punishment and retaliation.

This is certainly about the best we can do. This line attempts to convince us that there are some measures we normally take after an attack takes place in the real world. So the idea here is to apply these very same concepts to the world of security protocols and see what we can get out of it.

So how should we do this? Well in the first place we shouldn't just stop in the face of an attack as we find it, as we are used to doing up until now, we should continue the analysis after the attack and investigate further. Take the model checking perspective; once I find an attack, I may wonder whether there is another attack in the history. Perhaps if I continue the history after I find the first attack, I'll find that in fact there's another attack whereby I can retaliate against who mounted the first attack. Or, from a theorem proving perspective, I could assume the existence of the first attack and wonder whether I could have the same attack mounted twice by two different sets of bad guys. In this case, I wouldn't really know who the real attacker is, whether the set that mounted the first attack mounted the other one, and if you have the same attack mounted by more than one set of people then it gets more complicated.

Another related issue is why I should do this. When we find an attack nowadays, it means that the protocol designer failed. A successful attack means the protocol is flawed, we need to redesign, to go back to the drawing board and start from scratch, or fix the protocol somehow. That's what we normally think. But suppose you are the bad guys; you consider mounting an attack to be less convenient according to some criteria if you know that there is a sufficient threat that you can be retaliated against. Would you still mount the attack? In simple words, would I steal a thousand pounds from you today if there's a realistic threat that tomorrow some of you will steal two thousand pounds from my own account.

Perhaps if we find that a protocol allows this notion of retaliation, we could still live with a protocol that we consider flawed. We know that the bad guys should be kept from mounting the attack by balancing the benefits of the attack with the threats of the possible retaliation against them. So the issue I am addressing here is whether it's really convenient to attempt to mount attacks, and clearly in this scenario we should allow principals to change their behaviour.

What I've seen nowadays in protocol verification papers is that we tend to give people a fixed role so if they're good, they're going to be good forever, and if they're bad, they're going to be bad forever. But, realistically, I might decide to change my mind and become a bad guy if that's convenient. And on the other hand we may find the bad guys repent and change their minds. So we should allow for principals to change their behaviour. Certainly this notion of retaliation wouldn't have made sense if we didn't allow principals to change their behaviours.

So having said this, we all know those middle person attacks, where we see this middle person who can perhaps steal money from A's account if the B's are bad. However several people last year at the workshop showed that if this is possible, then retaliation is possible too; namely, B basically can retaliate against C, and steal money from C's account in the case where C mounted the first attack. So this is an example of retaliation.

So that's the general view, we have good guys, bad guys who are certainly trying to subvert the protocol, and ugly guys who normally conform to the protocol but might collaborate with the bad guys. Crucially, as I said, they can change their behaviour. One principal cannot be in two different sets, but can jump from one set to another as the protocol goes on.

We should all be acquainted with the notion of trace, it's like the log, or history, of the network (a list of messages, or emails, or similar). Then we also have the notion of projection over a set of agents. The projection only includes the details of the trace that pertain to that set. An attack is some function of the trace, and of the people involved.

This kind of verification setting is a little more formal. We consider a protocol vulnerable against an attack if there is a trace on which the property A holds. On the other hand, theorem proving would try to prove that there is no such trace. But this will get more complicated when we try to formalise retaliation. The protocol allows retaliation of an attack A by some set of bad guys, no matter how we take a trace and a triple of good applications. When we have this attack A, then we can extend the trace, extend the history of the network, and find a new triple agents G prime, B prime, U prime, such that the attack holds on the extended trace for the new triple of agents.

So what does this mean? Crucially, you see that the new bad guys, B prime, are people who earlier were either good or ugly, and those who were bad originally are now either good or ugly. So basically they changed roles. What I'm saying is, if there was an attack for the original set, then there is another attack if we continue the history of the network, mounted by the new bad guys. So you see the new bad guys are exactly those who were good, so basically those that were good are retaliating directly. Or, if the new bad guys include some of the original good guys, then we think of combining retaliation, so only some of those who were good are now retaliating. And if the new bad guys are only those who were ugly then we speak about retaliation.

John Ioannidis: Why is the retaliation something we get from just the same attack? If a good guy becomes a bad guy, does he carry state knowledge from when he was a good guy? Does it really matter that it's the same machine, or the same person?

Reply: Well I don't think it does really matter at this abstract level. I'm introducing the notion that if you change roles then you might mount an attack you wouldn't originally have mounted because you were playing good. I'm not being too specific at this level, I'm just introducing the notion that people can change their mind and continue the history and mount attacks that they wouldn't have done otherwise.

John Ioannidis: So in fact if I'm the key and I'm good, then I can use that key to do bad? So in that sense I become bad.

Reply: You might change your mind. Don't you find that realistic?

John Ioannidis: It's not a matter of realism, it's a matter of what one says.

Reply: I know, that's something we should think about. For the moment, clearly if you mount the same attack, it's retaliation for sure. I thought of using another symbol, but then we may want to clarify the relations between the original attack, the first attack, and the second attack. That is certainly a good point.

Matt Blaze: I think this is a useful way of thinking about some kinds of cryptographic protocols, but probably not all. We've had the luxury in cryptography of being able to attempt to design systems that we can say with very high confidence won't have an attack. With this notion of even a certificational weakness being sufficient to just knock the cryptographic scheme out of consideration, there's this tendency to assume that we should make all of our protocols meet the same standard, but of course no security protocol in the physical world is so securely air tight as we demand of our cryptographic protocols. One example where this starts to move into cryptographic protocols is denial of service. Even the semantics of a denial of service attack aren't completely clear, the boundary between conducting a denial of service attack and just normal service is entirely, in some cases, one of scale, rather than one of specific action. It's a specific fine line between this person conducting an attack, and that person just demanding quite a bit of service. So what that suggests to me is that where your notion of retaliation will lead, in a system where the attack is really defined by scale, is that it's really the more resources you consume the higher the probability that something bad will happen to you. It seems to be stirring up trouble.

Mads Dam: Might the retaliation have a completely different nature from the original attack? So you're using the same A at two places?

Reply: Then if we have a completely different attack we may want to decide if we still consider that retaliation or not, or another property worth investigating. That is completely up to us.

Alec Yasinsac: If we're talking about retaliation then surely the victim has to be aware that the attack took place.

Reply: Well, might become aware at some stage.

Alec Yasinsac: But how do they become aware given the nature of the attack if somebody's masquerading as a third party for example?

Matt Blaze: Could I respond to that?
Here's an example where you wouldn't have to know. Imagine a network itself instigates a denial of service attack. Every time you send a SIM packet, there is a one in two million chance that your network connection will drop for an hour (you know, that's probably roughly the service that I'm getting from my DSL provider anyway). However, if I start to engage in a denial of service attack where I'm sending out way more than my share of SIM packets, there's a chance that my network connection will go down, and the victim doesn't need to be aware of that.

Alec Yasinsac: From the point of view of defending, when using retaliation as an offence against attack, how do you know when you are under attack due to retaliation? There needs to be some awareness.

Matt Blaze: Well it might be a property in the protocol.

Birgit Pfitzmann: I would have said it's a bit like these optimistic fair exchange protocols. But like you said, you need some procedure for the honest guys, some test for them to see whether everything's still OK, and some procedure for the attack that they're supposed to mount.

Bruce Christianson: There are certain things that a good participant is allowed to do when an attack is detected. What you hope, for example, in the case of double-spenders, is that it's precisely the violation of a protocol by a bad person that makes him vulnerable to the counter-attack.

George Danezis: Yes indeed. If you are talking about retaliation, it is very important to talk about provocation. In your proof there is nothing that says that the original good guy can prove that something bad happened to him which therefore allowed him to turn bad and retaliate. This is a problem, because if it's not in the protocol that you can only retaliate if something bad has happened, as a direct consequence, then everybody will just retaliate.

John Ioannidis: I think we're ready to move on here, we need to go back to some protocol terms.

Reply: OK, so I have two examples, based on Lowe's attacks mounted by these same bad buys against the good guys. This is the second attack, mounted by someone who was originally good and has now become bad. This is what I would call direct retaliation, because those who were attacked are actually now attacking a little more formally. If we negate the retaliation property like this, once we have a trace with an attack, no matter how we extend it, we're never going to have an attack on an extended trace, no matter how many triple, good, bad, or ugly agents are involved. This should be suitable for model checking.

Pekka Nikander: I'm having a slight problem with the detail here. You seem to be suggesting that in addition to the good guys, or the ugly guys, and the bad guys, the bad guys are good guys, or ugly guys. I'm not sure if that would be the case. Why would the bad guys, if they are trying to interrupt communications, have the same goals as the good guys initially?

Reply: I can certainly imagine a situation when this is possible.

Pekka Nikander: Sure, but I think it's a pretty limited case. From my point of view, it tends to be that the good guys and bad guys are different. The retaliation isn't a different kind of attack against the bad guys playing honest, or at least in an ugly way. This doesn't seem too realistic to me.

Reply: OK, I take the comment. They certainly try to investigate a much subtler property than we're used to. I still find this realistic, although perhaps it's not so common. But it's still a subtle detail that might be of interest at some stage.

This definition makes sense to use. We're saying if we can't continue the history and mount another attack then there's no retaliation. We might prove that a protocol conforms to this property, and therefore that means that if we have an attack that is completely flawed, then we need to redesign because we cannot keep the bad guys from using this notion of retaliation.

Then we have this notion of detection which I am sure is going to raise lots of discussion. Take a trace in the triple of good, bad, ugly guys, and suppose there is an attack. No matter how I extend this trace, such that I have the same projection on the good guys I still can be attacked. So basically, suppose you find your window smashed, and you want to take these people to court. You need to prove that it was them, and show that no matter how the history goes your logs still always show the attack. This is the nature of this definition. But if the bad guys want to help themselves then they should show that there is another history where they actually didn't mount the attack. So if you take a trace where there is an attack by some ugly guys, there is another trace where the bad guys actually didn't mount the attack. I bet if you take that sort of evidence to court, then it would be a big help for the people whom you're calling bad guys.

Birgit Pfitzmann: Do you know who the good guys are? I mean, it should work, if you are talking about the trace projected to all the good guys. But typically, if you go to court, the court doesn't know who the good guys and the bad guys are. The court can't just say, OK, all the good guys come up and join that trace, and then we'll look at what attacks are in there. So the court has to judge just on things it's being shown but without knowing the distinction. That's why typically this accountability problem was actually specific, showing protocols where it said which was which, and then leaving it to the court to decide what they are supposed to do about it. But only on the court's view.

Reply: [Long Pause] Right, and how about suspicion? So we have suspicion of an attack if, no matter how we take a trace, we can always extend it in such a way that there's another set of bad guys and ugly guys who mounted the same attack. If we can always find more sets of bad guys mounting the same attacks, then we can't really tell who mounted the attack, so the attack would hold for any places we are safe from bad guys. Does that make sense?

Bruno Crispo: You mentioned there is a suspicion with the bad guys?

Reply: When I say there's suspicion, we're talking about finding the attackers who mounted the attack. You could think of the detection where you track back the actual attackers.

Partha Das-Chowdhury: In some cases detection would also be a problem, taking the example of the digital cache. If you issue credentials using that system, and if there is a transfer of credentials, then it certainly isn't valid like your scenario. In those cases how do you detect that there has been a violation of what was meant to happen because things were issued anonymously and so nobody can be sure to detect the transfer of credentials?

Reply: So you're saying it wouldn't detect the first attack anyway.

Partha Das-Chowdhury: Exactly, and that would be a problem.

Reply: Sure, I can't imagine that's a new case, definitely. And then there is the notion of no suspicion which is in my opinion even more controversial. Here it's clearly the negation of what I just showed you, and I'm trying to figure out what it means myself. So I have a trace, with an attack, but again I have another trace whereby no matter how I take another set of bad and good or ugly guys, I don't find the attack. So I don't have the attack.

What do we get out of this?

Birgit Pfitzmann: Basically this gives a not B and you are the one to be attacked.

Reply: Yes, that's all you know, so you cannot suspect anyone else. So the question here is whether this actually overlaps with the notion of detection.

N. Asokan: It wouldn't make suspicion?

Richard Clayton: Surely it's not suspicion if the same trace leads to a situation where there's no attack.

Reply: I guess that's so. Anyway this situation is rather new and I'm still trying to figure out what it means.

The conclusion of my talk is that there's certainly life beyond attacks, and it might be worth investigating. I suppose that whatever you have, like a man in the middle attack as a new version of Lowe against Needham and Schroeder, it should always be possible to retaliate against whoever mounts the attack. As I said, if we prove a considerable amount of possible retaliation for a protocol, then we might still win even against possible attacks on that protocol.

The properties of traces certainly get more complex than we're used to in the verification community.

Kamil Kulesza: I like the idea but I'm afraid that what you are trying to build will boil down to some sort of conflict order, and this will bring the analysers to the next level. I'm afraid we'll end up with something like, let's say, conflict modelling, or war gaming. The problem is that as far as I know at this stage we're not able to offer something this secure. Basically we can discuss statistical, or stochastic, properties of such a model, but what you are saying...

Reply: I see your point, but I'm happy with theorem proving, and I should think that proving a property where two quantifiers, even if they're alternative quantifiers, seems reasonable.

Kamil Kulesza: But the problem is this shadow of uncertainty. For instance, whether I can retaliate, and when, and the problem of deception. It's more like waging the war than proving, because there are no clear rules.

Brian Monahan: You've filled in the framework here for identifying partners in these protocol sets, but you don't seem to have fixed them as actually being good or bad, that does worry me. I would expect more comments on the concept of evidence, of the way of coming to those judgements, what do you have to accumulate? And essentially you are coming up with this game scenario which is inherently open-ended, and therefore not necessarily susceptible to a formal theorem proving approach. Maybe an open model approach would be more appropriate.

Reply: Good point. This model comes from a generalisation of spy theory. We don't just see one attacker, if people don't collude with each other. They might work for themselves and then we get more trouble when you talk about the problem of identifying someone as an attacker.

Brian Monahan: What about services, where you've got multiple protocols, or multiple runs, being played against each other? This is a Byzantine analysis, or a distributed system, it's clearly open ended and I don't see how you can find a nice problem and bring it back down to a large problem which can be attacked in a nice theorem driven manner. It would be nice to be able to do that but your automation quantifiers are a major problem.

Reply: Yes, I imagine that's how it goes. [Laughter] So we have come to the end. Certainly we have a problem of adapting mechanised source support, but perhaps next year we'll tell you how it goes.

Towards Flexible Credential Negotiation Protocols

Piero A. Bonatti[1], Sabrina De Capitani di Vimercati[2], and Pierangela Samarati[2]

[1] Dip. di Scienze Fisiche – Sezione di Informatica – Università di Napoli "Federico II", Via Cinthia, I-80126 Napoli, Italy
`bonatti@na.infn.it`

[2] Dip. di Tecnologie dell'Informazione – Università di Milano, Via Bramante 65, I-26013 Crema, Italy
`{decapita, samarati}@dti.unimi.it`

Abstract. Accessing information over the Internet has become an essential requirement in modern economy, and unknown parties can come together on the Net and interact for the purpose of acquiring or offering services. The open and dynamic nature of such a scenario requires new protocols allowing parties to communicate and enforce security specifications related to access control conditions to be fulfilled. In this paper we discuss several issues to be investigated in the development of these flexible interaction protocols.

1 Introduction

Today's Globally Internetworked Infrastructure connects remote parties through the use of large scale networks, such as the World Wide Web. Execution of activities at various levels is based on the use of remote resources and services, and on the interaction between different, remotely located, parties that may know little about each other. In such a scenario, traditional assumptions for establishing and enforcing access control regulations do not hold anymore. For instance, a server may receive requests not just from the local community of users, but also from remote, previously unknown users. The server may not be able to authenticate these users or to specify authorizations for them (with respect to their identity). The traditional separation between *authentication* and *access control* cannot be applied in this context, and alternative access control solutions should be devised. Early approaches departing from this assumption proposed associating authorizations with keys rather than with users' identities. This family of *trust management systems* (e.g., PolicyMaker [3], Keynote [2], REFEREE [6]) use credentials to describe specific delegation of trusts among keys and to bind public keys to authorizations. While these approaches provide an interesting framework for reasoning about trust between unknown parties, assigning authorizations to keys may result limiting and make authorization specifications difficult to manage.

An alternative promising approach is based on the use of *digital certificates* (or *credentials*), representing statements certified by given entities (e.g., certification authorities), which can be used to establish properties of their holder (such as identity, accreditation, or authorizations) [8]. Credential-based access control makes the access decision of whether or not a party may execute an access depend on properties that the party may have, and can prove by presenting one or more certificates (authorization certificates in [3] being a specific kind of them). The development and effective use of credential based-access controls requires tackling several problems related to credential management and disclosure strategies, requiring the design of new protocols for parties to communicate what their requirements (information they request or disclose) to their counterpart. Several researchers have addressed the problem of establishing a Public Key Infrastructure (which is at the basis of credential-management); managing credentials and credential chains and developing strategies for automated trust negotiation, that is for determining the credentials to be required and released when interacting with other parties [11,12,14,15,16,10,1,4].

The successful use of credentials for enforcing access control, and the consequent application of all the different trust management strategies that can be though of, requires a fundamental problem to be solved: *parties must be able to communicate to others 1) what credentials/properties they enjoy and 2) what credentials/properties they require from the counterpart in order to grant them access to specific requests.* Since all these features are application-dependent and policy-dependent (both on the client's and on the server's side), the traditional protocols results too rigid and limited. It seems then useful to perform some automatic *inferences* on the parties' policies to achieve the necessary flexibility.

2 Open Issues Towards Flexible Negotiation Protocols

The satisfaction of the requirements stated above, requires complementing traditional protocols with high-level flexible interaction protocols, enabling credential negotiation, credential understanding and explanation generation and handling, introducing a form of *negotiation* between clients and servers. In particular, it is important to devise how *security specifications (access rules) should be stated* in a way suitable with the new scenario; how they should *be translated for their communication* to the counterpart; and how the *parties communicate and reach consensus* in the transaction execution. Moreover, just like traditional security protocols, these new interaction protocols should give certain security guarantees (e.g., termination, correctness, no improper information disclosure).

Among the issues to be investigated, there are the following.

– *Ontologies.* Due to the openness of the scenario and the richness and variety of security requirements and credential-based properties that may need to be considered, it is important to provide parties with a means to understand each other with respect to the properties they enjoy (or request the counterpart to enjoy). Therefore, common languages, dictionaries, and ontologies must be developed [7].

– *Client-side restrictions.* The traditional distinction of client and server becomes loose as every party can behave as either a client or a server depending on the context. Also, while it is true that for each specific interaction there can be a clear distinction between the two roles, one assumption does not hold anymore: it is not only the server that establishes regulations. In traditional access control systems, clients need only to supply their identity (together with a proof for it), and servers need to support an access control system (i.e., include a system for stating and enforcing rules regulating access to their resources). Emerging scenarios require this latter ability to be supported by clients as well. Indeed, a client may—like a server—require the counterpart to fulfill some requirements. For instance, a client may be willing to release an AAA membership number only to servers supplying a credential stating that the travel agent is approved by AAA.
– *Credential-based access control languages.* Flexible and expressive languages able to express and reason about credentials need to be developed. Simple 'tuple-like' authorizations are obviously not sufficient anymore and richer languages are needed. Such languages may contain constructs to control negotiation.
– *Access control evaluation and outcome.* Users may be occasional and they may not know under what conditions a service can be accessed. Therefore, in order to make a service "usable", access control mechanisms cannot simply return "yes" or "no" answers. It may be necessary to explain why authorizations are denied, or - better - how to obtain the desired permissions.
– *Policy communication.* Since access control does not return a definite access decision, but it returns the information about which conditions need to be satisfied for the access to be granted, the problem of communicating such conditions to the counterpart arises. To fix the ideas, let us see the problem from the point of view of the server (the client's point of view is symmetrical). The naive way to formulate a credential request—that is, giving the client a list with all the possible sets of credentials that would enable the service—is not feasible, due to the large number of possible alternatives. In particular, the precise nature of the credentials might not be known in advance (as it happens with chains of credentials), and in the presence of compound credential requests such as "*one* ID *and one* membership certificate from a federated association", there may be a combinatorial explosion of alternatives, as each individual request can potentially be fulfilled in many possible ways.
– *Flow control.* Negotiations should not disclose "too much" of the underlying security policy, which might also be regarded as sensitive information. For instance, suppose that a given service is to be made accessible only to users who satisfy all the following conditions: 1) are registered at the server, 2) are US residents, 3) are members of a partner association. Instead of communicating all such requirements to the client, and therefore unrestrictedly

disclosing the whole policy, the server could first ask the counterpart for her login name (prerequisite); if she is not registered, there is no reason to proceed further. The situation is particularly complicated since the information against which access control rules is evaluated can be communicated in some cases and be considered sensitive in other cases. For instance, in the example above, the list of partner associations can be considered sensitive (and therefore used only for control at the server), or public (and therefore communicated to the user beforehand).
– *Negotiation strategy.* Credentials grant parties different choices with respect to what release (or ask) the counterpart and when to do it, thus allowing for multiple trust negotiation strategies[16]. For instance, an *eager* strategy, requires parties to turn over all their credentials if the release policy for them is satisfied, without waiting for the credentials to be requested. By contrast, a *parsimonious* strategy requires that parties only release credentials upon explicit request by the server (avoiding unnecessary releases).
– *Negotiation success.* The negotiation procedure should succeed whenever the policies of the two parties allow it. This aspect requires ensuring that the policy enforced by a party leads to the same result as the one communicated to the counterpart obtained from the original by removing sensitive information and conditions that could be evaluated only locally. Also, the interaction protocol between the parties must be guaranteed to terminate.

3 Conclusions and Perspectives

Standard kinds of protocols will keep on playing a fundamental role in the key handling infrastructure, but they will have to be complemented by higher-level, flexible interaction protocols, enabling credential negotiation, credential understanding and explanation generation and handling. The technologies needed for such an enhanced infrastructure involve aspects of knowledge representation and reasoning, and should be made practical by adopting knowledge compilation techniques and fast, lightweight inference mechanisms. In this paper we have illustrated some issues to be investigated in this direction.

References

1. C. Altenschmidt, J. Biskup, U. Flegel, and Y. Karabulut. Secure Mediation: Requirements and Design. In *Proc. of the 12th IFIP WG11.3 Working Conference on Database and Application Security*, Chalkidiki, Greece, July 1998.
2. Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The Role of Trust Management in Distributed Systems Security. In *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*. Springer Verlag – LNCS State-of-the-Art series, 1998.
3. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *Proc. of 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, May 1996.

4. J. Biskup and Y. Karabulut. A Hybrid PKI Model with an Application for Secure Mediation. In *Proc. of the 16th Annual IFIP WG 11.3 Working Conference on Data and Application Security*, King's College, Cambridge, UK, July 2002.
5. P. Bonatti and P. Samarati. A Unified Framework for Regulating Access and Information Release on the Web. *Journal of Computer Security*, 2002. (to appear).
6. Y-H. Chu, Joan Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. REFEREE: trust management for Web applications. *World Wide Web Journal*, 2(3):706–734, 1997.
7. Chen-Chuan K. Chang and Hector Garcia-Molina. Mind Your Vocabulary: Query Mapping Across Heterogeneous Information Sources. In *Proc. of the 1999 ACM-SIGMOD*, pages 335–346, 1999.
8. R. Housley, W. Ford, W. Polk, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, rfc 2459 edition, January 1999. `http://www.ietf.org/rfc/rfc2459.txt`.
9. S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible Support for Multiple Access Control Policies. *ACM Transactions on Database Systems*, 26(2):18–28, June 2001.
10. N. Li, W.H. Winsborough, and J.C. Mitchell. Distributed Credential Chain Discovery in Trust Management. In *Proc. of the Eighth ACM Conference on Computer and Communications Security*, Philadelphia, PA (USA), 2001.
11. K. E. Seamons, W. Winsborough, and M. Winslett. Internet Credential Acceptance Policies. In *Proceedings of the Workshop on Logic Programming for Internet Applications*, Leuven, Belgium, July 1997.
12. M. Winslett, N. Ching, V. Jones, and I. Slepchin. Assuring Security and Privacy for Digital Library Transactions on the Web: Client and Server Security Policies. In *Proceedings of ADL '97 — Forum on Research and Tech. Advances in Digital Libraries*, Washington, DC, May 1997.
13. M. Winslett, N. Ching, V. Jones, and I. Slepchin. Using Digital Credentials on the World-Wide Web. *Journal of Computer Security*, 1997.
14. W. Winsborough, K. E. Seamons, and V. Jones. Automated Trust Negotiation. In *Proc. of the DARPA Information Survivability Conf. & Exposition*, Hilton Head Island, SC, USA, January 25-27 2000. IEEE-CS.
15. T. Yu, X. Ma, and M. Winslett. An Efficient Complete Strategy for Automated Trust Negotiation over the Internet. In *Proceedings of 7th ACM Computer and Communication Security*, Athens, Greece, November 2000.
16. T. Yu and M. Winslett K.E. Seamons. Interoperable Strategies in Automated Trust Negotiation. In *Proc. of the Eighth ACM Conference on Computer and Communications Security*, Philadelphia, PA (USA), 2001.

Towards Flexible Credential Negotiation Protocols

(Transcript of Discussion)

Pierangela Samarati

Università di Napoli “Federico II”, Italy

The problem we are looking at is how you specify, enforce, and communicate security policies in a world where people, computers and objects move, change identities, present themselves without identification or authentication and still want to interact and access services.

Brian Monahan: Couldn’t you send the policies to the client, get them to do the computation and send back the proof, which is then used, so then you can check? That’s also nicely private as well.

Mads Dam: Firstly it’s dangerous to send off the policies which regulate access to the client. Second point, how do you make sure that the information that you sent away doesn’t reveal information that you don’t want to be revealed by a query?

Virgil Gligor: You can always map it into a score.

Ross Anderson: Absolutely, a credit rating.

Richard Clayton: If you are willing to disclose your business practice.

Ross Anderson: You can borrow twenty thousand pounds from me if your credit score is greater than three thousand two hundred. What’s the algorithm? Not telling you, it’s a secret! [Laughter]

Brian Monahan: But you still give them some information.

Frank Stajano: So wouldn’t this be a good place to use some secure multiparty computation where nobody gets the result? Then neither of the parties would be able to get the kind of answer that they may have and agree. So, if it’s false then nobody knows what the other one was putting in, but if it’s true then they know about everything.

Richard Clayton: It depends how you do it. If you’re a shopper and you go into a shop, you don’t tell the person behind the counter what your income and outgoings are so you can see whether or not you can have a new sofa, a third party does that, the credit rating organisation. However, you still have to give your personal information to the credit rating people, so although it may be a third party to the shop, it’s not a third party as far as you’re concerned.

Frank Stajano: If you're using a third party, I would suggest some protocol where they'd be able to do the computation but the party who gets the result is the one that actively runs services.

Ross Anderson: There is a strategic tension here. Usually what you're trying to do is price discriminate, and this is not what the customer wants, unless he's very poor. If the customer is not poor, then you do not have a collaborative environment, you have a hostile environment. That's the fundamental problem here. The client wants to appear to be poor, and the server wants to find out if he's lying so he can charge him more.

Larry Paulson: How to get a bargain if you're shopping in a foreign country, assuming you know the native language. Ask in English: "how much is this shawl?" The girl who's serving doesn't know how low you will go, so she asks the boss in her native language the lowest price he would take for it. Then you can bargain down to that price.

Matt Blaze: I'm not sure that we're not making a mistake here. The examples in this discussion have been security protocols that are partially implemented as relatively slow, off-line human intervention processes; such as applying for a mortgage, and so on. There's considerable cost for the client so you can't repeat it over and over again. The mechanisms that you'd actually be describing are, in contrast, on-line services, where the costs of reissuing a request from a client are in fact very low and you could probably do it over and over again. I understand how keeping your policy secret in the slow human-intervention type of protocol may be a feasible thing to do, but I'm less persuaded in the actual examples you're giving that it is feasible to keep aspects of the public policy secret. This is because the client can issue so many requests with different parameters, that they may be able to quite accurately reverse engineer your policy. So one has to be very careful in the way one responds to the queries.

Reply: There are occasions when the policy is sensitive, but the problem here is that what is sensitive is the information I use to evaluate the policy. And it's true you can have many attempts to find out my policy, but for instance, if I have a favourite customer to which I give a bonus of a hundred dollars, and I don't want to tell the others because they might get upset, this is policy information. The fact that I'm giving the bonus is certainly sensitive information. By making many requests you find out my policy on each one of the customers, so you won't be able to pretend to hold that property if you don't.

Matt Blaze: But again one of these heavyweight protocols, where there is actual money involved in doing the transaction, results in a high cost to the client. I think that in most of the examples, where we're doing completely automated processing in an on-line service, the ability of the client to do tweaking of the parameters, and essentially differential analysis of the policy, is quite good.

Ross Anderson: You could try and detect the differential attacks. If you're running a website, you might see somebody drop out of buying a book a thousand times until he gets a “product sent on demand” code. Once you’ve received a hundred requests from the same IP address you could slow down the service and give him a ten second response time.

Bruce Christianson: The response of the mechanism might not be to grant access; it’s to grant access at a particular price, subject to particular conditions that depend on the details.

Matt Blaze: Those kind of systems may not be as frequent as we like to believe they are, you know. There are, for example, a number that don’t only want to do business with people who already have accounts, they want to be able to encourage you to find out what the price is to encourage you to open one.

Reply: Yes, these are occasions when actually you want to publicise your policy. For example, if you spend one thousand dollars a year, then this is a policy you want to publicise. However, there are occasions when you don’t want to publicise the policy. Suppose you have access conditions. You need to be older than 21, European, etc. Now I can disclose the whole policy but maybe they were better off asking you the conditions one by one. I’ve evaluated and made assumptions that the client would get upset if he answers things one by one. He wants the service right away, so either he gets the service, or he gets upset.

Pekka Nikander: But you are again speaking about the cost of a human doing something, which is a cost not monitoring.

Reply: It doesn’t matter if it’s a human or a program, the thing is I don’t want my agent or my program to give information if I don’t have the answers back. If you need to give reassurance to the customer, you have to disclose the policy all at once.

Matt Blaze: I don’t believe the policies are sensitive as often as many of the holders of policies assume.

Reply: OK, but what are the policies for? Are they simply conditions or do they also act differently with each deal?

Matt Blaze: It’s frequently the case that I’ve heard the objection from, for example, people designing network security policies, that they want to keep their security policy secret, when in fact the desire to keep it secret does not stand up to any scrutiny whatsoever. Nothing is added by keeping the policy secret in many cases.

Virgil Gligor: But I think there is a privacy implication here as well, it’s not just keeping it secret because you make it more robust.

Reply: Yes.

Bruce Christianson: There’s client secret consideration on both sides. There’s often a regulatory framework that says you mustn’t ask for information if you don’t need it. If I don’t need to know your age and I just want to know that you’re over 18, I mustn’t ask you your age. But equally the server may have things that they know which aren't reasonably required and they don't want to have to reveal, sometimes they just want to keep it a secret whether their policy is a secret or not.

Virgil Gligor: So the problem with all these protocols is that you run them and some information is leaked. The question is, what do you do about it, and should we care about it. Matt is saying that very often the answer is no.

Reply: I guess it depends on the information that's leaked. It doesn't address the assumption that maybe clients want assurance before they give information.

Matt Blaze: Oh yes, absolutely.

Frank Stajano: You were asking earlier on what should be done on the policy and what shouldn't. It may be a minor point. I believe that given your example we should reject from the policy things such as the year of the request. This is not something that has anything to do with policy, it has nothing to do with whether if it's feasible or not to satisfy the request. I think it is wrong to put inside the policy things such as these accidents, because policy should be concerned with whether I will allow you access, not guarantees that the request is fresh.

Bogdan Popescu: So somebody cannot only access anything older than five years, he has to be able to access everything?

Ross Anderson: Well what's often happening in practice is that companies want to collect vast amounts of data via their customers for marketing purposes, and so their motives are actually the other way round from what you're proposing. Up to 99 percent of customers will choose to get a supermarket store card in return for half-a-percent off their shopping. They fill out forms telling the supermarket everything about themselves and linking all their purchases. This is the security policy that most electronic commerce vendors, and many High Street ones, actually operate: we give the discount if you tell us everything about yourself. Of course you can always lie.

Reply: But when you are on the Internet, and not in your supermarket, you don't quite know how the information that you send is stored, or how it's going to be distributed, or where.

Man-in-the-Middle in Tunnelled Authentication Protocols

Extended Abstract[*]

N. Asokan, Valtteri Niemi, and Kaisa Nyberg

Nokia Research Center, Finland
{n.asokan, valtteri.niemi, kaisa.nyberg}@nokia.com

Abstract. Deploying a new security protocol is expensive. This encourages system designers to look for ways of re-using existing infrastructure. When security protocols and components are re-used, it is critical to re-examine the security of the resulting system as a whole. For example, it has become a standard paradigm to run a legacy client authentication protocol within a secure tunnel. The commonest example of such composition is the use of HTTP authentication inside a TLS tunnel.

In this paper, we describe a man-in-the-middle attack on such protocol composition. The vulnerability arises if the legacy client authentication protocol is used both in tunnelled and untunnelled forms. Even when the client authentication protocol and the tunnel protocol are both secure, composing them in the customary manner results in an insecure system.

We propose a solution to this problem by using a cryptographic binding between the client authentication protocol and the tunnel protocol.

1 Introduction

When new services and applications are developed, their security requirements and trust assumptions may necessitate designing and deploying new security protocols. However, deploying a new security protocol is expensive. This encourages system designers to look for ways of re-using existing infrastructure. One form of reuse is to build on existing protocol components or frameworks so that the need to deploy new software is reduced. But the most difficult aspect of deployment is the cost of provisioning initial security associations, especially to end user devices. Consequently, there is considerable pressure to reuse security protocols and security context databases beyond their originally intended purposes and environments.

Re-use is generally considered good practice. However, when security protocols and components are reused, it is critical to re-examine the security of the resulting system as a whole. This is particularly important when different components of the composition have different endpoints, either in different layers in the same entity, or different entities altogether. The fact that there are no

[*] An earlier, longer version of this work appeared as a research report [5].

easy-to-use tools or methodologies to verify the correctness of security protocols makes protocol re-use a risky task.

Faced with this difficulty, designers who needed to re-use legacy client authentication protocols and/or security associations have been customarily using an obvious approach to secure legacy client authentication protocols when there is a need to re-use them in a new scennario. They define an authentication protocol as a combination of two protocols: The legacy client authentication mechanism is run inside a secure tunnel. The most common example of this type of construction is the combination of a server-authenticated TLS tunnel [9] and the HTTP Digest Authentication mechanism [11]. Shortcomings of the legacy client authentication protocol in the new environment can apparently be offset by the protective tunnelling. For example, HTTP Digest Authentication may be based on a simple user-name/password; but the TLS tunnel will protect the password from any eavesdroppers. At the same time, the legacy authentication method may continue to be used in legacy environments. Therefore, at first glance, secure tunnelling appears to be a reasonable approach. This is why, as described in Section 2, this type of protocol composition has been widely used when there is a perceived need for reuse of legacy authentication mechanisms.

Unfortunately, when such a legacy client authentication protocol needs to be run both in the legacy environment as well as in a tunnelled environment, a man-in-the-middle attack becomes possible. Even when the client authentication protocol and the tunnel protocol are both secure, the composing them in the customary manner described above, results in an insecure system. In this paper, we describe the attack, and discuss how it can be avoided by cryptographically binding the inner and outer protocols. If the inner client authentication protocol produces a suitable key, this binding *does not require any changes* to the inner protocol. This is a very important requirement because the whole point of using legacy protocols is that they are already widely deployed.

In section 2 some background information is provided by highlighting the common tunnelling approach and using PEAP as an example to describe how this is instantiated in practice. Then the man-in-the-middle attack is described in section 3. Ways of removing the vulnerability are presented in section 4. The use of weak authentication methods is discussed in section 5. The implications of this attack are discussed in 6. The conclusions are summarized in section 7. Finally, a brief status update is provided in section 8.

2 IETF Drafts on Tunnelled Authentication Protocols

2.1 The General Model

In this section, we present a general description of how tunnelled authentication protocols are usually constructed. Although we discuss examples that use extensible authentication protocol (EAP) as the inner client authentication protocol, the discussion is applicable for any tunnelled authentication protocol.

EAP, described in RFC2284 [17], is a standard framework for client authentication protocols. By using EAP in a system, it is possible to enable the system

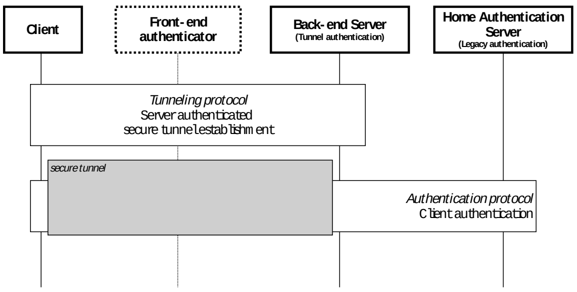


Fig. 1. Customary construction of a tunnelled authentication protocol

to support a number of legacy authentication schemes, including smart cards, Kerberos, public key mechanisms, One Time Passwords, cellular authentication mechanisms like GSM [14] or UMTS AKA (Universal Mobile Telecommunication System, Authentication and Key Agreement protocol) [3], and many others.

EAP is run between a client and a server. We call this server a *backend server*. There may be a a *front-end authenticator* between the client and the backend server. The front-end authenticator will simply forward authentication messages to the backend server. The backend server may use yet another server to help authenticate the client. We call this third server the *home authentication server*. Use of EAP allows new authentication methods to be developed without requiring deployment of new code on the front-end authenticator. The front-end authenticator acts as a "pass-through", and need not understand specific EAP methods.

Recently new protocols have been proposed in the Internet Engineering Task Force (IETF) for running EAP inside a server-authenticated tunnel. Examples of such protocols are PIC [23], PEAP [15], EAP-TTLS [12], POTLS [20], and most recently SLA [13], which led to the mechanisms for supporting legacy authentication methods in IKEv2 [10]. Some of these protocols, like PEAP, are motivated by a wish to correct perceived weaknesses of EAP, such as the lack of user identity protection and lack of a standardized mechanism for key exchange [15]. Others like PIC, POTLS and SLA are motivated by a desire to re-use legacy authentication credentials and databases for new applications. All of these new protocols are constructed in the same basic manner. First, a secure tunnel is set up using a suitable protocol like TLS. In most cases, the tunnel is also server-authenticated. Then the client authentication protocol is run inside this tunnel. This general model of this customary approach is illustrated in Figure 1. Other forms of tunnelled authentication protocols, such as HTTP Digest authentication inside a TLS tunnel, also conform to this general model.

2.2 Protected EAP

Protected EAP (PEAP) [15] is an example of the type of composition described in Figure 1: it wraps the EAP protocol messages within TLS [9]. It claims to provide user anonymity and built-in support for key exchange.

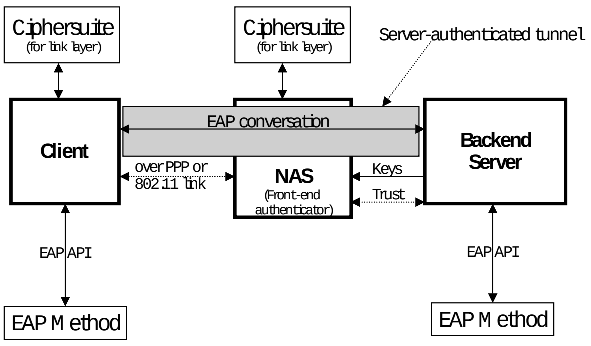


Fig. 2. Relationship between EAP client, back-end authentication server and NAS in PEAP [15]

The relationship between the EAP peer (client), front-end authenticator, known as the "network access server" (NAS) in PEAP, and a backend server, known as the "back-end authentication server" in PEAP, is depicted in Figure 2. As described in the figure, the EAP conversation "passes through" the NAS on its way between the client and the back-end authentication server. While the authentication conversation is between the EAP client and the back-end authentication server, the NAS and back-end authentication server need to have established trust for the conversation to proceed.

The client and the back-end server first set up a TLS channel over EAP. The client authentication protocol messages between the client and the back-end server are encrypted and integrity protected within this TLS channel. The NAS does not have knowledge of the TLS master secret derived between the client and the back-end authentication server, and cannot decrypt the PEAP conversation. The back-end server derives master session keys from the TLS master secret via a one-way function and conveys them to the NAS which can then use these session keys to protect subsequent link-layer communication between it and the client. The session key is transported from the server to the NAS using e.g. Radius [22] or DIAMETER [8] attributes.

In a recent Internet draft, EAP SIM GMM authentication [7], an application of PEAP to GSM authentication was presented. The same approach can be used to combine PEAP with AKA which is the client authentication mechanism in the third generation UMTS networks. The message flow in PEAP with EAP AKA client authentication in the context of wireless local area network (WLAN) access authentication is depicted in Figure 3.

The WLAN authentication server is in the role of the back-end server. The terminal has a smart-card containing a secret key. The same key is available in the subscriber database of HSS (Home Subscriber Server) located in the home network of the subscriber. The terminal sends a cellular identity like the International Mobile Subscriber Identity (IMSI) as part of EAP AKA. The WLAN server uses a protocol like DIAMETER to send the IMSI to the HSS. HSS returns an AKA authentication vector, which is a quintuplet containing a chal-

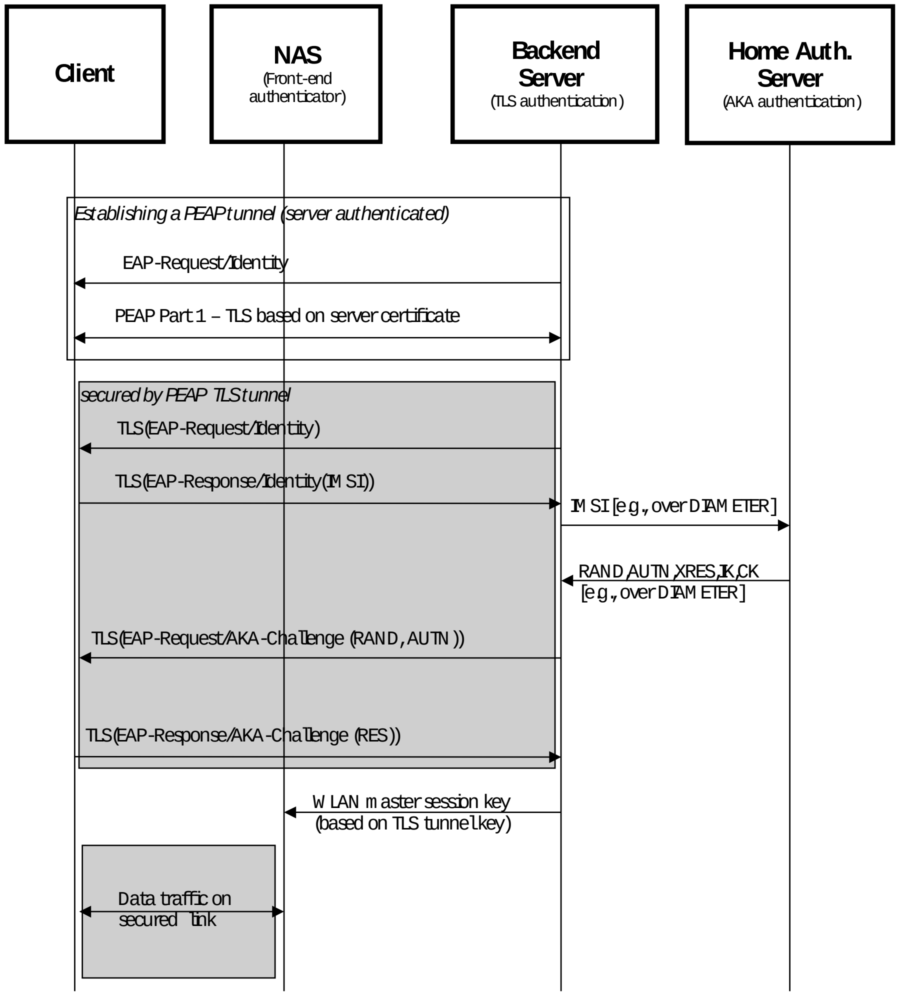

Client
NAS
(Front-end authenticator)
Backend Server
(TLS authentication)
Home Auth. Server
(AKA authentication)
Establishing a PEAP tunnel (server authenticated)
EAP-Request/Identity
PEAP Part 1 – TLS based on server certificate
secured by PEAP TLS tunnel
TLS(EAP-Request/Identity)
TLS(EAP-Response/Identity(IMSI))
IMSI [e.g., over DIAMETER]
RAND,AUTN,XRES,IK,CK
[e.g., over DIAMETER]
TLS(EAP-Request/AKA-Challenge (RAND,AUTN))
TLS(EAP-Response/AKA-Challenge (RES))
WLAN master session key
(based on TLS tunnel key)
Data traffic on
secured link

Fig. 3. PEAP with EAP AKA: Example Message Flow

lenge (RAND), an authenticator for the challenge (AUTN), expected response (XRES), and session keys IK (for integrity) and CK (for confidentiality) to the WLAN server. The WLAN server forwards RAND and AUTN to the terminal via EAP AKA. The terminal can now verify AUTN to confirm that RAND comes from HSS, and compute its own response RES and the keys IK and CK. The terminal sends RES back to the WLAN server using EAP AKA. If RES and XRES are the same, WLAN server considers the terminal to be authenticated. The AKA protocol was defined as part of the third generation (3G) cellular standardization activities. More detailed descriptions of 3G protocols are available elsewhere, e.g., [16].

Again, only the TLS master secret is used to derive the session keys to be used to protect the WLAN link. The secret key material carried within the UMTS AKA [16] authentication quintuplets is not used.

3 Man-in-the-Middle Attack

Why can the tunnelling approach go wrong? There are two reasons:

– The legacy client authentication protocol is used in other environments, e.g., plain EAP without any tunnelling, or without any EAP encapsulation at all, e.g., direct use of one-time passwords, or cellular authentication protocols.
– The client cannot or does not properly authenticate the server (even when the authentication protocol is used *within* a supposedly server-authenticated tunnel).

The active attack by a Man-in-the-Middle (MitM) proceeds as follows:

1. MitM either waits for a legitimate device to start an untunnelled legacy remote authentication protocol or actively fools the legitimate device into initiating an untunneled remote authentication protocol. It then captures the initial message sent by the legitimate client.
2. MitM initiates a tunnelled authentication protocol with a backend server.
3. After the tunnel is set up between MitM and the backup server, the MitM starts forwarding legitimate client's authentication protocol messages through the tunnel.
4. MitM unwraps the legacy authentication protocol messages received through the tunnel from the backend server and forwards them to the legitimate client.
5. After the remote authentication ended successfully, MitM derives the session keys from the same keys it is using for the tunnel.

Some legacy authentication protocols do not support mutual authentication or session key agreement. When they are used in an open environment, they are obviously vulnerable to MitM attacks. However, tunneling a generic authentication protocol framework like EAP makes the MitM attack possble even when the actual authentication method used within the framework does support mutual authentication and session protection. This situation arises because the backend server cannot be certain that the client in the legacy authentication protocol and the client endpoint of the tunnel are the same entity.

Figure 4 shows how the MitM attack works in PEAP with EAP AKA as the example client authentication protocol in a WLAN access authentication setting. The victim terminal assumes that the MitM is a UMTS radio access network. Note that the UMTS AKA protocol itself [16] and its EAP encapsulation [3] provide for mutual authentication of the user terminal and the radio access network: AUTN parameter is used to authenticate the source of the RAND challenge. The attempt to tunnel EAP AKA through PEAP effectively defeats

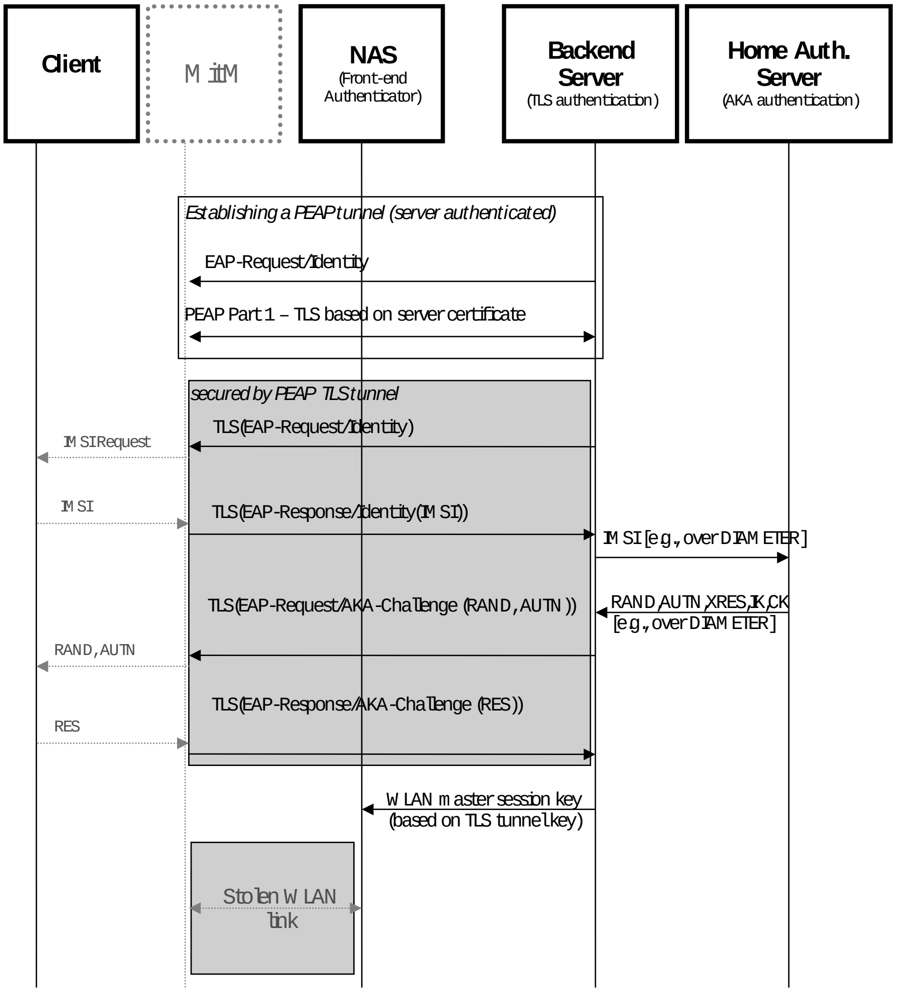

Client
MitM
NAS
(Front-end Authenticator)
Backend Server
(TLS authentication)
Home Auth. Server
(AKA authentication)
Establishing a PEAP tunnel (server authenticated)
EAP-Request/Identity
PEAP Part 1 – TLS based on server certificate
secured by PEAP TLS tunnel
TLS(EAP-Request/Identity)
IMSI Request
IMSI
TLS(EAP-Response/Identity(IMSI))
IMSI [e.g., over DIAMETER]
TLS(EAP-Request/AKA-Challenge (RAND, AUTN))
RAND, AUTN, XRES, IK, CK [e.g., over DIAMETER]
RAND, AUTN
TLS(EAP-Response/AKA-Challenge (RES))
RES
WLAN master session key (based on TLS tunnel key)
Stolen WLAN link

Fig. 4. Man-in-the-Middle in PEAP, e.g., with EAP AKA

this protection. In other words, even though the inner authentication protocol (EAP AKA) and the outer protocol (TLS) are both secure as such, composing them in the customary manner results in an insecure system.

Surprisingly, this type of protocol composition is very popular. As mentioned before, the most widespread of these may be the use of HTTP authentication [11] through a TLS tunnel (i.e., to an `https` URL). Many web sites use this type of a scheme for controlling access to their resources; for example, on-line access to bank accounts are usually controlled this way. Naturally, if the authentication method is usable without a TLS tunnel (e.g., to a plain `http` URL), a MitM attack is possible.

4 Avoiding the Attack: Cryptographic Binding

The vulnerability discussed above is due to the fact that the legitimate client and the backend server had no way to verify that their peer in the client authentication protocol is the same as the entity at the other end of the outer tunnel which possesses the session keys at the end of the protocol. We can enable this verification either implicitly or explicitly. The goal of *implicit authentication* is to ensure that the only the legitimate parties are capable of gaining access to the correct session. In *explicit authentication* a separate authentication step is performed to verify that the entities possessing the master secrets from which the session keys are derived, are indeed the legitimate parties. In both cases, the authentication is provided by a cryptographic binding between the inner and the outer protocols.

These solutions are applicable to mutual authentication protocols that are constructed as a combination of two authentication protocols. The outer protocol is assumed to support the construction of a secure tunnel based on server authentication. As we saw, this is the case with all the protocols cited in section 2.

A pre-requisite to cryptographic binding is that the inner authentication protocol must produce a key S in one of two ways:

1. S is a session key resulting from a run of the inner protocol; i.e., it is an authentication *and* key agreement protocol; or
2. S is the client's long-term authentication key and is accessible for derivation of session keys directly, outside the authentication protocol.

A typical example of a protocol of the first type is EAP AKA. Figure 3 illustrates how EAP AKA can be used within the PEAP protocol. The AKA quintuplets contain 256 bits of secret session keys which would be readily available for session key derivation by the backend server. These AKA session keys are also available in the client's device. There are several examples of existing client authentication protocols that support session key derivation. For such protocols, no changes are required to provide the necessary cryptographic binding.

In the second case, it is necessary to specify an additional key derivation application that has direct access to client's authentication key. This would be in addition to the existing authentication protocol and is to be judged separately for each case.

Let K be the ultimate session key. In PEAP and EAP-TTLS the session key K is the master key that finally becomes available to the local backend server (access server), which uses K to derive further session keys. Let T denote the master key that is used to derive the secret keys for the protection tunnel. For example, the TLS master key derived in the TLS handshake of PEAP is a typical example of T.

We achieve the binding between the inner and outer protocols by binding S and K. On the network side some entity, let us call it "binding agent", is responsible for collecting the secret key information S and T and creating the binding value. Typically the binding agent is either the local backend server or the home authentication server. If S is a long term authentication key of the client, then the binding agent is preferably co-located with client's home authentication server, to avoid transfer of S across the network.

As mentioned, there are two ways of using S to achieve the necessary binding of S to K. In implicit binding, the binding is established directly by taking S in addition to T as input to the session key computation. The binding agent and the client each compute its copy of the session key K from S and T using a pseudo random function suitable for key derivation. The binding agent distributes the session key to the network entities that need to use it for further communication with the client. This provides implicit authentication of the client.

In explicit binding we make use of a cryptographic check value to verify that the client who is in possession of T is also in possession of S. The binding agent and the client each compute its copy of a verification value V from S and T using a cryptographic hash function or a message authentication function. They then transfer their verification values to some network entity responsible for comparing the two verification values. If they are equal, the client is granted access to the network service. The comparing entity can be the backend server or the home authentication server. This provides explicit authentication of the client. If explicit binding is used then the session key K can be based on S or T or both.

For both the implicit and explicit binding mechanisms we assume that the process to construct the tunnel secret T is contributory. However, it is not necessary that the server is explicitly authenticated by the client in the tunnel establishment. It is also possible for the client to rely on the home server for the server authentication. Then the binding of S to K provides server authentication also in the cases where the client neglects the result of the server authentication during tunnel setup, or an anonymous tunnel is used.

Explicit binding is always necessary in case the session key K is based solely on the tunnel secret T. A variation of explicit binding works as follows. In the computation of the verification value V, some tunnel-indication data D is used instead of T. For example, D can be an entire protocol message sent by the client, or the addresses of the parties or just a text string unique to the application. This variation is useful in cases where access to T is not available because of implementation constraints (e.g., a black-box implementation of TLS will not allow the TLS master key to be accessible outside TLS usage). In typical scenarios, D does not specify T in a unique manner. Then this approach is weaker than the normal explicit binding, described earlier, because it relies on the client carrying out server authentication properly during tunnel setup.

It is also possible to implement both implicit and explicit binding. In such a case the explicit binding verification acts as a key confirmation for the agreed session key.

5 Weak Authentication Protocols

Consider a client authentication protocol based on weak secrets like passwords. Suppose further that the protocol does not use any modern techniques to protect the weak secret against guessing attacks. Many of the tunnelling protocols described earlier, e.g., the PIC protocol, are particularly intended to address such weak authentication protocols, which are insecure without added protective tunnelling. One may ask what is the impact of the cryptographic binding described in section 4 on such a case. Does the binding possibly improve the overall security? It is clear that if the client is unable to perform server authentication for the outer tunnel protocol, then the binding does not improve the security: an attacker could masquerade as the server in protocol run, and use the information gained in a dictionary attack on the password.

However, all the examples we considered in section 2 do assume that the client can perform server authentication for the tunnel protocol. In this case, the attacker cannot pretend to be the server end of the tunnel towards the client. If the attacker is a passive eavesdropper, he cannot perform the dictionary attack on the binding because the binding includes the outer tunnel key which is assumed to have sufficient entropy. If the attacker pretends to be the client towards the server and if the server does the binding first (e.g., if the server sends the binding verification value to the client, or if it starts using the key K or S to encrypt or authenticate messages), then he can perform a dictionary attack. But this is easily prevented by requiring that the client does the binding first and demonstrate knowledge of S and T to the server. If this step fails, the server must terminate the session.

Thus, the cryptographic binding does not *reduce* the security of a weak authentication protocol. However, to be fully secure, weak authentication protocols used with server authenticated tunnels must satisfy two conditions:

A1 *Correct server authentication*: the client **must** perform server authentication correctly, and

A2 *No mixing of authentication modes*: if a client uses tunnelled authentication, it **must not** use the same authentication protocol outside secure tunnels.

Assumption A2 is unavoidable for weak authentication protocols. But it clearly diminishes the capability of the tunnelling protocol to leverage the advantages of already deployed legacy authentication protocols. Well-designed authentication protocols do not need assumption A2.

If assumption A2 can be made in general, cryptographic binding is not necessary, but the legacy advantages are lost. On the other hand, cryptographic binding is not applicable to the legacy authentication protocols that do not yield a key suitable for cryptographic binding.

Generic tunnelling protocols, by definition, should be able to work with all types of authentication protocols while making as few assumptions about them as possible. Therefore a generic tunnelling protocol must not take A2 as a blanket assumption to be imposed on all authentication protocols. Instead it should allow the possibility of using the type of cryptographic binding described in section 4 where appropriate. This way, the tunnelling protocol can be secure, generic (supporting both weak and strong authentication protocols), and non-invasive (avoiding unnecessary restrictions on strong authentication protocols).

6 Impact and Outlook

The attack is applicable to a number of proposed protocols, including PIC, IKEv2, XAUTH, PEAP, EAP TTLS and POTLS [1]. These are not academic toy protocols, but protocols with a good prospect of being deployed and used widely. For IKEv2 it concerns, in particular, the Secure Legacy Authentication (SLA) protocol, which is intended to be included in the main body of IKEv2. The conclusion from the discussion at the IPSEC list [24] was that an explicit cryptographic binding for SLA was to be specified. The binding is mandated in case the legacy authentication method provides a shared secret. This change has already been included in a more recent version of the IKEv2 draft [10].

The sheer number of different protocol combinations where the same MitM problem has manifested is surprising. One reason is the lack of easily accessible tools for specifying and verifying security properties of protocols. The security of protocol composability is still an emerging area of research. In [18], Meadows points out the current status of research as well as other practical examples of security failures arising from protocol composition. Also, even though the security protocols research community has been working on protocol verification for decades, the current tools and methodologies require significant expertise, and are not yet ready for use by people who are not actively doing protocol verification research. It is only recently that experts from the security protocols community have started analyzing draft specifications from open standardization bodies [18]. As Meadows points out, the possibility to analyze draft protocols before they become standards is an important opportunity for researchers in this area.

A second possible reason is that, in all of the cases examined, the legacy authentication protocol was a framework, rather than a concrete protocol. This means that the properties of individual authentication protocols are not readily noticeable to the designers of the tunnelling approach. A particular problem is created by the large variety of legacy protocols which differ in the level of security provided: e.g., some of them provide strong mutual authentication, and others are based on a user selected password. It is not clear if it is useful or even possible to accommodate all different protocols within the same tunnelling framework.

7 Conclusion

In this paper we have shown that when a client authentication protocol is tunnelled within another protocol, it is necessary for each endpoint to demonstrate that it has participated in both protocols within the authentication exchange. If this is not demonstrated then the tunnelled authentication protocol is vulnerable to a Man-in-the-Middle attack.

We have also shown that the required demonstration can be provided in an implicit or explicit way in a form of a cryptographic binding between the tunnel protocol and the authentication protocol. In our proposals the binding facility is implemented in the outer tunnel protocol. It requires the authentication protocol to provide some secret key values for the use of the binding. This approach is preferred since it requires minimal or no changes to the authentication protocols. It allows for flexible and secure usage of an authentication protocol in multiple authentication environments without the authentication protocol being aware of the specific environment.

The cryptographic binding proposed in this paper does not diminish the security of the tunnelled protocols in any case. If the inner authentication protocol is a weak authentication protocol based on a weak client secret, the tunnel must be constructed based on server authentication, and the client should not use the same secret in different environments. Otherwise, the protocol is vulnerable to dictionary attacks, with or without cryptographic binding. Strong authentication methods are not vulnerable to dictionary attacks, and hence should not be restricted to tunnelled environments only. Hence the need for the type of binding described in this paper.

8 Current Developments

We pointed out the MitM problem to the authors of some of the tunnelling proposals in early October, 2002. Since then a number of positive developments have taken place. The IETF EAP working group has recognized the problem of securely binding a sequence of authentication methods together. It is now marked as an open issue to be solved [6]. Another draft [21] describes the type of solutions prescribed in this paper. Designers of some of the affected protocols have published new versions of their specifications: for example, the new version of the PEAP draft specification [2] adopts the techniques specified in [21] to protect against MitM attacks. Other affected specifications, such as EAP SIM GMM authentication [7] and PIC [23] have been withdrawn altogether.

We therefore expect that in due course of time the problem will be solved in all the identified specifications. However, it requires significant analysis and development for each protocol. In each case one must decide at which protocol message the binding is implemented, which parties shall create the binding, and how to derive the shared secrets used for the binding. Different decisions result in different protocol properties.

Client authentication protocols may take additional safeguards to protect against MitM. For example, in a newer version of the EAP AKA specification [4], the authentication response must include RES as well as a message authentication code on RES. Thus the attack described in Figure 4 is no longer possible. However, the MitM can still pretend to be a backend server towards the victim. Therefore, if mixed mode usage is unavoidable, then ultimately the tunnelling protocols have to be fixed.

As we mentioned already, the problem is not limited to EAP only. For example, HTTP Digest AKA specification [19] effectively uses the HTTP authentication protocol [11] as a framework. The standard combination of HTTP authentication with TLS is thus rendered insecure when the underlying HTTP authentication is HTTP Digest AKA, despite the fact that HTTP Digest AKA is a secure protocol providing mutual authentication and strong session key agreement.

Acknowledgments

We thank Henry Haverinen for many interesting and useful discussions on this and other related topics. We also thank Jan-Erik Ekberg, Dan Forsberg, Pasi Eronen, Philip Ginzboorg, Tao Haukka, David Jablon, Hugo Krawczyk, Jose Puthenkulam, Jarno Rajahalme, Heikki Riittinen and the participants of the IST-SHAMAN project for discussions and valuable feedback on our ideas.

References

1. Bernard Aboba. Review of man-in-the-middle problem statement draft. Message to IETF saag mailing list, January 2003. `http://jis.mit.edu/pipermail/saag/2003q1/000684.html`.
2. H. Andersson, S. Josefsson, Glen Zorn, Dan Simon, and Ashwin Palekar. Protected EAP Protocol (PEAP), September 2002. IETF personal draft `draft-josefsson-pppext-eap-tls-eap-05.txt`.
3. J. Arkko and H. Haverinen. EAP AKA Authentication, June 2002. IETF personal draft `draft-arkko-pppext-eap-aka-04.txt`.
4. J. Arkko and H. Haverinen. EAP AKA Authentication, January 2003. IETF personal draft `draft-arkko-pppext-eap-aka-08.txt`.
5. N. Asokan, Valtteri Niemi, and Kaisa Nyberg. Man-in-the-middle in tunneled authentication protocols. Technical Report 2002/163, IACR ePrint archive, October 2002. `http://eprint.iacr.org/2002/163/`.
6. L. Blunk, J. Vollbrecht, and Bernard Aboba. Extensible Authentication Protocol (EAP), October 2002. IETF *pppext* working group draft `draft-ietf-pppext-rfc2284bis-07.txt`.
7. Adrian Buckley, Prasanna Satarasinghe, Vladmir Alperovich, Jose Puthenkulam, Jesse Walker, and Victor Lortz. EAP SIM GMM Authentication, August 2002. IETF personal draft `draft-buckley-pppext-eap-sim-gmm-00.txt`.
8. Pat Calhoun et al. Diameter Base Protocol, December 2002. IETF *aaa* working group draft `draft-ietf-aaa-diameter-17.txt`.
9. T. Dierks and C. Allen. The TLS Protocol Version 1.0, January 1999. IETF RFC 2246.
10. Charlie Kaufman (Editor). Internet Key Exchange (IKEv2) Protocol, February 2003. IETF *ipsec* working group draft `draft-ietf-ipsec-ikev2-05.txt`.
11. J. Franks et al. HTTP Authentication: Basic and Digest Access Authentication, June 1999. IETF RFC 2617.
12. Paul Funk and Simon Blake-Wilson. EAP Tunneled TLS Authentication Protocol (EAP-TTLS), February 2002. IETF *pppext* working group draft `draft-ietf-pppext-eap-ttls-01.txt` (expired).
13. Dan Harkins, Derrel Piper, and Paul Hoffman. Secure Legacy Authentication (SLA) for IKEv2, December 2002. IETF personal draft draft-hoffman-sla-00.txt.
14. H. Haverinen and J. Salowey. EAP SIM Authentication, October 2002. IETF personal draft `draft-haverinen-pppext-eap-sim-06.txt`.
15. S. Josefsson, A. Palekar, D. Simon, and G. Zorn. Protected EAP Protocol (PEAP), March 2003. IETF personal draft `draft-josefsson-pppext-eap-tls-eap-06.txt`.
16. Heikki Kaaranen, Siamäk Naghian, Lauri Laitinen, Ari Ahtiainen, and Valtteri Niemi. *UMTS Networks: Architecture, Mobility and Services.* John Wiley &amp; Sons, 2001.

17. J. Vollbrecht L. Blunk. PPP Extensible Authentication Protocol (EAP), March 1998. IETF RFC 2284.
18. Catherine Meadows. Formal methods for cryptographic protocol analysis: emerging issues and trends. *IEEE Journal on Selected Areas in Communications*, 21(1):44–54, January 2003.
19. Aki Niemi, Jari Arkko, and Vesa Torvinen. Hypertext transfer protocol (http) digest authentication using authentication and key agreement (aka). IETF RFC 3310, September 2002.
20. Yoshihiro Ohba, Shinichi Baba, and Subir Das. PANA over TLS (POTLS), September 2002. IETF personal draft `draft-ohba-pana-potls-00.txt`.
21. Jose Puthenkulam, Victor Lortz, Ashwin Palekar, Dan Simon, and Bernard Aboba. The compound authentication binding problem, March 2003. IETF personal draft `draft-puthenkulam-eap-binding-02.txt`.
22. C. Righney et al. Remote Authentication Dial In User Service (RADIUS), June 2000. IETF RFC 2865.
23. Y. Sheffer, H. Krawczyk, and Bernard Aboba. PIC, A Pre-IKE Credential Provisioning Protocol, October 2002. IETF *ipsra* working group draft `draft-ietf-ipsra-pic-06.txt`.
24. IETF IPsec working group. Secure legacy authentication for IKEv2. Discussion thread on the IPSec mailing list.
`http://www.vpnc.org/ietf-ipsec/mail-archive/threads.html#02763`.

Man-in-the-Middle in Tunnelled Authentication Protocols
(Transcript of Discussion)

N. Asokan

Nokia Research Center, Finland

John Ioannidis: I have to interrupt here and be even more offensive than usual. But you are using the worst rackets in industry as a justification for what you're doing. There are all sorts of people just generating garbage protocols, a couple of which you have already mentioned here. We're trying to reverse their work, whereas you're trying to advocate we use all these garbage protocols.

Reply: I'm not saying that. I'm saying that something is wrong here. You are trying to do the right thing but you are going about it the wrong way. The reality is that people are going to use existing credentials because they obtained them at great expense, and they want to reuse them. I'm not justifying it.

Bruce Christianson: I think he's going to come up with a very good new reason why this is a bad thing to do, in which case it's more ammunition for you JI, or he's going to show that the reasons for which we usually think it's bad are wrong, in which case we're going to have to change our position anyway. Either way you should let him go on for a bit.

Reply: The most common use of this kind of authentication through the tunnel is essentially to guide the application inside. I guess actually the authentication was not intended as a general framework but it's being used as one. So the PAP was supposed to be used running EAP, AKA inside that, while sending a random challenge. Since this is an authenticator tunnel, anybody could make that, including the man in the middle. The man in the middle is sent a random challenge and authenticated, he could turn around, pretend to be a server network and get the client to send a response. Notice that the client thinks that it's his own network server, and instead he does mutual authentication. And at this point he goes back and the client has been authenticated to send these keys to the NAS and that would leave the man in the middle with a stolen key.

Ross Anderson: But surely this attack would not work if the certificates that people use from TLS actually worked?

Reply: The man in the middle is not pretending to be a TLS server, he's pretending to be a server network. So the server network has it's own usual authentication but this is effectively defeating that.

Jari Arkko: I think the important point here is that you actually have two strong methods. TLS is strong, AKA is strong, but when you put them together and don't bind them, then you get the problem. That's the real issue here. I think the main question we should ask here is whether it makes sense to put the tunnel on top of that, because if you have a strong method like AKA, or EAP or whatever, why would you put a tunnel outside that? Basically you get session secrets from that, but you already get them from AKA and other route methods. I guess maybe there's some case when you have something really bad, like a fake password, or someone plays someone that is not really strong, and then you get some advantage from this tunnel. But you get some disadvantage through the tunnel as well, so it's not really clear to me why you actually want to do this.

Reply: That's actually the point I was going to make, because this framework doesn't reveal the methods underneath. So people who make these tunnels make the system work on EAP. When these are used in combination then the result is less secure than using either of them. But the conditions for this failure to happen involved having the same credentials used inside and outside the tunnel. In the previous case, this AKA authentication, which is basically a smartcard, means you should be able to use the same smartcard for server access and also for PAP access, but the point is that that's what they were trying to do in the first place.

So the best thing is to say that you can't use the same credential in both places, but that would mean that it's not legacy authentication anymore. For that you have to have a separate SIM card for server phone access and another SIM card for authentication. The tunnelling method requires mutual authentication, but then there doesn't seem to be any need to have a second authentication at the tunnel. The third possibility is that you could bind the outer protocol and the inner protocol, and this can be done explicitly by, for example, deriving a new session key based on these two keys, or even simply doing a key transformation.

But again this would require changes, it would require changes to the framework, or to the tunnelling protocol, which may not be appropriate because all of this came from the motivation of using legacy authentication.

Pekka Nikander: If you are creating session keys based on the legacy somehow, then you are actually creating the session keys in a different way from the way you do in the legacy system. And since you are using the session keys in a different way, there isn't that engineering problem with combining and getting the session keys from the other systems. You are using them in a different way anyway. So is that appropriate?

Jari Arkko: I don't understand why you think it is. If you require a mutual application tunnel protocol then, if that is possible, you don't need a tunnel in the first place. It seems to me that many of these situations could be avoided by just avoiding the tunnel.

Reply: If you use TLS with client authentication, and your goal is authentication of the client, then you wouldn't need a separate client authentication protocol, you can just use TLS. There is one caveat though, one of the reasons why people want to use these tunnels is identity privacy. With EAP the identity is revealed to eavesdroppers. If they want to hide this then that's when they want to put it in the tunnel. The argument might still hold that you want to use some kind of server authenticator through this tunnel, and then run a separate authentication method through that just because you want to hide identity.

Jari Arkko: And if you did that you wouldn't even think of using the session keys for a unauthenticated tunnel.

Reply: That, I agree, might give you a false sense of security.

Virgil Gligor: I think, as JI pointed out, you have to stay aware of your previous mistakes.

Jari Arkko: I mean, you have *some* legacy stuff that actually works, and then you have the EAP framework which also sort of works. But now if you try to use some new things like these tunnels, where we have no legacy, we are no longer completely free to choose what to do. I'm questioning whether we should do these tunnels at all.

Matt Blaze: I think I'd like to go further. The usual argument for legacy anything is that there are some components in the system that you can't change, especially if you believe they have to stay the same for a long period of time because changing them would be very, very expensive. But I can't quite figure out what that component is here. Is there some thing that we seem to want to avoid changing, or is the motivation to avoid changing things the desire to just use existing protocols even though the components that are implementing them are being updated constantly.

Reply: No, in this case I think the important thing is the credentials that have been provided to the users, the smartcard the users have. Changing them is going to be expensive.

Matt Blaze: So it's the SIM card. So how expensive is it to give people new SIMs when they're buying new phones every two years?

Reply: Expensive enough that operators don't want to do it anymore.

Ross Anderson: But these are the third generation phones where only trivial numbers have been sold so far. This is a terrible legacy hanging round your neck.

John Ioannidis: They were incorrectly designed. Those of us who were involved kept yelling about the incorrect design. Now we have this incorrect design, and these broken protocols that are trying to provide legacy in non-existent space.

Virgil Gligor: That's the challenge.

John Ioannidis: It is a challenge.

Matt Blaze: The thing we're not allowed to change is the SIM, and that's . . . OK.

Reply: I would say that's the most difficult thing to change, but of course the SIM is not the key. The SIM also has the key in it, so imparting those to some part of the protocol is a difficulty.

John Ioannidis: But don't you have to do something with the EAP? You can design a completely new protocol if you need to deploy something, you don't have to use EAP. I don't know of any device that can change EAP, do you?

Reply: If you do wireless LAN access then you do. But changing the handset is easier.

Matt Blaze: So we're already changing the handset?

Reply: Yes.

Matt Blaze: OK, I guess I misunderstood the design of the model of the SIM, because essentially a SIM is running software.

Reply: Yes, well a SIM is actually software.

A: Right. A SIM is essentially a Smartcard, but it can only be updated remotely. The only way to change anything about the SIM is to mail the user a new one.

Reply: Yes, exactly. With the keys.

Ross Anderson: Doesn't 3G support end-to-end session keys?

Reply: Yes.

Ross Anderson: So what you could do is to sell a new service to your customers, provide the customers a run-out to do wireless LAN. He gets better use, and he can set up an end-to-end session key with somebody who's trying to log on. That way you can forget about all this TLS stuff, and you charge the guy ten cents for authentication.

John Ioannidis: Hear hear!!

Reply: Except you wouldn't be conforming to the current standards.

Pekka Nikander: In a way there are two pieces of legacy in here; one is the SIM card and the other is EAP which is employed in it.

Reply: You can say that in thirty, forty years they're going to use a different type of wireless LAN, you know. All operators will have to have two different kinds of wireless LAN, not just one; one to support telephone users, and one to support non-telephone users.

John Ioannidis: Even if all the firmware were upgradeable, operators cannot just release a new code overnight. You want to take advantage of the protocol, because right now what exists there does not use a protocol tunnel.

Reply: The step before that is that you have to get this accepted in the right standard.

John Ioannidis: No you don't.

Pekka Nikander: The sad thing here is that people are using an enormous amount of money devolving these kinds of tunnel protocols and I don't know exactly what to do about it. People like me have known for a number of years that this is insane, but it still continues to happen. Part of the problem is that the people who design these protocols don't have access to the knowledge and tools that people in this room use.

Matt Blaze: In all fairness, that doesn't sound like a very persuasive argument, I'm afraid. Essentially the argument here is, the people who design the protocols and the standards group, aren't as smart as we are. The solution to that problem is that they should listen to us, or behave in a smarter way, not that we should adapt protocols to be able to be designed by people who aren't very smart. The obvious solution to that problem is to just use an end-to-end protocol for the things that need to be end-to-end, and don't compose the protocol.

Reply: That was what I said here. Basically everybody agrees that this is the case.

Jari Arkko: There's a very big difference between re-using the existing credentials, and combining protocols. I think that combining the things is what is wrong here, but re-using strong protocols, even if they exist, is fine in my opinion.

Pekka Nikander: In a way the problem is not really that people aren't very smart, they might be. What I'm saying is that many of them say that something can't be done because of some reason which is more or less insane; there are artificial constraints on the design space here, constraints which actually don't exist. And I don't know how to do that.

John Ioannidis: But even if we don't know how to do that, and I don't know how to do that either, we should not be making the situation worse by designing more bad protocols.

Ross Anderson: But these are all essentially well known roads of failure, things become more complex, like the inevitable evolution of the Swiss Army Knife.[1] These protocols become so nasty, with 400 page specs, that nobody can understand except possibly a beginning graduate student who is really, really keen to find the hack that will write his name in lights. It's exactly the same scene we have in the space of API attacks; they are the same kind of animal. Put sixteen different protocols inside a black box, and hope that they'll live together comfortably and occasionally share keys. What happens is boom, the thing was broken for twenty years and nobody notices. This is exactly the same thing, it's the pre-manufacture of API failures on an industrial scale. Mind you, some people might say there's some social value in it. One of the main reasons that we know what's actually going on in Iraq is that we know how to hack commu-

[1] Roger Needham, "Security Protocols and the Swiss Army Knife", LNCS 2133, 1–4.

nications and there's reasonable amounts of news coming out. The things we're talking about here may assure that for the next twenty or thirty years digital communication will be open to surveillance by third parties.

Bruce Christianson: It's true that a lot of these security failures in protocol design come from a failure to keep the hidden agenda hidden.

Birgit Pfitzmann: Quite recently in industry, trying to talk with people about standards, I find they won't listen even for a second. You have to stay very, very close to what they are doing, point out individual failures, and you might get the stuff tweaked. But you don't get them to listen at all unless you get very close to their own language and try to find out whether maybe some of the hidden agenda makes sense. Sometimes you do find things they really want to do, perhaps because of a power struggle, or because of the patent. You have to understand these things otherwise you'll never influence them, that's my experience.

Bruce Christianson: But it's a selling problem. Consider rice farmers who've been growing rice in the same field in exactly the same way for three thousand years, and you try and sell them a new way of growing rice.

Jari Arkko: But I don't think we have that problem right now, because no-one is actually using that combination at the moment. I think we should stay away from that. I think it's fine to use the strong methods in the legacy, but not do this combination. That is the thing that introduces the attack here.

Michael Roe: Although in principle you could write a new protocol, people tend to stay very much away from that. They say, "Well, we've got these three other things that already exist, why don't we just work them together?"

John Ioannidis: Why use a hammer on a screw, when you have a wrench?

Michael Roe: It's quite a persuasive argument. The trouble is that occasionally you get problems when you plug these things together. They don't quite work in the way you expect them to work. It's not just a matter of the cost of writing code, there's also the cost of testing the code, which is often considerably greater, and worse yet interoperability. After spending five years going to IETF meetings, finally all the manufacturers have agreed what the RFC means, so if you can pick up some of someone else's stuff you can use his little box. Yes, you could write a new protocol instead and go through this process again but ...

Bruce Christianson: But you know by the time you've done it that these people will have gone to a security workshop, and will come along telling you that it's all old hat, and you need to do it again.

John Ioannidis: Well maybe, but not very successfully.

Michael Roe: The strength of this work we're talking about is very real. It's useful to be able to know which combinations, or existing building blocks, have been plugged together nicely in the past.

Matt Blaze: But I think part of that comes from a lack of recognition that composing two protocols is the same as designing a new protocol. We all know that you have to look at how these protocols interact, but a lot of the engineering work on this does not recognise that, and believes naïvely that this is much simpler than it is.

Reply: People think that by composing two protocols you can avoid any security analysis because they were all analysed separately. It's a reality that if we don't do anything then this will get deployed. What's the point of doing this research if you can't stop these from being deployed?

John Ioannidis: But you're not suggesting stopping something being deployed, you're suggesting a way of fixing something that's broken. But you're fixing something that should not have been broken in the first place. Designing it properly would probably be easier than trying to fix it.

Reply: If we were starting from a clean slate that would be so.

John Ioannidis: Then let's see it as a clean slate.

Alec Yasinsac: That's the question: is designing it properly easier than trying to fix it? If you take two protocols, prove it will work, and compose them, then that may be easier than starting from the beginning. We don't know.

Bruce Christianson: From an engineering point of view, it's not obvious. As Matt says, it doesn't let you off having to do a security analysis. From a design point of view it's exactly the same as designing a new protocol, but from the engineering and product management point of view, and for the whole issue of configuration and testing, the composite protocol is much, much easier. These are advantages that people at the sharp end value very highly.

Jonathan Tuliani: It doesn't matter which one's easier to analyse, what matters is what's cheapest, and that includes factors like time to market.

Alec Yasinsac: But that's not contradicting what I'm saying, composing bad protocols will always give you bad protocols.

Reply: I don't know of any obvious reasons to argue with that. But they were composing two seemingly good protocols.

Jari Arkko: Even for the strongest protocols that you can imagine, if you combine them together and they don't find the keys from two layers to get that, you will get a problem.

Matt Blaze: I guess the problem then is communicating that. When we say reuse existing protocols, we don't mean compose protocols.

John Ioannidis: Even composition of protocols has not had a very specific meaning. What does it mean to compose two protocols?

Bruce Christianson: They are very often layered. What else is a tunnel anyway?

Towards a Framework for Autonomic Security Protocols

Simon N. Foley and Hongbin Zhou

Boole Centre for Research in Informatics,
Department of Computer Science,
University College, Cork, Ireland
{s.foley, zhou}@cs.ucc.ie

Abstract. This paper proposes a belief logic based approach that allows principals to negotiate and on-the-fly generate security protocols. When principals wish to interact then, rather than offering each other a fixed menu of 'known' protocols, they negotiate and generate a new protocol that is tailored specifically to their current security environment and requirements. This approach provides a basis for autonomic security protocols. Such protocols are self-configuring since only principal assumptions and protocol goals need to be a-priori configured. The approach has the potential to survive security compromises that can be modelled as changes in the beliefs of the principals. A compromise of a key or a change in the trust relationships between principals can result in a principal self-healing and synthesising a new protocol to survive the event.

1 Introduction

Networked services and applications are typically commissioned with a fixed repertoire of security protocols that provide for necessary authentication, key-exchange, non-repudiation, delegation, and so forth. Applications and services are expected to negotiate with each other and agree on appropriate security protocols that both can (and are willing to) use. A simple example is a web-service that requires clients to establish SSL based connections. When negotiating a connection, the client and server agree on the version of the protocol to use. While protocols may be designed to support a range of different underlying authentication protocols, and so forth, it is not feasible to expect principals to be, a priori, conversant in all possible protocols. Protocol agnostic approaches such as Jini [8] allow resource providers to register the protocol, that its clients should use, with a Jini Server. While more flexible, the provider's protocol is fixed and is not generally suitable for security protocols.

We are investigating the use of protocol synthesis techniques [5,7,11,13] to allow principals negotiate and on-the-fly generate security protocols. When principals wish to interact then, rather than offering each other a fixed menu of 'known' protocols, the protocol negotiation process generates a new protocol that is tailored specifically to their current security environment and requirements. A principal's security environment reflects the keys that it knows, the trust relationships with other principals and any other assumptions it holds.

We conjecture that applications that use such *Self-configuring* protocols would not require configuration to provide a fixed number of 'known' protocols. Instead, the application designer specifies the necessary security requirements for valid interaction. For example, a server may be willing to accept any connection from an authenticated principal; the security requirements and current environment of the server (and client) are used to synthesis a protocol that meets these goals.

Protocols could be generated on the basis of the security environment of the principals. A change in the security environment of a principal may result in the re-negotiation of a new security protocol. This provides a basis for *survivable* security protocols that have the potential to, in effect, *self-heal* and adapt to recover from changes in the security environment.

These characteristics—self-configuring, self-healing and survivability—are are properties that form part of the autonomic computing manifesto [2]. In the next section we outline our current research on this area.

2 Autonomic Security Protocols

The Basic Protocol Synthesis Protocol (BPSP) is a bootstrapping protocol that is used by principals to negotiate and generate a new protocol specification. It is used when an Initiator requests connection to a Responder. Principals specify their protocol goals and assumptions using the Simple Logic [4].

The Simple Logic is a BAN-style belief logic that uses abstract channels similar to the Spi Calculus [1] to represent keyed communication between principals. This results in a very simple, yet expressive, logical system. A synthesis technique is also proposed [4] that can be used to guide the (manual) systematic calculation of a protocol from its goals.

As part of our current research we are exploring automated protocol generation techniques such as [5,11,7] can be used within our framework. We have developed an automatic verification tool [10] for the Simple Logic and implemented

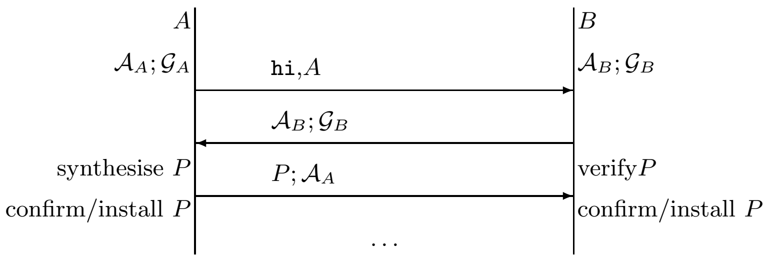


Fig. 1. Basic Protocol Synthesis Protocol (BPSP)

using Theory Generation [9]. This tool has been extended [13] to support the automated synthesis/generation of protocols. The approach in [13] combines and automates the manual synthesis rules from the Simple Logic with Guttman's [7] manual design process. We adapt the synthesis rules of the Simple Logic to guide an automatic backwards search for a sub-protocol from a single goal. Given a number of individual goals, an automated technique is described to combine synthesised sub-protocols into final candidate protocols.

Figure 1 depicts the basic Protocol Synthesis Protocol. A protocol initiator A requests connection to B. B responds by passing details of its protocol goal $\mathcal{G}_B$ and the assumptions $\mathcal{A}_B$ that it currently holds. Principal A uses its own assumptions and those presented by B to synthesise a new protocol P that meets their respective goals. This protocol is returned to B, which attempts to validate using the validation tool. If validation is successful then A and B install and engage in the protocol P.

Example 1. Consider a service B that expects connections to be authenticated. The protocol requirement for the server is that a connecting principal Q should be authenticated. This is expressed in the logic as the goal

$$\mathcal{G}_B(Q) \triangleq B \models Q \mid\!\mid\!\sim (B, Nb) \tag{1}$$

where N_B is a nonce. For reasons of space we do not describe the logic in detail, however, the connectives '$\models$' and '$\mid\!\mid\!\sim$' have the usual interpretation [6] of 'believes' and 'recently said' in this run of the protocol.

The assumptions of B reflect its belief that it knows A's public key K_A:

$$\begin{aligned} \mathcal{A}_B \triangleq\ & B \models w(C_A) = \{A\}; B \in r(C_A); \\ & B \models \# N_B; \end{aligned}$$

where C_A is the channel established by the public/private key pair K_A, K_A^{-1}, and may be written only by A and read by B (and possibly others).

For simplicity, we assume that A has no goals, and for the moment its assumptions are not relevant. On receipt of B's protocol goal $\mathcal{G}_B(A)$ and assumptions $\mathcal{A}_B$, the connecting client A can immediately synthesise an (idealised) protocol P that corresponds to

$$\begin{aligned} P \triangleq\ & B \rightarrow A : N_b; \\ & A \rightarrow B : \{B, N_b\}_{K_A^{-1}}; \end{aligned}$$

which, in turn can be verified by B. Assuming that it is possible for the principals to automatically engineer a protocol implementation from P, then the principals install their protocol components and execute it. △

In the above example it is possible for A and B to negotiate a new protocol without the participation of other principals. The protocol synthesis protocol will have to handle situations where the participants need the assistance of third parties, both in terms of deriving protocols and in implementing protocols.

Example 2. Suppose that C shares a secret key K_{AC} (securing channel C_{AC} and requests a connection with A. Suppose further that B trusts A in the sense that $\mathcal{A}_B$ includes

$$B \models (A \mid\!\equiv\!\sim \phi_1) \Rightarrow (A \models \phi_1) \tag{2}$$
$$B \models (A \mid\!\equiv\!\sim (C \mid\!\sim \phi_2)) \Rightarrow (C \mid\!\sim \phi_2) \tag{3}$$

for arbitrary ϕ_1, ϕ_2. These formulae reflect B's belief that A is honest and that A is competent in deciding whether C at some time in the past said ('$\mid\!\sim$') some message. Given these beliefs, then C (given $\mathcal{A}_B$) can synthesise the protocol

$$\begin{aligned} P_2 \triangleq B &\rightarrow C : N_b \\ C &\rightarrow A : \{B, N_b\}_{K_{AB}} \\ A &\rightarrow B : \{C, B, N_b\}_{K_A^{-1}} \end{aligned}$$

The basic protocol synthesis protocol (Figure 1) must be extended to include participation of A in the setting up and execution of the final protocol. $\triangle$

Note that the Basic Protocol Synthesis protocol as described is potentially vulnerable to attack. In this case an attacker tricks one of (or both) the principals into using a different protocol. Having generated a protocol P, techniques such as [3] may prove useful in making the protocol robust against such protocol-identity attacks. This requirement may also be expressed as additional goals that should be verified by the principles B and A, respectively,

$$B \models A \mid\!\equiv\!\sim P$$
$$A \models B \models A \mid\!\equiv\!\sim P$$

We assumes that the assumptions and goals of the principals are included as part of the specification of P (this ensures that $B \models A \mid\!\equiv\!\sim \mathcal{A}_A$, and so forth).

3 Discussion

We have sketched an architecture that is proposed to support on the fly synthesis of protocols. This approach provides a basis for autonomic security protocols. Such protocols are self-configuring since only principal assumptions and protocol goals need to be a-priori configured. The architecture has the potential to survive security compromises that can be modelled as changes in the beliefs of the principals. A compromise of a key or a change in the trust relationships between principals can result in a principal self-healing and synthesising a new protocol to survive the event. For example, if B no longer trusted A on what C said in the past then (Formula (3)) then a new protocol must be derived.

Much work remains to be done on this architecture on a number of fronts. We have slightly extended the Simple Logic and synthesis framework to better suit the task at hand. In tests, the protocol generation tool has performed well

[13]: given mutual authentication/key exchange goals and assumptions that indicate a trusted third party, then the tool can generate approximately 500 valid (4/5 message) mutual authentication protocols within 40 seconds. On inspection, many of these candidate protocols are similar containing minor textual and redundant variations; we estimate that in this set there are 24 reasonable distinct four-message protocols and 76 reasonably distinct five-message protocols. Whether there is sufficient variety in such protocols for them to be used as "session protocols" is a topic for future research.

Currently, the architecture expects the protocol initiator to synthesise the final protocol. One challenge is to determine how assumptions held by other principals can best be discovered by initiating principals, and whether it would be better to hand-off sub-goals to other principals to synthesise.

Other future research includes exploring how to automatically translate an idealised protocol into executable software components. Some previous work [12] exists on this problem that we hope to draw on.

Acknowledgments

This work is supported by the Boole Centre for Research in Informatics, University College Cork under the HEA-PRTLI scheme.

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.
2. A.G.Ganek and T.A.Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, January 2003.
3. Tuomas Aura. Strategies against replay attacks. In *Computer Security Foundations Workshop*, pages 29–68, 1997.
4. L. Buttyán, S. Staamann, and U. Wilhelm. A simple logic for authentication protocol design. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 153–163, Washington - Brussels - Tokyo, June 1998. IEEE.
5. John A Clark and Jeremy L Jacob. Searching for a solution: Engineering tradeoffs and the evolution of provably secure protocols. In *proceedings of 2000 IEEE Symposium on Security and Privacy (SP 2000)*, pages 82–95. IEEE Computer Society, 2000.
6. Li Gong, Roger Needham, and Raphael Yahalom. Reasoning About Belief in Cryptographic Protocols. In Deborah Cooper and Teresa Lunt, editors, *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, 1990.
7. Joshua D Guttman. Security protocol design via authentication tests. In *proceedings of 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 92–103. IEEE Computer Society, 2002.
8. Sun Microsystem Inc. Jini technology core platform specification version 1.2. www.jini.org, November 2001.

9. D. Kindred and J.M. Wing. Theory generation for security protocols. *ACM TOPLAS*, July 1999.
10. D. O'Crualaoich and S.N. Foley. Theory generation for the simple logic. Technical report, University College Cork. In preparation.
11. Adrian Perrig and Dawn Song. Looking for diamonds in the desert — extending automatic protocol generation to three-party authentication and key agreement protocols. In *proceedings of the 13th IEEE Computer Security Foundations Workshop*, pages 64–76. IEEE Computer Society, 2000.
12. D. Song, A. Perrig, and D. Phan. Agvi–automatic generation, verification, and implementation of security protocols. In *proceedings of 13th conference on computer aided verification CAV 2001*, pages 241–245, July 2001.
13. H. Zhou and S.N. Foley. Fast automatic synthesis of security protocols using backward search. In *ACM Workshop on Formal Methods for Security Engineering*, Washington, DC, USA, 2003.

Towards a Framework for Autonomic Security Protocols
(Transcript of Discussion)

Simon N. Foley

University College, Cork, Ireland

We've been looking at what we call the security protocol negotiation problem. People who use security protocols to authenticate, do key exchange, or whatever, are typically stuck with using whatever protocol is available. If the participants can't agree on a suitable protocol, then they don't get to use each other's services. We are interested in self-configuring security protocols where the participants have to say what their requirements are and synthesise or configure their own suitable protocol that would meet their respective goals. If we have this idea of self-configuring security protocols then, as a consequence, we also have self-healing security protocols. If the requirements change, or if the assumptions about our principals and keys change, then we have this ability to reconfigure and produce a security protocol that will meet those new goals, based on those assumptions.

To motivate the work we use a very simple example based on the movie 'When Harry Met Sally'. There's these two people in the movie and they don't get on at all. The problem is that they both have their own strategies for interacting, for getting what it is they want. And so Harry has a bunch of strategies that he uses when he tries to chat up women, and Sally has her own strategies, and they can't agree on a common strategy, so there's no interaction. Only after a lot of time do they realise that they need to reconfigure their strategies, use different ones that will be consistent with each other, and so what we have is self-configuring and self-healing, and survival strategies. And these are properties (among others) of autonomic systems. The properties that we're interested in are self-configuring, *i.e.* the ability to compute their own protocol and use it, and self-healing, *i.e.* the ability to recompute a new protocol if something goes wrong, so that you can continue interaction.

Currently if I'm using a particular protocol like SSL, it will start off by negotiating the particular ciphers that we want to use, the version of the protocol, and so on. In some cases we may be willing to negotiate a particular protocol, but it's always going to come from a fixed menu. So the participants, Harry and Sally, have a fixed menu of protocols that they're willing to use, and Harry says to Sally, "which protocol do you want to use when you interact?" And the problem is that if they if can't agree on one of the fixed pre-programmed protocols then they get no service. Also fixed protocols are based on a fixed set of requirements and assumptions, and so on, so we have very limited reconfiguration in response to any environment changes. If the key gets compromised, then we can probably react to that by re-keying, but there are a number of other things that can go wrong which we can't react to, because we're just using a fixed set of protocols with these fixed assumptions.

Matt Blaze: Quick clarification, are you using the word protocol in the usual meaning or is there some subtle distinction between a protocol and its parameters?

Reply: I'm using it in the usual sense: the description of the flows as opposed to just the configuring part. Most existing protocols won't even allow you to change the configuration, except in some limited cases, although sometimes you can use different versions of a particular protocol, in effect using sub-protocols of some larger protocol.

One of the few architectures that might provide some kind of self-configuration is protocol-agnostic Jini. Here Sally first of all registers the protocol with the Jini server, and in effect she's saying, if anybody wants to talk to me, then the protocol they need to use is going to be available from the server. Jini has been used in systems which people would claim to be self-configuring, self-healing systems. However, the difficulty is that Sally is just registering a single fixed protocol. But perhaps Harry has his own particular collection of requirements, and he doesn't want to follow Sally's protocol because it's not got certain properties that he's interested in.

So we want to provide some kind of framework so that complete strangers can negotiate and generate suitable protocols on the fly. These protocols are going to have to be self-configuring, they're going to have to figure out what message exchanges are going to be necessary to keep the parties involved happy. And we want them to be self-healing, so that if something happens at one of the principals that they decide they don't like, or if they don't trust certain aspects of particular principals anymore, then perhaps they want to go away and renegotiate a new protocol.

So obviously what we're going to be doing is synthesising a protocol from the collection of assumptions about the principals, and a collection of goals. The goals are what it is we want this protocol to do before we're willing to interact with somebody. And so we get a collection of assumptions about keys, principals, and so forth, we have our goals which are the requirements for our final interaction, and what we're going to do is automatically synthesise a protocol on the fly. The architecture works something like this: Harry and Sally both have their own assumptions and their own goals, Sally has her goals which are, 'for me to provide service to somebody this is what I expect before we start interacting.' When Harry contacts Sally and says, "I'm Harry," Sally then provides her assumptions and goals to Harry. Harry has then got to go away and synthesise the protocol using his own assumptions and goals plus those of Sally. And having synthesised the protocol he would send his protocol plus his own assumptions to Sally so that she can then go ahead and verify the protocol given the assumptions and her goals, and if everybody's happy then they will begin to install and then execute the protocol. So in very, simple terms that's how the synthesis should work.

We decided we'd start off using SimpleLogic, a simple, easy-to-use, BAN-like belief logic. What's interesting is that it provides us with these synthesis rules that guide manual protocol calculation from goals to assumptions. So you can type in your goals and SimpleLogic says, 'this is your goal and these are the steps which given the particular rule could lead you to this goal.' You are in effect doing a backward synthesis towards the protocol that would meet that particular goal; synthesis in the true sense of the word, it's like protocol calculation. And so we use these synthesis rules to guide automatic backwards search for a protocol that meets the goals. Now unlike other work, our synthesis rules are in effect directing our search of the security protocol space. We're not having to search the entire space of all protocols, because we're working from the goals backwards towards potential protocols, so we're only ever going to be producing protocols that are correct within the logic. There's a lot of heuristics that we do to try and limit the protocol search space because we're trying to do this on the fly, we certainly don't want Harry to have to go away and spend two hours synthesising a protocol to give to Sally. We've got reasonable performance in our prototype which will do an on-the-fly synthesis in two seconds to find a three party protocol.

So the logic gives us synthesis rules for synthesising protocols manually, and we have a tool which will automatically search all possible avenues afforded by those rules trying to find suitable protocols.

Here's a really simple example: Sally wants to authenticate Harry. What we're trying to synthesise as a goal here, is that Sally believes that Harry recently sent Sally some fresh nonce N_s. You can see that it has synthesised four possible alternatives just from that goal. Here's the one that we're particularly interested in (here it's done manually, but the tool that we have does an automatic search as well). It says that Sally believes that Harry said N_s and Sally also believes that the nonce is fresh. So we, to select the particular avenue that we're interested in. Mark one of the assumptions – that Sally believed the nonce was fresh – if we now look at the current state of our protocol, we have a sub goal that we have to try and synthesise further backwards.

We saw there were four alternatives, and here's two of those possible alternatives: one is suggesting a public key where Sally knows Harry's public key, and this one's suggesting Harry and Sally share a symmetric key. Sally goes ahead and selects the public key version, and does some cleaning up, and here's my final protocol synthesised. I've found the original goal, I've evolved the final running protocol which is in effect, Harry sees a challenge which is S and N_s from Sally, and then Sally sees the S and N_s back again signed by Harry's private key.

So that was to give you a feel of how you do it manually. We've got a search engine which will search through all the possible avenues automatically. Harry says to Sally, 'I'm Harry,' and Sally knows Harry's public key, and she wants to authenticate him. She has her own beliefs, which are her assumptions, and she also has her particular goal which is, if anybody wants to talk to me I'm going to have to authenticate them, I want to know who they are. Are they who they claim to be? And so she sends her goals and assumptions to Harry who has to go off and synthesise the protocol which is more or less the one I just sketched, this is done automatically. Sally will get that protocol, she uses a verifier (which we have) to do the verification and then finally get her own protocol, and hopefully everybody will be happy.

So it's a self-configuring protocol. We can add in a self-healing protocol. Suppose Sally no longer believes Harry's public key, for example. She's used it in the past but now she's not sure whether she should trust it anymore, so she throws it away, she removes it from her assumptions. But she trusts a mutual friend Jess. So let's suppose this is her current set of beliefs, her assumptions: in simple terms that she's sharing a symmetric key with Jess, that Jess is honest, Jess is competent, and we assume Jess also has a collection of assumptions, and Jess knows Harry's public key. So Harry's got to go off, compute a new protocol, send the protocol to Sally, and establish communication again.

One problem is a synthesiser in the middle. Sally's giving work to Harry to synthesise the protocol, and Harry sends the protocol to Sally, but of course you can have some agent in the middle who gives a different protocol to Sally, and in effect is trying to do a protocol interleave attack. The attacker knows he can't attack the individual protocols, but stick them together and we have a vulnerable protocol, so initial goals have to be checked, and one goal is that Sally needs to believe that Harry actually sent this protocol which she's received, and also that Harry will want to believe that Sally got the protocol as well. So there's a bit more to the synthesis than just the protocol.

Matt Blaze: Doesn't Sally also have to protect against Henry, or someone in the middle, sending her a protocol that will convince her to do things with her keys that can be used in other protocols? For example turning her into an on-line oracle for signing things with her key.

John Ioannidis: Or for that matter, how can Sally know that the protocol that Harry's suggesting doesn't actually have her say, "here's my key"?

Alec Yasinsac: How do you know that there will be a limited number of protocols that these requirements will be synthesised to; for any given objective, there are probably seven million ways to get to that objective, and so how does your synthesiser pick the good ones, as opposed to the not so good?

Reply: We have some results which I can give you in a few slides which will hopefully answer those questions in terms of the experiments that we've run, and the protocols we've synthesised. If you look at existing protocol synthesisers, the automatic protocol generators are searching huge spaces of all possible protocols. We're narrowing it down so we're only actually looking amongst protocols that we know will be valid within the logic, because we're doing this backward search from the goals, as opposed to a straightforward forward search as the other techniques do. So we're not actually having to verify these security protocols within the logic because we're guaranteed to produce individually secure protocols within the logic. Of course, it's only a belief-logic based verification, so it's limited, and you might want to also do some further validation to any protocols that you generate by running a different checker.

John Ioannidis: What are the primitives out of which you can compose protocols? Are they at the level of "send this message"? Are you also worried about things like making it failsafe?

Reply: The primitives are just straightforward high level descriptions of protocols. We haven't got to the stage where we're looking for ways to write robust protocols that are less likely to be vulnerable when you start composing them with other protocols.

Ross Anderson: There's one level back from this as well: it's not enough just to have a naming convention, so you won't get a chosen protocol attack[1], you've also got to worry about somebody having a meta-synthesiser computer. If you've got a protocol which said you will put a hash of the name of the person who thinks he owns the protocol then you could attack that by saying in this *other* protocol synthesiser we'll simply give a table of 160 bit words which will be allocated as follows. . . There's no obvious solution to this within any simple structure.

Reply: This is very preliminary, we're just trying to see whether such an approach is even feasible within a reasonable time. If I move to our preliminary results, we've looked at some simple mutual authentication protocols with a trusted third party, and run them through the synthesiser. The synthesiser has cost functions associated; it just generates protocols of a certain size, so that it's not going down unproductive blind alleys, so in three seconds it could produce a valid protocol. And if you look at the Perrig and Song work[2], it takes them ten minutes to produce a best protocol according to their criteria. If you look at a mutual authentication and key exchange protocol, it's usually a key distribution centre, then we can produce a four message protocol, again working within the logic. In ten seconds it will generate 24 possible valid protocols, while it would take two hours in Athena to produce a protocol. And we can do that for five message protocols.

So it does seem to produce this relatively quickly, though all of this is conditioned on working within authentication logic so we recognise there are potential vulnerabilities there.

Alec Yasinsac: I'm still not sure that I understand how it works backwards to narrow the scope. There are lots of ways within this authentication logic that those goals can be met, unless you've given it some standard mechanisms – in which case it becomes not a formal method but a standard compiler.

Reply: I guess in the synthesiser there are some sort of engineering principles which we apply. Obviously if you do the backward search like that, you could potentially have an infinite search space, so we're stopping at a certain depth,

[1] See J. Kelsey, B. Schneier, D. Wagner, "Protocol Interactions and the Chosen Protocol Attack", LNCS 1361, 91–104.

[2] Adrian Perrig and Dawn Song. "Looking for diamonds in the desert — extending automatic protocol generation to three-party authentication and key agreement protocols". Proceedings of the 13th IEEE Computer Security Foundations Workshop, pages 64–76. IEEE Computer Society, 2000.

which is in effect the number of messages in the protocol. You give us your goals, initially four possibilities, and then we explore those four possibilities as sub-goals, and turn to the sub-goals within sub-goals, and so on. We stop after a certain depth, and prune the tree at this point, and backtrack up. We're not searching the entire space, we're limiting what we can find by what is, in effect, the number of messages in the protocol.

OK, so the logic based synthesis makes it potentially feasible to generate protocols which are self-configuring, self-healing protocols. There's some work been done by Berkeley people on automatic code generation from these descriptions. One issue is how we actually detect changes in assumptions and goals so that a principal can recognise if something goes wrong and know how to change their assumptions and goals. And lastly, while we are synthesising mutual authentication key exchange protocols there are obviously other protocols that should be worth looking at as well.

Pekka Nikander: You had these additional requirements that Harry must believe that Sally has got the right protocol and so on. How do you actually get there; do you need to feed that through the protocol generation? Can you model that in your current framework, what happens to the search?

Reply: That's an interesting question and we haven't got to that yet.

Brian Monahan: I think this is extremely interesting work. Going back to your slide on the protocol-synthesis protocol, why did you decide that there was no negotiation of the goals?

Reply: I guess, because it's just preliminary at this point. We haven't actually come up with the final protocol, this is just a very basic protocol-synthesis protocol. Like Pekka said, there are other goals which we would need to consider to make it a fair protocol.

Brian Monahan: And equally only one party generates the program? So in actual fact you might imagine that to be a mutual process?

Reply: Sure. Another possibility might be where Harry can't synthesise the sub-goals any further, and then contacts his friend and says, "look here's the sub-goals, can you help me out here?" You're not just synthesising protocols between people, it could be a distributed synthesis involving many different parties.

Brian Monahan: We did try to do a little bit of this work. I ended up building a checker for a very small range of protocols; we invented the goals and the initial conditions, and we produced a technical report. But you're right, you do want to go beyond the BAN logic, we were using the strand logic.

Reply: We've augmented the synthesis tool to use strand bases to get better at designing protocols, but the thing is, while the protocols you're producing are probably better than simple logic protocols at the moment, the only guarantee that we have is that they're valid protocols within the logic. But we believe we're producing stronger protocols which we could then look at and analyse.

Brian Monahan: Are you using a genetic algorithm rather than just back-chaining and rules? I mean obviously, if you're restricted by the kind of logic that you use, so again, that might be a more creative way of approaching this.

Reply: Yes, choosing such a simplistic restrictive logic means that its usage is not restricted. If you look at the Athena-based generator which takes two hours to produce a simple two party exchange then obviously if it's not feasible, it's not useful.

Frank Stajano: You say that people with specific assumptions should synthesise a new protocol just for that case. One could object that people have only a certain number of limited ways in which they interact so they might as well just look for a library. Make a hundred proofs and look behind them instead of just looking up. If principals want to precisely specify something which is not in this library of a hundred protocols, how do we know that they're going to draw up the assumptions and the goals correctly? It looked like complicated formal methods that people like Harry and Sally are likely to get wrong.

Matt Blaze: It works better with Alice and Bob. They have more experience. [Laughter]

Frank Stajano: So if Sally is supposed to think out the assumptions then she'll probably be making fewer mistakes by selecting from a catalogue of things that have been tested by people in this room?

Reply: I guess that criticism can be levelled at anything: you know, is Sally competent to write a set of beliefs? In fact, is Sally competent to do anything?

Alec Yasinsac: What you assume is that you have a heterogeneous network in which Sally's probably got her set of beliefs about what she wants pretty well worked out, and she's interacting with somebody different who has a different set of beliefs. She's not going to do this every time she's going to talk to Bob, she just does it once and she keeps track of those beliefs that she refines. But I think there are a lot of legitimate places where this can work. In Camp David, for example, we've got lots of different people with different security goals, and general approaches.

Frank Stajano: So would it be the case that people study their own assumptions and goals very carefully, and the complexity only comes when they interact with other people who don't share them?

Alec Yasinsac: That's absolutely an issue you have to deal with. When you get to a place where somebody needs to negotiate then you'll get told a different set of problems.

Reply: Another point is, if you can't get your assumptions right how do you know you can get your one hundred protocols right? And in the list of a hundred protocols, maybe protocol number 98 shouldn't be there? At least this way I know what the goal is, it's just a simple expression.

Birgit Pfitzmann: What is the range of the goals? If there is only a certain set of goals, then you can pick up one provable that would prove all the goals you express (because there are no negative goals in your logic). So can't you just put them all in one universal goal? Even if it's got 50 clauses, that might be quicker than your search? And you might be able to take the trust management out, so that all goals can express trust as well.

Ross Anderson: Sometimes Alice wants to impose a non-repudiation on Bob, sometimes this is unacceptable to Bob, therefore there is no universal protocol.

Bruce Christianson: That's a very interesting point, can you express lack of non-repudiation[3] as a goal in this logic?

Ross Anderson: OK, let me give a simpler example then. Sometimes Alice and Bob want to share a secret that's generated by Alice, sometimes they want to set up one that's generated by Bob, for a covert channel, or trusted hardware, or other such reasons.

Birgit Pfitzmann: You could express that in the protocol by having a secret generated by me joined to a secret that's generated by you.

Bruce Christianson: Generate my secret for me and get your own secret free?

Ross Anderson: You might have a goal that says you don't want to use modular arithmetic, if it requires quantum computers. [Laughter]

Mike Roe: There are conflicting goals: in some protocols you might require that Alice be anonymous, and in others have a requirement that Alice be authenticated, so clearly there's going to be no single protocol that meets all the requirements. We're worrying about how many protocols there really are. If there's only some small set that requirements express then you can work them all out, and get some experts to look them over to make sure.

Bruce Christianson: But Simon's argument is that his approach gives you an evolution path, which allows you to put in things that you didn't think of to begin with and take out things you decide were a bad idea.

Ross Anderson: But that would make everything far too explicit; it would make the hidden goals of the protocol visible and the power relationships of the principals all too overtly villainous (and likely to fail in the marketplace).

Brian Monahan: I think the scenario that Simon presented right at the beginning was extremely good, and probably shows exactly what's going to happen. In the future there are going to be lots of different kinds of systems all trying to do business, and how do you get the business, unless you have this distributed workflow idea which somehow captures that model?

[3] This property is also called plausible deniability. See M. R. Roe, Cryptography and evidence, Ph.D. dissertation, University of Cambridge, Cambridge, 1998.

Client v. Server Side Protocols, Interfaces and Storage

Geraint Price

Information Security Group, Mathematics Department,
Royal Holloway University of London, Egham, Surrey,
TW20 0EX, United Kingdom
`geraint.price@rhul.ac.uk`

Abstract. We present a model for increasing the effectiveness of the communication between an application and a security protocol. The model is a development of previous work by Syverson and Meadows. It also incorporates our notion that transferability and control of data across a secure interface can improve the use of security services by applications.

1 Introduction

The design and analysis of security protocols has been the mainstay of security research for the past 20 years. The analysis has been primarily focused on the protocol's ability to distribute shared keys between communicating parties, or to convince one party of another's identity.

We believe that there needs to be an extension of this analysis to understanding how these key sharing and authentication protocols affect the data which is subsequently secured using these keys and assertions.

In this position paper, we propose a strategy for expanding the analysis from the protocol to the applications that these protocols serve. The notion of the application we present is one of either an end-point in a client-server interaction, or a peer in a peer-to-peer communication.

The ultimate focus of our discussion is the interface that lies between the security protocols we design and the applications that use the services the protocols provide. To help clarify this point, figure 1 represents a traditional model of the protocol stack at a host. The peer interaction at the *security* layer is where we would traditionally find the security messages being passed from one protocol participant to another[1].

Our aim is to extend the analysis from the consideration of the messages themselves to the effect they have on the data directly manipulated by the protocol. In pursuing these goals, we are driven by two factors:

– Firstly, the observation that protocols do not operate in a vacuum. It is a traditional viewpoint – with which we broadly agree with – that security is

[1] Such as Alice sending an encrypted nonce to Bob as part of a protocol run.

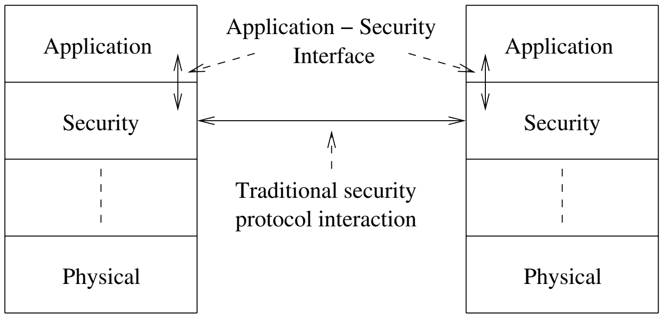

Application
Application – Security Interface
Application
Security
Security
Traditional security protocol interaction
Physical
Physical

Fig. 1. Generic protocol interactions

best left to be designed and implemented by a specialist. In an ideal world, all the security services required by an application would be designed into an application directly. Unfortunately, this is unlikely to be the case and the situation is surely going to get worse before it gets better. With an increase in commercial off-the-shelf (COTS) security products, we believe it is important that we try to influence how these products are used by applications. In a recent paper by Kain et al [8], they note that they were able to manipulate the digital signatures on electronic documents. One of the reasons they were able to do this was that there was poor binding between the application and the security package. It is important that we both try and learn what using these COTS security packages provides to an application, as well as providing a more meaningful and intuitive interface for these applications to use.

– Secondly, in earlier work [12], we demonstrated how the control of protocol elements could be used to redress an imbalance in trust. Given the nature of distributed communication, understanding how the transfer and control of data in a similar manner might be useful for an application could provide a fruitful avenue of research. This should also help an application to make intelligent use of a security service.

The remainder of this paper is organised as follows. In Section 2 we review an existing, closely-related model, looking at how to adapt it for our requirements. In Section 3 we provide an overview of our notion of control and its transfer within a distributed environment. Section 4 outlines how we aim to adapt our principles into the model, and we draw some initial conclusions in Section 5.

2 Communication and Logical Models

In this section we introduce the both the communication model we base our interactions on and the formal model we intend to use and adapt.

The early work on protocol analysis was primarily focused on the participant's understanding of the messages and keys they exchanged with others [3].

In an open communication environment where message tampering is a possibility, a client can only attest to a data item they have received from a third party if that data item has been secured in some manner. Thus, if we are able to analyse the underlying communication protocol from the participant's point of view, then it would appear to be natural to extend the analysis from the protocol to the data.

We now briefly describe the communication model which we are interested in, where we consider a distributed computing environment. Our discussion is primarily focused on the interaction between a client and a server, or peer-to-peer interactions that are supported by the security service of a third party [2].

We concentrate on this communication model for two main reasons:

1. If it is possible to demonstrate a strong trust relationship between the communicating parties, then there is the possibility we could reduce the complexity of both the protocol and interface.
2. If there is a weak trust relationship, then there could be a requirement for increased control and interface awareness at the client (or peer) end of the protocol.

The attack of Lowe [9] on public key Needham-Schroeder [11] provides an illustration of what we mean in our discussion above. The original assumptions of the protocol were such that they discounted a malicious insider, which Lowe exploited.

Our work will help us to develop a better understanding of the affect of the security protocol has on both the application interface and the data shared between entities. This should help isolate the differences highlighted in the two examples given above. Achieving this would help us make better informed decisions about what the protocol should provide in relation to the application's security requirements.

2.1 Overview of Previous Work

Gaining an understanding of the requirements on a protocol from the environment that uses it, is not something new. While the BAN logic of Burrows, Abadi and Needham [3] sparked a surge in interest in formally analysing a protocol to ensure that it met its goals, their work did not address the appropriate requirements for an application.

Syverson and Meadows [15,16] addressed this problem by developing a logical language for the specification of cryptographic protocol requirements. Because the requirements and assumptions surrounding the protocol can vary greatly, their language makes it easier to state and reason about the protocol's requirements. This allows them to specify a set of requirements for a whole class of protocols and not just one protocol at a time. This allows them to separate the reasoning stage from the specification of each particular protocol.

Syverson and Meadows were then able to use the requirements language to test protocols to see if they met these requirements. The testing was carried out using the NRL Protocol Analyzer, by searching for the reachability of undesirable

[2] Such as a CA in a PKI or the Authentication Server in Kerberos.

states according the specification. Their goal in this work was to present a set of formal requirements which can be applied at the design stage.

More recently, Canetti, Meadows and Syverson [4] discuss the limitations of such formal mechanisms on identifying the requirements of security protocols. Their discussion focuses on the lack of consideration most formal mechanisms have for the environment in which the protocol is executed. Here are the key points of their research which are of interest to us:

– In terms of the *environment* they discuss in their paper, they consider the following four factors: (a) the intentions and capabilities of an attacker; (b) the applications using the protocol; (c) the functions on which the protocol depends, and; (d) other similar protocol executions with which the current execution may or may not interact.
– The aim of their paper was to demonstrate that even the strongest requirements are not enough to guarantee the security of a protocol, unless we put restrictions on the environment as well as the protocol.
– Controlling and specifying the application environment is obviously more difficult than the security protocol. Even so, without some form of formulation on how the environment should behave, they are able to demonstrate attacks even on fail-stop-protocols [7].
– They conclude their paper by discussing a set of *rules of thumb* which are similar in manner to the *prudent engineering practices* of Abadi and Needham [2]. Of these, the 6th is the most interesting to us here: "*Any fix to the environment should be made as close to the protocol as possible.*"

We agree with this sentiment entirely, and it matches our belief that the requirements on the interface is the logical place to provide such mechanisms.

2.2 Where Our Needs Differ

We highlight existing work in the previous section and use it as a basic construct for our work here. We now proceed to discuss why what we need from such a model differs slightly from what has gone before.

First, in terms of the requirements, we note that the work by Canetti et al discusses the difficulty of formally modelling the requirements of the protocol for every environmental instance. We believe that this becomes more important as security protocols are being pushed into wider contexts. For this reason we treat the formal requirements stage as only a building block in the process.

Second, if we look at the transferability property we discuss in Section 3, we notice that it is a concept more suited to dissection process. By this we mean that we can represent the general notion of what we are trying to achieve in multiple independent but related concepts. It is for this reason that we have chosen to add a taxonomy to the modelling process.

Finally, from Canetti et al, they promote the use of strengthening of protocols to include the names of the recipients. They do however note that this is not always possible. Taking limitations like this into account is the challenge that needs to be met when increasing the ability of an application to make use of a security service.

2.3 A Three-Tiered Approach

We will now outline the structure of our approach. We build one layer of analysis upon another, ultimately providing input to the application/security layer we discuss in the introduction. We provide three layers to our model, built as follows:

Taxonomy layer. The first layer will be a taxonomy, not unlike that of Syverson's taxonomy on replay attacks [14]. At this level we will classify the different ways in which cryptographically protected data can be shared between peer-entities in a distributed environment. This will help us isolate the data control and sharing notions we discuss in Section 3.

Requirements layer. The requirements layer will take the various data sharing and control strategies of the taxonomy layer and provide a set of requirements for protocol designs. We will adapt the model of Syverson and Meadows for our work here [15,16].

Interface layer. The interface layer will then take the requirements as guiding principles for how the applications should interact with the various protocols. This will depend on the element within the taxonomy which the application requires.

The ultimate goal in this analysis is to provide the interface layer with a clear set of tools, from which an application can easily make use of the transferability properties we discuss in Section 3.

The rule set should provide similar direction as provided by the *prudent engineering principles* of Abadi and Needham [2] as well as the *rules of thumb* discussed in Canetti et al [4]. Stepping over the border from the security to the application would appear to go against the traditional *black box* view of cryptography. As we mention previously, it is important to recognise the reality of the situation. In fact, Canneti et al themselves allude to a limited but controlled extension of security awareness:

> "*RoTh 6. Any fix to the environment should be made as close to the protocol as possible.*"

3 Transferability and Control

In this section, we briefly outline what we mean by the control of data between participants in a secure communication. We start by looking at some past work, then broaden our discussion to provide new avenues for understanding how the notion of control can be used for data.

3.1 Past Work on Control

In the past we have used control of the flow of a protocol, coupled with the freshness elements of that protocol as a means of ensuring servers are diligent in their actions [12]. In brief, our work allowed the monitors of resources on a network to dictate the freshness requirements of a particular protocol. This allowed the monitor to convince itself that the server was either completely corrupt or live and active in the protocol run. Previous designs of the protocol allowed the server to be *lazy* and pre-compute tokens for distribution by an accomplice.

Such mechanisms tend to increase the complexity of interfaces and protocols. In return, there is the possibility of a reduction in the storage requirements for some protocol participants.

3.2 Some Themes on Transferability

In this section we will talk about some of the required properties we seek in maintaining the control of the data. In broad terms we will call this property *transferability*. Although we will not formally define our property, it expresses the ability to take control over an object, or to demonstrate to another that control.

When comparing it to traditional security properties, we consider it to be a mixed representation of integrity, authenticity and timeliness of data.

Evidence. We briefly discuss how the means by which a cryptographically secured data item is generated can affect the way in which it is relied upon.

We consider a scenario in which Alice processes a signature from Bob as part of a protocol run. Alice may decide as part of the validation process to carry out an OCSP [10] check before acting upon the information present in the signed message.

What Alice subsequently does with that OCSP request depends on what security properties she requires from the transaction. In the context of our current discussion, is she likely to want to transfer the knowledge that the OCSP request represents, to a third party, at a later date?

If, at some point in the future, Alice realises that she has been duped into doing something foolish, she may wish to pursue Bob for compensation. At this point Bob may try to repudiate the original signature.

The security property that is represented by the response to the OCSP call (i.e. freshness and authenticity) is now represented in the data item Alice received as part of the protocol run.

If Alice was prudent, after running the protocol, she would have kept the signed receipts from the OCSP responder. This, in conjunction with any other data she used to verify the signature at the time, could be used to hold Bob to the signature at a later date as well.

In this case, we need to understand who is concerned with the integrity of the data generated under the OCSP request. Once a message has been signed or encrypted, not only is the means by which the message transferred important – in this case that particular run of the OCSP protocol – but what happens to the data afterwards is equally important.

To generalise: “Who should look after what?” Once a message has been signed or encrypted then, in this case, it is up to Alice to make sure she holds onto the signed OCSP response for potential future use[3].

We return to our discussion of evidence in Section 4.

Control. We now look at how the control of data used on an interface, and where that data is stored, can be affected by the trust assumptions between the communicating parties.

We use XKMS [1] as an example. In a traditional PKI environment there is potentially a heavy burden on the client to perform processor intensive operations. XKMS permits the delegation of trust processing decisions to one or more specialised *trust processors.*

A key objective of the protocol design is to minimise the complexity of the end user implementations, by allowing them to become clients of the trust processors. This then shields them from the complexity and syntax of the underlying PKI used to establish trust relationships.

In this case, the storage and control of security sensitive data is transfered from the client application to the trust processor within the XKMS architecture. Understanding the extent to which the client can offload the processing and storage of security critical data is crucial if we are to ensure that the actions carried out by the trust processor conforms with the client’s security policy.

In the stack model we present in figure 1, we consider the XKMS client and trust processor to be peer-entities in the security layer. It is then important for the application making use of the XKMS client to be aware of the service being implemented. In this case, this is the processing of PKI certificates, policies etc.

The interface between the XKMS client and the trust processor has to be able to express the right level of control to minimise the complexity of the protocol between the client and the server, whilst still expressing the security requirements of the client application.

Plausible Deniability. We briefly consider how plausible deniability [13] – originally the concern of authentication and key exchange protocols – could be considered part of our transferability model.

Although we only briefly touch on the subject here, it provides a good illustrative example of why we should be concerned with the effect of the underlying protocol on data.

Assume that an application is making use of a key distribution protocol with plausible deniability built into it. If the application is not careful with what he subsequently uses that key for, there is the chance that the application could negate the plausible deniability provided by the protocol. This could happen if the application were to subsequently encrypt or sign some authenticating information.

[3] We do note that in an ideal world, Alice would use a full dispute resolution protocol. As we noted in the introduction, one of the driving factors for this work is to make safer use of the systems we have. Canneti et al [4] discuss a variant of this, where they note that it is not always possible to tailor the protocol to the environment.

Is it delegation? The question we posed when initially discussing our concepts was: Is what we are presenting just delegation?

Delegation as a property, and protocols to implement the property have been part of the security literature for the last couple of years [5,6].

Although our thoughts on this are not entirely complete we believe that, while some of what we present could be seen as delegation, our notions of transferability could also be seen as a super-set of those properties. While the previous example with XKMS could be regarded as delegation, we provide two examples of our property which we do not believe are delegation:

- Alice might explicitly take the right to control a data item from someone else. In some sense, this could be seen as the reverse of delegation.
- In the notion of evidence we discuss above, Alice may purely wish to hold on to the evidence for her own peace of mind. This does not seem to be an embodiment of an authorisation in the same manner as traditional delegation.

4 An Outline and Future Work

In this section, we provide a starting point for our analysis. We take one of the principles we present in Section 3 and outline how it fits in with the three-tier model we discuss in Section 2.

4.1 A First Example

We now provide an example principle within the three-tier model we propose. The characteristic we use as an example is the *evidence* principle.

Under the headings below, we break down how it would fit into our three-layer hierarchy:

Taxonomy Layer

- The notion of evidence we provide here belongs to the category of data control elements within the taxonomy. The taxonomy is likely to be split into two broad categories: data control and flow control. There would also then be sub-categories of traits that will belong to items in both branches of the taxonomy.
- Transferability would be one sub-category of the data control branch of the taxonomy which will separate evidence out from other types of data control structures. This will have an impact on the choice of cryptography that the requirements layer is likely to support.
- Yet another defining feature of evidence is its timeliness.

Requirements Layer

- Formalising the requirement for forwarding a cryptographically authenticated message to a third party is likely to require the following elements within Syverson and Meadows's model, where the first requirement might be modelled as follows:

`forward`$(user(A, honest), user(B, Y), (), (M, X))$
  - The ability for one party to forward a message from a second party to a third party.
  - The ability for another party to receive that message
  - The ability for a party to learn the key associated with the message
- Formalising the notion of timeliness within the language should be straightforward, given that it uses a state space. It would require us to add the notion of learning the time associated with a particular message. They already possess this ability in the requirements to model the timeliness for keys [16].

Interface Layer

- From the requirement of transferability that implies use of asymmetric cryptography, we would ensure that the application was able to verify his relationship as relying party to the authority that signed the certificate (whether that was a CA or an OCSP responder).
- If the application is requesting a real-time validation, question whether the application needs to store this. In the process, clarify the use of a real time element to the freshness. Warn that if they use a nonce-based approach the stored data is not of any use to them if they want to combine it with the "showing" element of the asymmetric cryptography.

While our initial descriptions above are informal in nature, they demonstrate the way in which the nature of the control is expressed at the three different levels.

4.2 Progression of Our Model

Given that we are still in the process of building our model and that the example presented above is unlikely to be the final word on the control of data as evidence. How do we see our model progressing from here?

Taxonomy Layer. In our mind, it is the taxonomy layer that has the greatest requirement for completeness, mainly due to the fact that it is a building block for our upper two layers.

We need to provide a more comprehensive description of the data and flow types which we have isolated so far.

We will continue to look at other related security primitives such as delegation. Firstly, to learn from and build up similar models for our notions of transferability. Secondly, to incorporate the existing body of knowledge into our model.

Requirements Layer. We need to build a catalogue of requirements. Although the use of a requirements specification language is slightly different to a traditional logic, studying unified logics such as Syverson and van Oorschot [17] will provide us with a good starting point. They have already drawn together the useful functional elements from various logics to provide a combined view of their processing elements.

Once we have a list of requirements, then a refinement plan from the requirements list to the taxonomy would be next. This will allow us to build sets of constraints that relate to particular sections of the taxonomy.

Interface Layer. This is likely to be the most difficult part of our model to get right.

One reason why we wish to build it on top of a requirements level is that the delivery of the principles in the taxonomy to the application should not rely on a single instance of a protocol or mechanism.

We might also want to provide guiding examples, as case studies, of what is likely to be safe use of the requirements depending on each given section of the taxonomy.

5 Conclusions

We have presented a model for developing a better means of interacting between the application and a security protocol. Our model is partially built on the work of others, and partially on our own ideas of transferability.

We believe that it is important that we develop a greater understanding of the role played by both interfaces and data shared on those interfaces in relation to security protocols.

Carrying out this work will allow us to highlight the issues faced when binding a data processing function to a security service. In a paper by Kain et al. [8], they note that one of the reasons they are able to manipulate the digital signature on an electronic document is because of the poor binding between the application that displayed the document and the PKI that enabled the signature functionality. The status of this is unlikely to change, thus, developing methods for more considered use of security services is important.

Acknowledgements

We would like to thank Birgit Pfitzmann for her helpful comments on the initial presentation of this work.

References

1. XML Key Management Specification (XKMS 2.0). W3C Working Draft, March 2002. http://www.w3.org/TR/xkms2/.
2. Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. Technical Report 125, Digital Equipment Corporation Systems Research Centre, June 1994.
3. M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 246:233–271, 1989.

4. Ran Canetti, Catherine Meadows, and Paul Syverson. Environmental Requirements for Authentication Protocols. In *Proceedings of the International Symposium on Software Security*, volume 2609 of *Lecture Notes in Computer Science*, pages 338–355, November 2002.
5. B. Cripso. Delegation protocols for electronic commerce. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, July 2001.
6. B. Crispo and G. Ruffo. Reasoning about Accountability within Delegation. In *Proceedings of ICICS 2001:3rd International Conference on Information and Communications Security*, November 2001.
7. Li Gong. Fail-stop protocols: An approach to designing secure protocols. Technical Report SRI-CSL-94-14, S.R.I. International Computer Science Laboratory, S.R.I. International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, October 1994.
8. K. Kain, S.W. Smith, and R. Asokan. Digital signatures and electronic documents: a cautionary tale. In *Sixth IFIP Conference on Communications and Multimedia Security*, September 2002.
9. G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *Software - Concepts and Tools*, volume 17, pages 93–102, 1996.
10. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. IETF RFC 2560, June 1999. http://www.ietf.org/rfc/rfc2560.txt.
11. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the A.C.M.*, 21(12):993–999, December 1978.
12. Geraint Price. The Interaction between Fault Tolerance and Security. Technical Report 479, University of Cambridge, Computer Laboratory, December 1999.
13. M. Roe. *Cryptography and Evidence.* PhD thesis, University of Cambridge, 1997. Available from http://research.microsoft.com/users/mroe/.
14. Paul Syverson. A taxonomy of replay attacks. In *Proceedings of the Computer Security Foundations Workshop VII*, pages 187–191, 1994.
15. Paul Syverson and Catherine Meadows. A logical language for specifying cryptographic protocol requirements. In *Proceeedings of the 1993 IEEE Symposium on Research in Security and Privacy*, May 1993.
16. Paul Syverson and Catherine Meadows. Formal Requirements for Key Distribution Protocols. In Alfred De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 320–331, 1994.
17. Paul F. Syverson and Paul C. van Oorschot. A Unified Cryptographic Protocol Logic. Technical Report NRL CHACS Report 5540-227, Naval Research Laboratory, Center for High Assurance Computer Systems, 1996.

Client v. Server Side Protocols, Interfaces and Storage
(Transcript of Discussion)

Geraint Price

Royal Holloway University of London, UK

Protocols designed and analysed in the last ten years have focused on how we share keys and how we authenticate one end to the other. They convince Alice that Bob is at the other end of a line, or that Alice and Bob now share a key that is good to communicate over a particular type of channel. The output that the server, the CA, or whatever has told them is, yes, it is Bob at the other end, so it's fine to go ahead and start talking to him.

I'd like to extend some of this work onto the interface the protocol shares with the application and the data that's shared on that interface. The idea here is that the protocols that we use are not designed and used in a vacuum. There's somebody at one end of the protocol who has to use this protocol to support a particular policy within an application, to make sure that a particular piece of data conforms to some part of that policy.

So basically I'd like to extend protocol descriptions to the specification of the data on the interfaces. So much early work on protocol analysis was based on some understanding of the messages that the participants communicated with each other, and the keys that they exchanged, probably through a third party. If you're talking about an open environment where messages are possible, then you can only really assert something that's been said by another party in a secure protocol in some manner. So if you are concerned about that, then you should be equally concerned about the data they shared: you're doing something with this data, what are you going to do with it, why are you concerned about it in the first place? If I'm willing to freely communicate across the Internet without the security protocol, then if I get my fingers burnt it's my fault.

If we're going to analyse the underlying security properties that the protocol relied on then it's natural to extend this to the data that it shared across that interface. The interface interacts with certain parts of the application data that we're concerned about. So I'd like to define a distributed application interface so that we're happy with what it does, and we can manipulate it in an easier manner to serve our own goals.

One of the reasons for doing this work is that eventually it should increase ease of use of security protocols from an application point of view. For example, it has been shown that you can manipulate digital signatures on electronic documents; and the main reason they were able to manipulate digital signatures on electronic documents is that MSWord would display the document for you, and then use some PKI compliant security application in order to do the signature.

But there was very little talking between them. You assume as long as you use the PKI to sign the document then you're fine, but this is obviously not the case. So the design of the protocol interface in some manner reflects the way the protocol is used and we should be able to make it more explicit what is going on across that interface.

Say we've got a traditional three-party authentication protocol. We've got the authentication server, Alice, and Bob, and say four messages of a particular protocol. Let's say you analyse these protocols, with a BAN logic, and say, "well according to my goals this particular protocol serves my purposes and I'll go away and use it." But generally there's some application that relies on the protocol in some sense, and there is data in the application, and Alice should be wondering, following the run of the protocol, am I using this in a useful manner?

So we limit this into a communication model with some formal distributed environment, be it a client-server or peer-to-peer or third-party server. If we can demonstrate some sort of strong trust relationship between parties, it might be possible to reduce the protocol interface, and its complexity, because then we're assuming that the parties trust each other more explicitly than they may in another run of the protocol. By necessity the application interface to the protocol can then be simplified, so we can analyse these a bit more rigorously because there is less to worry about. If we could demonstrate through analysis that we have a weak trust relationship, then we need to increase what we share across this protocol interface: maybe we need to change the time stamps to nonces if we're worried about not having a secure TIME server, and so on.

So now I'm going to provide some example situations that give a feel for this work. The first one is gathering evidence; if we take the example where Alice and Bob are communicating to each other and Alice wants Bob to sign a document. Alice might be willing to accept Bob's signature at face value initially, or she might want to go and check that that key hasn't been revoked since Bob received it. Now I want to be able to demonstrate that how the data that you receive through the OCSP response matters depends on what Alice's policy is. So I'm willing to receive an OCSP response now and throw away the data in some manner, but at some point in the future I might need it, or Alice might need it, to take in front of a court if the OCSP response becomes part of a dispute later.

At the naive level, what Alice then does with the OCSP response to the CA doesn't really matter to the CA. The CA just gives Alice the message that Bob's key hasn't been revoked. In the first context, where Alice doesn't care about any particular future arbitration, then that would be OK, but in the case where Alice is really doing this to protect herself from future repudiation from Bob it really becomes an issue. So on the face of it, if you design an OCSP protocol in a particular way, it doesn't really matter from the CA's point of view, but it matters to Alice, because the data that Alice gets from this yes/no response from the CA becomes an issue depending on what Alice wants to do with the data after the particular run of the protocol.

Peter Landrock: So actually the OCSP response may not be enough?

Reply: It may not be enough, I agree. What I'm trying to get at here is, if you just gave somebody a confirmation protocol for revocation, then you could sit down and analyse it with its traditional logic and say, yes, that's secure because Alice gets a response. But what I'm trying to do is extend that issue of what does a response mean, to what does Alice need to do with that data after the run of the protocol has ended. Then you're broadening what's going on with the data from the protocol.

A second thing I might want to do is store and control data. If I can control the data and the protocol run in some manner, then I can force other parties in the protocol to act in the way I want them to. An example I gave in the past is a Kerberos-like authentication, where the service that Alice wants to connect to in the network might not explicitly trust the authentication server. I was able to demonstrate there that having some piece of data, in this case a nonce, running through the protocol can change the way the protocol interaction happened, in such a way that the service itself can be sure that the authentication server was forced to act in a certain manner. But what this also demonstrates is that these tend to complicate protocols.

So I'd love to look at it from a different point of view, and the example I'm using is XKMS, the XML key management system, where to minimise the complexity of a protocol between Alice and the PKI there's a way in which Alice can talk to a trust server and allow the trust server to do all the computation from the PKI's point of view. The protocol now has to be minimal in some sense, because the whole point of XKMS is that it's run on small machines, and you can reduce the amount of computation that Alice has to carry out herself. So now some of the state that Alice is concerned with is stored at the other end of the protocol, rather than just at Alice's end of the protocol.

So let's compare this with the previous example, where Alice had the OCSP response protocol with the CA. There Alice kept control of the signed response because she was concerned that it might be needed at a future date. In this case she's giving up control to the trust server, which she talks to through XKMS. So the protocol itself now has to be able to express Alice's security policy explicitly across the protocol interface.

A third example of something I'd like to consider is demonstrating interface compliance. Basically compliance with some sort of policy or rule, has been part of good security as long as there's been good security. One example of how this has changed in recent times is WS-Security, the new web services specification[1]. Part of this specification allows a service to force a client to receive a token before they're willing to talk to them across the interface. So an example might be that a consumer has to get a certain credit rating before the service is willing to talk to the client. The interface itself can express the policy in a manner so that no data can be shared across the interface until it has been demonstrated that the policy has been conformed with. It is now up to the client to receive the information that they want and then go and present this data across the interface as part of the protocol. Looking at it from the service point of view,

[1] See http://xml.coverpages.org/ws-security.html.

they've got to be able to be satisfied that the interface is representing their policy in their manner before they let the protocol continue.

Birgit Pfitzmann: Which interface are you talking about? Do you mean the protocol message exchanged between the parties, or the layering in the protocol?

Reply: It's the interface between the protocol and the application. Maybe I'm using WS-Security slightly out of context here, but yes, the idea is you can express this across the interface between the application and the protocol. An example is client puzzles and DoS attacks; I present you with a piece of data and unless you come back with some representation of that data in a manner that I'm happy with, I'm not willing to progress the protocol.

Birgit Pfitzmann: Is this one of the requirement specifications of the interface when you look at what messages come in from the application and what messages go out from the application, like a temporal logic specification?

Bruce Christianson: The argument is that the protocol is specified in an externally representable form, you have to comply with the message format. The applications don't always knit these themselves; usually there's something else that they're interacting with, and at each stage the application is either saying, yes go on, or no I'm not proceeding to the next step yet. Is that the interface you mean?

Reply: In this case, yes. In the example I gave with the signature and the OCSP response then, yes, it's still that because again I want to be sure that when I've finished the protocol I've got this piece of data that I can run with at a future date.

Bruce Christianson: So it's the decision to proceed? When you are satisfied of what should be true at this stage?

Reply: Yes, coming back to Simon's[2] point in a previous talk, of knowingly stopping protocols when the protocol matches your goals.

Bruce Christianson: So the level of abstraction of the specification here is absolutely crucial?

Reply: Alice might be concerned about her ability to demonstrate when this piece of data was generated, so the protocol then needs to be able to include some notion of timeliness – balances to a time stamp that I can then use to demonstrate to somebody else what that timestamp means, so they've got to agree with me at a future date when that timestamp was generated. There's no point me running the OCSP response protocol unless I can use that data to satisfy my security policy that I can demonstrate in front of a judge at some time in the future. So the data has to be able to have some notion of time unit in some sense.

[2] S. Foley, these proceedings.

Delegation is another characteristic. Can I give this piece of data to somebody else? Is my security policy allowing me to present the data across the interface so that the XKMS trust server can map and use that data to give me a valid response that I'm going to be happy with under the PKI that it is running for me?

Another characteristic of the data would be transferability where, if I get something from somebody, I want to be able to demonstrate it. I want to be able to pass that on to somebody else. If I get a token from a trust server, I want to be able to pass that token on to a service in a manner that the service is willing to accept before going on and communicating the rest of the protocol with me.

So these are the sort of characteristics that the data that is passed within the client-server, or peer-to-peer network, might be interested in.

Birgit Pfitzmann: Have you thought about getting rid of this data transferability? We don't care much about the data at the interface, we just get an output that certain things have been received, and we know it's transferred the protocol. We know it's going to give an input next, and transfer this and this, but we don't need to care about what it looks like in external form. We only know that it transferred real evidence from protocol run so and so, and back to me, and then internally that would work on the things we all worry about. I mean, it might still be implemented in a token-based way in the sense that the protocol passes it back but it would be passed back transparently to the peer application.

Reply: But the protocol's got to be able to do that in a manner so that the application should be able to specify that interface to the protocol. So if the application has this idea of the token that it wants to use for a particular input under the interface then the protocol would know how a judge would like evidence of it. The evidence has got to work so that's also part of the protocol. When the application gets the input transferred, the application wouldn't now have to worry about how much of the stuff it has to store, it would just know this is a transferable protocol.

Maybe the protocol I use because of the policy of my application has to have transferability. Coming back to the work Simon Foley has done, the point is that, when I'm generating these goals because I want my protocol to succeed, I want that particular goal to give transferability. Or, as Ross said, in the case of anonymity maybe that's not what you want. So from a point of view of liveness, aligning a certificate to a particular application, you might want to be able to switch on or off transferability of the protocol. But that then has to be a decision between the application and the protocol.

Birgit Pfitzmann: Yes but maybe you can put the entire data under the interface parameters?

Reply: I'm not too sure that's the end of it. The question I'd like to ask from these characteristics of the data is, am I going to have to do something at this stage afterwards? As I said at the start, protocols don't exactly operate in a vacuum, the application has to use them in a particular manner. So if the data needed to be demonstrated to somebody else, or if the data needs to be stored for re-use at some point, where does that originate from? In the sense of an authentication protocol, you know, that's quite a straightforward issue. If you can convince yourself that you're talking with somebody, then if the authentication server has said, yes, Bob is at the other end of the line, then I've got a fair idea that as long as I trust the authentication server, then this data did come from Bob. But maybe I have to show to somebody else where it came from. So the idea is, that what happens to the data after I've initially received it, can support what it says. Then it's straightforward. It may be my policy says check this is the revocation before I go on and do what Bob says in this signed document. Or, do I need to convince anybody else of what it said?

So, some of these questions might be answered already. A simple example is, you wouldn't use a freshness protocol with a nonce if you had to convince a third party that that protocol was fresh: I generate a nonce myself, hand it to you, you hand it back, somebody else will ask, who generated that nonce and at what point in the past? So there is a need for the transferability of the freshness of that nonce in that sense. These sorts of things are relatively well understood. Extending that onto the data shared across the interface, or other choices you make when you share information across that interface, is important.

There needs to be some class or typing system for security interactions across system interfaces, and to develop some form of taxonomy for the interfaces, a taxonomy that characterises this in a more formal manner. Hopefully that might help shape the way we build and specify the interfaces, and have some form of analysis for all kinds of taxonomy. So hopefully once I've drawn up a better list of the properties that are shared across the interface, you can analyse the particular protocols that meet your requirements.

We need a greater understanding of the relationship between the protocols and the data that's shared across the protocols. If you use a protocol to do something, OCSP response for example, then you want to know what you're doing with that signed message afterwards. Do you understand how that interface affects the way the application talks to the protocol, and what happens to that data? In terms of transferability across networks of particular data, and in symmetric vs. asymmetric cryptography, interfaces play a big role in what's going on.

Guaranteeing Access in Spite of Distributed Service-Flooding Attacks

Virgil D. Gligor*

VDG Inc., 6009 Brookside Drive, Chevy Chase, Maryland 20815
gligor@umd.edu

Abstract. We argue that open networks designed using end-to-end arguments are particularly vulnerable to flooding, and that this vulnerability persists as hardware and operating systems technologies advance. An effective end-to-end approach to counter distributed flooding attacks against public services and provide access guarantees to their clients is to establish and enforce *"user agreements"* among clients outside the public services they access. Among the user agreements designed to protect servers from flooding attacks, those requiring client proofs of work (e.g., client puzzles using hash functions) are both ineffective and unnecessary whenever strong access guarantees are desired. In contrast, simple rate-control agreements can be defined to provide strong guarantees based on waiting-time limits. These agreements are established by special-purpose servers and verified before request processing at network-line rate, and hence cannot be flooded.

1 Introduction

Imagine that all distributed denial of service (DDoS) attacks at and below the transport layer of an open network, such as the Internet, are handled, and that all protocol and server flaws/features that enable DDoS attacks at the application layers are removed. In this ideal setting, flooding attacks against the publicly accessible services remain a significant and persistent concern. Vulnerability to flooding arises because request rates of network access points (e.g., the peak network "line" rate) exceed the throughput (e.g., the peak request rate) of publicly accessible application servers to which they are attached, often by more than an order of magnitude. For example, analysis of Internet flooding events shows that peak line rates can exceed 600K packets per second [19], yet specialized firewalls designed to protect servers from TCP SYN flood attacks can be disabled by line rates as low as 14K packets per second [5]. Simulations show that even a highly replicated content-distribution service (i.e., 64 servers) designed to withstand flooding attacks reaches less than 40K requests per second, with the average request of 6KB [22] fitting in a single packet. Hence, clients' aggregate request rates to application servers could legitimately reach much higher levels than application servers could tolerate and yet *not* trigger any network-layer alarms. For

* Permanent address: Department of Electrical and Computer Engineering, University of Maryland, College Park, Maryland, 20742.

example, during the much-publicized flooding attacks of February 2000 against public services of Yahoo!, Ebay, and E*trade, and against Microsoft's name servers in January 2001, Internet service providers (ISPs) did not notice any unusual network traffic in most attack instances. This clearly shows that flooding attacks are an *end-to-end problem* that requires *end-to-end solutions* no matter how effective IP-layer flooding countermeasures might be.

In this paper, we argue that open networks designed using end-to-end arguments [21] are particularly vulnerable to flooding, and that this vulnerability persists as hardware and operating systems technologies advance. We present an end-to-end approach aimed at providing access guarantees to clients, in spite of an adversary's attempts to flood public servers with requests from rogue programs running on adversary-controlled client computers. We review several types of access guarantees based on waiting-time limits (e.g., maximum [8,9], finite [25], and probabilistic [18] waiting-time limits), explain their relationships, and show how to achieve some of the strongest of these based on the notion of *"user agreements"* [25]. Simple agreements are generated and enforced on dedicated special-purpose servers that cannot be flooded as they are designed to operate at the peak line rates reached at their network-access points. We strengthen these guarantees by controlling client proliferation on adversary-captured machines with either reverse Turing tests (e.g., CAPTCHAs) [24,1] or with remote verification of human-activated, hardware-implemented, trusted path on TCPA-equipped client machines [20].

Although user agreements are necessary for preventing end-to-end flooding attacks, some agreements cannot offer strong client guarantees. We argue that agreements requiring client proofs of work (i.e., client puzzles using hash functions) are both ineffective and unnecessary. They impose a high overhead on legitimate client requests and only offer very weak guarantees, yet other user agreements exist that provide same guarantees, or stronger, at the same or lower request overhead. We illustrate this observation with simple examples.

Our approach to countering distributed flooding attacks does not require user registration or state maintenance. However, it does rely on routers to perform the typical *ingress filtering* at the network edges already required for other reasons (e.g., prevent IP address spoofing by virus propagation mechanisms). We do not aim to handle flooding attacks within network layers and assume that these are handled by separate protocols [14].

2 End-to-End Flooding and Countermeasures

We therefore can't just jump from the observation that a resource is held "in common" to the conclusion that "freedom in a commons brings ruin to all."
Lawrence Lessig [17]

Open networks offer public services for various applications and, at the same time, rely on public services themselves for infrastructure functions, such as security (e.g., user registration, authentication), naming (e.g., DNS), and user-level communications (e.g., mail servers). Thus, in principle, all open networks are vulnerable to flooding. However, open networks that are designed using the end-to-end argument, such as the Internet, are particularly vulnerable. The persistence, if not the existence, of the gap between network-line and application-server rates is a consequence of applying the end-to-end argument to network design and, if anything, the gap will grow larger in the future.

The end-to-end argument suggests that network computers (e.g., routers) perform very simple functions that are common to all applications, while complex functions required by fewer applications be implemented in end-servers. This is arguably the single most important technical idea for assuring network performance on large scale since it makes the "common case" fast (viz., Amdahl's law [13]). It is also the single most important reason for the gap between the network-line rates and application-server rates: common, simple network functions are made fast whereas less common, complex functions that would slow down the network are pushed to end-servers. The intended consequence is that network-line rates are at least as high as the request rates of the vast majority of the end-servers. The unintended consequence is that the rate gap becomes pervasive and enables flooding attacks that cannot be detected and handled within the network.

Persistent Vulnerability. Simplicity of network functions translates ever-increasing hardware performance into higher network-line rates almost directly. In contrast, similar performance increases are quickly offset by increased complexity of new end-server applications and operating system features.[1] Thus, there is no reason to anticipate that future technologies will close the rate gap. On the contrary, there is reason to believe that the rate gap will grow larger in the future. The typical rate gap of the early ARPA network was smaller than that witnessed in the Internet today.

End-to-end arguments also rule out the closing of the rate gap artificially by slowing network-access points to match application-server rates. This would only shift the flooding vulnerability from servers to network-access points where there is less freedom, and thus harder, to remove it. Attempts to push back traffic along the multiple routes terminating at an access point of the Internet would likely encounter ISPs that could justifiably refuse to allow external pushbacks into their networks for security or policy reasons (e.g., the external pushback request may appear to be a disguised attempt to sabotage their local network traffic by a competitor, insufficient selectivity of, or justification for, the pushback request) [14]. Further, the alternative of closing the rate gap globally in an adaptive manner would be ill-advised, as it would require solving a complex optimization problem continuously on a global scale – thereby penalizing the performance of

[1] Complexity of applications is not the only reason why cycles of improved hardware performance vanish rapidly in end-servers. Lampson's explanation [16] of where orders of magnitude in improved hardware performance went in PCs also applies here; i.e., hardware resources traded for quick time to market by software developers, lots of features delivered quickly lacking first-class design, integration of different features, compatibility with old hardware and systems, and faster response to user's actions.

common functions – for the purpose of eliminating a relatively infrequent event [19] that could be handled, whenever required, end-to-end.

Attacks. The rate gap is a cause of undesirable dependencies among clients of shared services. Since all shared services have finite resources (e.g., finite request queue), whether a client's request receives a (timely) response depends on other clients' behavior in using that service. However, the rate gap allows a group of otherwise-legitimate client programs running on a large number of adversary-controlled hosts to prevent other legitimate clients from accessing an application server for a long time, and thus service is being denied [8,9]. The detrimental nature of undesirable dependencies to service availability has a direct analog in economics, and thus is not a mere artifact of technology. A public service is a "rivalrous" resource since its use by a client competes with that of others; and, if use of a rivalrous resource is open to all, the resource may be depleted by the consumption of all.[2][17]. This is known as *the tragedy of commons* (i.e., "freedom in a commons brings ruin to all" [12]).

End-to-End Countermeasures. In computing, the remedy proposed to counter undesirable service-access dependencies, and hence to avert the tragedy of commons, has been the establishment and enforcement of *"user agreements"* among clients outside services they share [25,18]. User agreements are constraints placed on the behavior of service clients to counter undesirable dependencies and help guarantee client access. For example, constraints could require that clients solve a cryptographic puzzle before server access, which would act as a pricing mechanism for server access and could prevent flooding; e.g., "junk mail" flooding [7]. This idea has been used to counter connection-depletion attacks [15,23] and flooding of authentication servers [2], such as that of TLS [6]. The role of user agreements in public service access seems to have also been observed by economists in many different contexts and for much the same reason; i.e., communities regulate over-consumption of rivalrous resources by establishing *"norms" of user behavior* [4]. Hence, independent motivation for user agreements indicates that they can lead to sound end-to-end solutions to flooding problems.

3 Client Guarantees and Attempts to Provide Them

We cannot enter into alliance with neighboring princes unless we are acquainted with their designs.
Sun Tzu, *The Art of War*

General questions regarding user agreements include these: (1) given a specification of request scheduling in a public service, what is the weakest user agreement that supports a desired guarantee of client access?, and conversely (2) if the scheduling specification could be changed, what change would assure a desired

[2] The fact that a service or resource is publicly accessible, or open to all, does *not* imply that it is incorrectly priced. It only means that the pricing model is different from the access model.

guarantee for a given user agreement? Answers to both these questions require that client guarantees be defined.

Access guarantees that specify upper bounds on waiting times are intuitively appealing because they help establish performance bounds for network applications. A wide variety of such guarantees can be defined according to their (1) scope (e.g., per request, per service), (2) type of upper bounds (e.g., maximum, finite, and probabilistic waiting time), and (3) quality of these upper bounds (e.g., constant, variable, variable-dependent or -independent of the attack). Guarantees that do not necessarily specify upper bounds of waiting times, such as those based on average waiting times, are also helpful particularly when they are used to refine the upper-bound guarantees. For example, a strong waiting-time guarantee may have a constant upper bound and yet have an average waiting time that is higher than that of a weaker guarantee that has only a variable upper bound.

Guarantees that specify bounded waiting time on a per-request basis are of practical interest for open networks. Per-service guarantees of similar type and bound quality are stronger, but some may be impractical for open networks. For example, per-service maximum waiting-time guarantees with tight constant bounds, which are useful in real-time systems, typically require special-purpose networks with limited client populations. In contrast, guarantees that do not specify bounded waiting times can be very weak. For example, a guarantee which specifies that a "server will not crash in a flooding attack" is implied by all waiting-time guarantees, and is extremely weak. Clients' requests may be denied access to a service in the absence of server crashes, and the majority of requests that are accepted may be those of adversaries' clients.

3.1 Definitions

The principal parameters used in the per-request, waiting-time guarantees are: (1) S = a server's maximum request rate; (2) L = maximum number of entries in a server's request queue; (3) τ = length of time a queued request has to wait until the server processes it, and thus $\tau \leq L/S$; (4) Δ = propagation delay incurred by a client request to reach a server, and is bounded by our assumption that the network layer is denial-of-service free; (5) T = waiting-time bound specified at the time of the request, and thus $T \geq \Delta + \tau$; (6) N = the maximum network-line rate, and $N - S > 0$ is the rate gap. Using these parameters, we specify the following types of *per-request*, waiting-time guarantees, of which the first three have been defined in prior work [8,25,18]:

1. Maximum Waiting Time (MWT). A client's request is accepted for service in time T.

2. Finite Waiting Time (FWT). A client's request is accepted for service *eventually.*

In practice, this specification means that there is a finite number of clients whose request can be accepted for before any other client's request. Thus, if a client has to retry its request, then the number of retries is finite.

3. Probabilistic Waiting Time (PWT). The probability that a client's request is accepted for service in time T is not less than ρ, where $\rho \neq 0$ is independent of the flooding attack.

The attack-independence of the lower bound ρ means that the expression for ρ does not include any parameters of, or dependent on, the adversary's attack; e.g., the number of adversary's clients launching the attack, their individual and aggregate request rates, the number of request retries forced by the adversary on a legitimate client's request. Nevertheless, ρ may be a variable that depends on network parameters, such as those of the rate gap, S and N. PWT guarantees can be strengthened by asking that ρ be a constant, or even one [18]. It can also be weakened, by allowing a different non-zero lower bound p that may depend on flooding attack parameters.

4. Weak Probabilistic Waiting Time (wPWT). The probability that a client's request is accepted for service *eventually* is not less than p, where $p \neq 0$.

Although $wPWT$ is a very weak guarantee, it is not the weakest. The weakest non-trivial guarantee, denoted by $WPWT$ below, is one in which the lower bound p of $wPWT$ could be zero. This guarantee is meaningful and not unusual. It refers to cases when an adversary can deny service to some class of client requests but not others. It means that "some client requests will be accepted" by the server, and assures that during the "up periods" of a server, at least one client request completes service.

With appropriate experiments defined, one can obtain slightly weaker v (i.e., probabilistic) variants of the MWT and FWT guarantees that help establish the following relationships between the above guarantees:

$MWT \Rightarrow PWT \Rightarrow wPWT \Rightarrow WPWT$,
$MWT \Rightarrow FWT \Rightarrow wPWT \Rightarrow WPWT$, and
$FWT \not\Leftrightarrow PWT$,

where the relationship "$A \Rightarrow B$" means that if guarantee A is satisfied so is B. Using the relationship " $\Rightarrow$ " one can extend the simple lattice above with the new top being the per-service MWT with the smallest practical T.

A variety of other guarantees beyond those considered in this section may be (more) useful. As in the case with *most* results based on bounds, care must be exercised in determining the "strength of guarantees." The intent of the partial order above is to illustrate one such notion of strength.

3.2 Guarantees of Client Puzzles Based on Hash Functions

Client puzzles require that each client solves a challenge puzzle as proof of work along with each request to a server. Only after receiving the proof of work the server decides whether to consider a clients' request for processing. The level of strength k of a challenge puzzle is either determined by the server or chosen by the clients themselves. Servers that schedule requests on the basis of the strength of the puzzles solved – by preempting queued requests with lower puzzle levels when a new request arrives with a solved higher-level puzzle – give clients and incentive to select increasingly stronger puzzles in a bid to obtain service ahead of other clients (viz., the "puzzle auctions" [23]). The server scheduler checks all puzzle solutions supplied by clients' requests, at the network-line rate. All client's requests that solved the server-specified puzzle incorrectly, or not at all, are dropped. If the clients' aggregate request rate is still high at puzzle level k, the server drops the extra requests and either expects the clients whose requests are dropped to bid higher-strength puzzles or specifies a higher-strength challenge puzzle, for example $k+1$, itself.

Typical puzzles use cryptographic hash functions, which assures that the cost of a puzzle solution to a client is exponential in k, where k is much smaller than the length of the hash function output; i.e., the range of k is between 1 and 64 bits whereas the hash function output is between 128 and 160 bits. Finding a hash function output that has k consecutive zeros in the high-order bits is a typical client puzzle [2], and attempts to solve such puzzles can be viewed as Bernoulli experiments that have probability 2^{-k} of success [23].

An advantage of user agreements based on hash-function puzzles is that they can be *stateless* [2]. Additional properties of client puzzles using hash functions are defined below in terms of the guarantee parameters defined in the previous section, and (1) $N_Z^{k_r}$ = the aggregate request rate at which Z clients solve puzzles of strength k_r; (2) s = the rate at which clients can execute hash function operations, (3) t_L^r (t_Z^r) = the time necessary for Z clients to solve at least L (Z) puzzle at level k_r; and (4) c_r = additional time buffer or, equivalently, space Sc_r, beyond the server's processing time, τ, and queuing, $S\tau$, needs.

Time Buffer. The additional time buffer, c_r, includes three components. The first is $t_Z^r - t_L^r$. The second, denoted by c^N, accounts for any possible randomization of request routes from clients and servers in the network, and for other random delays that might be introduced by the network layer to help slow the coordination of an adversary's clients. The third, $L/S - \tau$, represents the buffer provided by additional storage for requests explicitly provided by the server. Thus the total additional time buffer $c_r = t_Z^r - t_L^r + c^N + L/S - \tau$. The component $c^N + L/S - \tau > 0$ enables the use of lower puzzle levels than otherwise might be necessary to control the aggregate request rate and avoid flooding.

Proposition 1 (Solution Latency for Level-k_r Puzzles):
With high probability,
a) $Z \geq 2L + 2\sqrt{6L+9} + 6$ clients solve at least L puzzles at level k_r in $2^{k_r-1}Z$ steps and time $t_L^r = 2^{k_r-1}/s$; and
b) Z clients solve at least Z puzzles with level k_r in $2^{k_r+1}Z$ steps and time $t_Z^r = 2^{k_r+1}/s$.

Proof:
a) We find the bound of the desired probability as follows:
$Pr[$ Z clients solve at least L puzzles at level k_r in $2^{k_r-1}Z$ steps$]$
$= 1 - Pr[$ Z clients solve fewer than L puzzles at level k_r in $2^{k_r-1}Z$ steps$]$
$\geq 1 - Pr[$ Z clients solve at most L puzzles at level k_r in $2^{k_r-1}Z$ steps$]$
However, $Pr[$ Z clients solve at most L puzzles at level k_r in $2^{k_r-1}Z$ steps$]$
$\leq exp(-x)$, where $x = \frac{(Z/2-L)^2}{Z}$. The proof of this fact is omitted as it is

similar to that of Proposition 2 in [23]. It is obtained using the variant of the *Chernoff/Hoeffding Bound* [11], which states that $Pr[E[X] - X \geq \epsilon E[X]] \leq exp(-\epsilon^2 E[X]/2)$, where $0 < \epsilon < 1$, by requiring that $0 < \epsilon \leq 1 - 2L/Z$. Thus, $Pr[$ Z clients solve at least L puzzles at level k_r in $2^{k_r-1}Z$ steps$] \geq 1 - exp(-x)$. $Z \geq 2L + 2\sqrt{6L+9} + 6$ implies that $x \geq 3$, which means that the lower bound of this probability is at least 0.95.

b) This follows along the same lines as part (a).
$Pr[$ Z clients solve at least Z puzzles at level k_r in $2^{k_r+1}Z$ steps$]$
$= 1 - Pr[$ Z clients solve fewer than Z puzzles at level k_r in $2^{k_r+1}Z$ steps$]$
$= 1 - Pr[$ Z clients solve at most $Z - 1$ puzzles at level k_r in $2^{k_r+1}Z$ steps$]$
However, $Pr[$ Z clients solve at most $Z - 1$ puzzles at level k_r in $2^{k_r+1}Z$ steps$]$
$\leq exp(-y)$, where $y = \frac{(Z+1)^2}{4Z}$. Thus,
$Pr[$ Z clients solve at least Z puzzles at level k_r in $2^{k_r+1}Z$ steps$] \geq 1 - exp(-y)$.

Proposition 2 (Request-Rate Control): Let the time buffer c_r be such that $3\frac{2^{k_r-1}}{s} \leq c_r < Z/S$. Then with high probability, $N_Z^{k_r} \leq S \Leftrightarrow k_r \geq 1 + \lceil \log(\frac{Z}{S} - c_r)s \rceil$ in a time interval of length $t_L^r + c_r$.

Proof:
By Proposition 1, with high probability, Z clients solve at least Z puzzles in time $t_L^r + c_r$, where $c_r \geq t_Z^r - t_L^r \geq 3\frac{2^{k_r-1}}{s}$. Also, by definition, $N_Z^{k_r} \geq Z/(t_L^r + c_r) = Z/(2^{k_r-1}/s + c_r)$. Thus, $S \geq N_Z^{k_r} \geq Z/(2^{k_r-1}/s + c_r) \Leftrightarrow k_r \geq 1 + \lceil \log(\frac{Z}{S} - c_r)s \rceil$.

Proposition 1 gives a lower bound for the additional buffer time c_r whereas Proposition 2 gives an upper bound for c_r and a lower bound for k_r. With high probability, $c_r < Z/S$ for *any* puzzle scheme based on hash functions. This bound holds for all puzzle levels up to and including k_r, at which point $N_Z^{k_r} \leq S$. Were $c_r \geq Z/S$ for some $i \leq r$, then client puzzles would become unnecessary at level k_r since the aggregate request rate would already be at or below the server rate; i.e., $N_Z^{k_r} \leq S$.

WPWT Guarantees. Proposition 2 shows that user agreements based on client puzzles using cryptographic hash functions can provide the weakest guarantees, even when extra time buffer or storage is unavailable (i.e., when c_r reaches its lower bound). If a server that keeps dropping and re-challenging clients' requests with higher-difficulty puzzles, or if the clients' bid for access with increasingly higher-difficulty puzzles, the server will not crash and thus *some* clients' requests will be serviced. This seems to have been the goal of most prior work in this area [15,2,6].

wPWT Guarantees. Wang and Reiter [23] provide an interesting model for analyzing "puzzle auctions" and show that an upper bound on the probability of denied access at the r-th retry, denoted by $Pr[\xi_r]$ below, can be found if one makes an approximation assumption on the size of L. The bound is:

$$Pr[\xi_r] < (1 - 2^{-k_0})^{\frac{2^{k_0-1}L}{Z} - s\tau} \prod_{i=1}^{r} (1 - 2^{-k_i})^{\frac{(2^{k_i-1} - 2^{k_{i-1}-1})L}{Z}} \neq 0,$$

where $k_0, \cdots, k_r$ is a sequence of puzzle levels a client would have to solve and bid to get access to a server in $r+1$ rounds (i.e., r retries after the first attempt). A *wPWT* guarantee is provided, but not a *PWT* guarantee. That is, for any number of retries R that is fixed at the time of the request,

$Pr[$ any client C's request is accepted in time T $] \geq$
$1 - Pr[$ any client C's request is denied after R retries $]$

is dependent on the attack (i.e., on the number of clients Z controlled by the adversary).

The approximation assumption says that if L *is long enough*, the adversary's probability of solving L puzzles at level k within $2^{k-1}L$ steps can be ignored. How long is long enough ? The variant of the Chernoff/Hoeffding bound used in Wang and Reiter's Proposition 2, requires that L has a strict lower bound in terms of the number of adversary's clients, Z, namely $L > Z/2$.

3.3 Limitations of Client Puzzles Based on Hash Functions

In this section we argue that user agreements based on client puzzles implemented with hash functions are ineffective in the role of user agreements for preventing DDoS attacks, as they combine weak service-access guarantees with high request overhead.

Attack Coordination. Proposition 2 provides a lower bound for the puzzle level k_r such that $N_Z^{k_i} \leq S$. Hence, for all $k_i \geq k_r, i \geq r$, an adversary has little or no chance to deny access to *all* legitimate clients. However, Proposition 1 implies that, after solving at least L puzzles (i.e., after t_L^i), the adversary can coordinate its clients to issue their requests at rate $N_Z^{k_i}$ over an interval of length δ at every puzzle level, where $L/N_Z^{k_i} < \delta < Z/S$, such that $N_Z^{k_i} > S$, even when $L > \frac{Z}{2}$. The goal of such an attack would be to ensure that in these intervals *at least* one client's request and retries will be denied access at successive puzzle levels k_i repeatedly with non-zero probability. These attacks would deny strong access guarantees, and yet would be undetectable at the network layer, since $N_Z^{k_i} \leq N$.

Weak Guarantees. When a coordinated attack is in progress, (1) the ability to distinguish among client requests based on puzzle levels is lost, since all requests have the same level k_i, and (2) a legitimate client's best chance of access is at the highest puzzle level $k_m \geq k_r$ in an interval of length $\frac{2^{k_m+1}}{s} - \delta$. Let $p_m = \max(p_i), i = r, \cdots, m$ denote the probability of client access in this interval, for any request scheduling discipline based on puzzle levels. By assuming that a client has this chance at all puzzle levels k_i, $k_r \leq k_i < k_m$, we obtain an upper bound on the probability of a client's access within m retries, denoted by $Pr[\gamma_m]$, as follows.

$$Pr[\gamma_m] < p_m + (1-p_m)p_m + (1-p_m)^2 p_m + \cdots + (1-p_m)^{m-r}$$
$$= p_m \sum_{i=0}^{m-r} (1-p_m)^i = 1 - (1-p_m)^{m-r+1} < 1, \text{ for any finite } m \geq r > 0.$$

This indicates that a *FWT* guarantee cannot be supported. In a similar manner, one can obtain a non-zero lower bound for the probability of client access that would also depend on attack parameters – just as in the example of puzzle auctions above – indicating that a PWT guarantee cannot be supported.

Attempts to counter coordinated attacks by adding more constraints on the number of adversary's clients, Z, and $N_Z^{k_i}$ would be impractical. Both parameters may change dynamically and may not have a useful upper bound. In fact, any bound on Z, and hence on $N_Z^{k_i} < N$, would be impractical in open networks. Since end-to-end solutions cannot distinguish legitimate requests from those of an adversary's clients, a bound on the *total number* of clients would have to be imposed. This would be incongruent with the very notions of open networks and public services. Further, if different scheduling policies, independent of puzzle levels, would prove effective for providing PWT and FWT guarantees, they would be effective stand-alone, and would not need client puzzles, as illustrated in the next section.

High Request Overhead. As an example of request overhead, we use the five-bid sequence $k_1 = 15, \cdots, k_5 = 19$ of a puzzle auction assuming that that the rate of hash function operations is $s = 10^6$/second [23]. Assuming a round-trip message delay of 140 milliseconds, which is typical in the Internet, this five-bid sequence takes 1.72 seconds. However, fairly typical Web servers have a response time to client requests ranging from 1 second down to to 100 ms. Thus, the overhead of this sequence would range from over 170% to over $1,000\%$, for every client request, legitimate or not. This overhead would only be cut in half if clients start bidding at the top of the sequence, namely at $k_5 = 19$. The above sequence would also impose much higher overhead on specialized public servers of the network infrastructure that are designed to operate a higher speeds. For instance, the response time of a TLS/SSL server under load is only about 16 ms [6]. The request overhead of the five-bid puzzle sequence would exceed $1,000\%$ even if we could increase the rate of hash function operations five fold. Some clients, such as those running on low-power devices, would not be able to tolerate such overhead.

3.4 Simple User-Agreement and Service-Scheduling Specifications

In this section we give *very simple* user agreements and service scheduling specifications that can provide $wPWT$ guarantees at the same request overhead as, or lower than, that of puzzle auctions. We also give a very simple user agreement that provides a PWT guarantee, although its lower bound is very low. In both examples, the probability bounds are independent of the number of adversary's clients, Z.

User Agreement Specification. Suppose that instead of solving any puzzle, a client simply agrees to resubmit a dropped request or retry. This is the weakest possible agreement in the sense that it does not place any constraint on clients' behavior. However, the adversary's desire to keep the aggregate rate of all its clients be under N – to avoid tripping network-layer alarms – imposes implicit self-constraints for both the number of clients and their request/retry rates.

1. Service Scheduling Specification: Random L requests without preemption. Suppose that the service scheduler buffers up to $N\tau > L$ requests, picks up to $L = S\tau$

of them at random, places them in a L-entry queue, and drops the rest. This is the weakest scheduling strategy without preemption in the sense that is does not depend on any request parameter. For these specifications, we obtain the following bound:

Pr[any client C's request is accepted *eventually*] $\geq$
Pr[any client C's request is accepted after r retries]
$= 1 - Pr$[any client C's request is denied after r retries] $\geq 1 - [1 - \frac{S}{N}]^{1+r} \neq 0$

This bound depends only on gap rate parameters S and N and on the number of retries r. Just as with the client-puzzle bound when $p_m = S/N$, it approaches 1, but only asymptotically, as an attack forces $r \rightarrow \infty$. Hence, the guarantee provided is still only $wPWT$, instead of FWT.

Relative Cost. The cost of this user-agreement scheme relative to that of puzzle auctions depends both on the round-trip message delays and on the size of the rate gap, S/N. For instance, for a round-trip message delay of 140 milliseconds, and for a S/N ratio that can be as low as 1/8, this type of user agreement offers about the same guarantees as the five-bid puzzle auctions above for about the same cost. That is, for $r + 1 = 12$ the probability of *denied* access is 0.2248, and the total cost is 1.68 seconds. For the same probability of denied access and $L = Z = 1024$, $\tau = 250$ microseconds, the cost of the five-bid puzzle sequence $k_1 = 15, \cdots, k_5 = 19$, is 1.72 seconds. On the other hand, if the rate-gap ratio S/N is somewhat higher but still a very low 1/5, then for the same probability of denied access, the number of round trips drops to $r + 1 = 7$, and the total cost would be only 0.98 seconds. In contrast, even if we double the clients' speed, s, the cost of the five-bid sequence would be higher, namely 1.2 seconds. Finally, if a message round-trip time decreases to 15 milliseconds, which would not be unusual in a 50 Mbps network, the cost per request for $S/N = 1/8$ and $S/N = 1/5$ would be 0.18 and 0.10 seconds, respectively, yet the cost of the five-bid sequence would only drop to 0.5 seconds even when s doubles.

2. Service Scheduling Specification: Random L requests with preemption.[3] If request buffering is unavailable, the server schedules any new request as follows. If the server's L-entry queue is not full, the server queues the new client's request. Otherwise, it picks a uniformly distributed random number in the interval $[0, L]$. If the picked number is zero, the server drops the new request and asks the client to retry. If not, it frees a queue slot by dropping the client's request already placed in the scheduler queue's at the entry indexed by the (non-zero) random number (i.e., it preempts a request), and asks that client to retry. Then, it places the new client's request in the queue slot just freed. For this type of user agreements we have a PWT guarantee. That is

Pr[any client C's request is accepted for service in time T $\geq \Delta + \tau$]
$\geq Pr$[any client C's request is queued in time Δ]

[3] Courtesy of Vijay G. Bharadwaj.

$\times\ Pr[$ client C's request is not preempted by n new requests in time $\tau]$
$= [1 - \frac{1}{L+1}][\frac{1}{L+1} + \frac{L-1}{L+1}]^n = [\frac{L}{L+1}]^{1+n} \geq [\frac{S\tau}{S\tau+1}]^{1+N\tau} \neq 0$

which depends only on the rate-gap parameters S, N and thus is independent of any attack parameter. Hence, a PWT guarantee can be satisfied, although the non-zero lower bound is very small; e.g., much smaller than the bound of scheduling random L requests without preemption above.

We note that the scheduling of random L requests with preemption could be added to puzzle auctions (1) to remove the approximation assumption, and (2) to support a *PWT* guarantee. The high per-request overhead of client puzzles would still remain a significant concern.

4 Simple Rate-Control Agreements for MWT Guarantees: An Example

In our approach, a *rate-control service (RCS)* is associated with each application service. RCS ensures that the aggregate rate of all clients, both legitimate and adversary's, does not exceed the maximum rate of the application service L/τ in any time interval τ or larger. To accomplish this, RCS uses an access reservation policy whereby each client is allowed one or more server accesses in a reserved time window. (Of course, other rate-control policies can be used to accomplished this goal [3].) The user agreement requires that each client obtains an access ticket from RCS and places its server request within the time window specified by RCS in the ticket. For services that require multiple accesses within a well-defined protocol, such as those for authentication, naming, and mail, RCS authorizes multiple accesses with a single time window. However, the number of accesses may be lower than that required by specific protocols to prevent degradation of service performance by the adversary's clients obtaining but not using request tickets – a potential hazard with all time-window reservation policies. For services that are accessed at the client's discretion, RCS's default is a single access within a time window. In either case, per-request MWT guarantees are provided since each client request can count on accessing the server within the upper limit of the time window.

Each client request to a server must include the ticket authorizing access to the server. Ticket validity is checked by a *Verifier* located between the network access point and the application server. The Verifier checks that a client's request is authorized in the current time window, and if not, it drops the request. The Verifier and RCS are time synchronized within tolerances comparable to network delays, and share a symmetric key. RCS uses the key to generate a Message Authentication Code (MAC) for each ticket issued to a client, and the Verifier uses the key to check the authenticity of each client's request ticket. MAC generation and verification are the most time consuming operations RCS and the Verifier perform. However, since request tickets are short (e.g., under 1 KB) and these operations can be executed fully in parallel [10], RCS and the Verifier can execute their operations at rates that far exceed the network line rate, and hence cannot be flooded.

Request Tickets. A client request for an access ticket includes the number of accesses desired, the source IP address from which the requests will be issued, the start time of the window in which the request will be issued, t_s, and, if a multiple-access ticket is desired, the maximum interval between two client requests for that application, δ_r. RCS verifies that the requested IP address is the source address of the requester's message, that the number of requested accesses and δ_r are consistent with the server-access protocol, and that t_s is within the ticket postdating limit allowed. If these checks pass, RCS issues a ticket and its MAC. The ticket contains (1) a start time, t_i; (2) an end time, t_{i+1}; (3) a count of the maximum number of accesses, w, where $1 \leq w \leq L/n$ and $n \geq 1$ is the number of clients that can access the server within the same window; (4) the source IP address for the request; and (5) the time of ticket issue, t_{RCS}. The window start time is set to $t_i = t_w + \Delta$, where $t_w > t_s$ is the first available time window when this client can issue its first request. The RCS time, t_{RCS}, allows the client to synchronize its request-issue time with the Verifier, Δ ensures that the ticket is valid upon client receipt, and t_s that the client has time to issue the request. The window end time is set to $t_{i+1} = t_i + w(\tau + 2\Delta + \delta_r)$. This allows for the network delay Δ before the request reaches the Verifier, for request processing in τ units of time after receipt by the Verifier, and for ticket validity before the next access, $\Delta + \delta_r$. For tickets authorizing multiple accesses, the Verifier maintains a cache of tickets seen within the current time window and the number of accesses already exercised within that window for each ticket, to prevent ticket reuse by clients beyond w times. Since the time windows are relatively small, the ticket cache will be held in primary memory and much, if not all, of it will fit in the processor cache.

Ticket validity checks by the Verifier require that requests must be placed within the allocated time window, must not exceed the allowed access count, w, must be issued from the IP address for which the ticket was issued, and must pass the MAC check. All these checks can be performed at a faster rate than that of all network access points. To prevent ticket theft and/or reuse from spoofed IP addresses, typical *ingress filtering* at network edges already required for other reasons (e.g., to prevent IP address spoofing by virus and worm propagation) must be performed at all clients' access points to the network.

Transparent Operation. The Verifier monitors the aggregate rate of clients' requests starting with the first possible client request (e.g., the client-server binding request). Whenever this rate exceeds a threshold, the Verifier enters the rate-control mode by directing each client to RCS. In rate-control mode, each client request must present a valid RCS-issued ticket. When rate-control mode can be exited, the Verifier returns an exit notification to the client along with the server's response. The client (i.e., the proxy code of the server running on the client machine) implements client-RCS and the client-Verifier request-response protocols, and thus these protocols are completely transparent to the client and server application code. However, an additional API is implemented by the client proxy so that the application can test whether the server operates in rate-control mode and retrieve the MWT. Hence, the client application can execute other concurrent threads while waiting for a server response.

Session Cookies. Although RCS and the Verifier can ensure that the aggregate request rate of all clients, legitimate or not, stays below the server's rate, an adversary can start a large number of clients on different machines, obtain valid tickets, and exercise them in an authorized manner. This would cause legitimate users' ticket start time and end times to be pushed into the future increasing the MWT for legitimate clients, possibly beyond reasonable values. To prevent uncontrolled client proliferation by an adversary and large MWT values, we require that a client request for a ticket to RCS be accompanied by a cryptographic cookie attesting that the client had a human user behind it. The cookie can be obtained by the client in two ways: (1) after passing a reverse Turing test (or CAPTCHA [24,1]), or (2) after verification of human-activated, hardware-implemented, trusted path on a TCPA-equipped client machine [20] that is registered with designated Internet servers. A cookie has a similar structure as a ticket, and includes: (1) a start time, (2) an end time, (3) a list of IP addresses from which ticket requests can be issued, (4) the time at the RCS, which issues the cookie, and (5) the MAC for the cookie. The time window in which the cookie is valid is comparable to that of a login session, and consequently the reverse Turing test or the verification of human-activated trusted path is required only once, upon session start by a human user. Also the cookie allows a limited number of clients to operate on behalf of the same human user from different IP addresses. Ingress filtering would also ensure that cookies cannot be passed around and used from addresses not included in the IP address list of the cookies.

Performance Considerations. The request overhead of this scheme is significantly lower than that of client puzzles. For instance, assuming that the typical round trip network delay is 140 milliseconds, the overhead for a request that otherwise would take one second is 14% instead of over 170% for the five-bid sequence of the puzzle auction mentioned in the previous section. The per-request overhead of a protocol requiring two to ten accesses each taking only 100 milliseconds would be between 14% and 70% when using a multi-access ticket, as opposed to over 1000% in the five-bid sequence of puzzle auctions.

For all applications, the question of how many accesses should be specified in a ticket arises naturally. Typical services that implement infrastructure protocols require between two and ten client accesses per application. In contrast, typical client server accesses for content distribution networks range from about six to sixty per application. If a single access is allowed, the communication cost for the client increases due to multiple RCS round trips for tickets. In contrast, if all accesses are included in single ticket windows, unused tickets by adversary's clients could decrease server utilization significantly due to reserved but unused windows. Hence, an optimal window size could be determined, at least theoretically, that trades off request overhead against server under-utilization.

Let c_1 be the unit cost of a round trip to RCS for ticket retrieval, c_2 be the unit cost of lost application server utilization caused by requests not issued by adversary's clients in a reserved time window, A_r be the access count per application, and l be the percentage of legitimate clients in the system, $0 \leq l < 1$. The optimal access count per window can be computed as a simple minimization of the total cost $C_{total} = C_{client} + C_{server} = c_1 A_r/w + c_2(1 - l)w$. Setting $\frac{\delta C_{total}}{\delta w} = 0$, we obtain $w_{opt} = \sqrt{\frac{c_1 A_r}{c_2(1-l)}}$, where $1 \leq w_{opt} \leq \min(L, A_r)$. The corresponding time window size is $L/S \leq t_{opt} = w_{opt}/S \leq L/S_{min}$, where S_{min} is the minimum request rate of the server. The unit costs c_1 and c_2 can be determined in terms of computation and communication latencies for a typical server and network. These costs must also take into account the communication and computation costs relative to a particular application. Other considerations, such as the size of the Verifier cache, can also play a role in determining the optimal window size. We note that even unoptimized widow sizes would lead to significantly lower request overhead than any client puzzle mechanism.

5 Conclusions

The vulnerability of public services to flooding attacks in the Internet will continue to grow as the gap between network line rates and server rates continues to increase. Whether this vulnerability materializes as a significant service threat depends on whether new Internet services (e.g., voice over IP) will provide sufficiently attractive economic targets for attacks. Countering this vulnerability requires end-to-end solutions, yet those currently available based on client puzzles are inadequate as they offer very weak access guarantees at high per-request overhead. The solution presented in this paper is one of a class of user agreements that can be implemented without new network layer support, and thus offers practical advantages over current proposals.

Future research includes (1) the determination of rate-control parameters and policies for different types of Internet services, (2) the performance evaluation of different types of rate-control servers and request verifiers, (3) the experimental deployment of several rate-control servers and verifiers for different types of application services, (4) the investigation of MWT guarantees for client requests to multiple servers as might be necessary is distributed transactions over the Internet, and (5) the prevention of network-link flooding.

Acknowledgments

Discussions with V. Bharadwaj, R. Bobba, L. Eschenauer, O. Horvitz, R. Koleva, and H. Khurana over the past year helped sharpen many of the arguments presented in this paper. This work was supported by the Defense Advanced Research Projects Agency and managed by ITT Industries under the NRL contract N00173-00-C-2086. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, NRL, ITT Industries, or the United States Government.

References

1. L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," Advances in Cryptography - EUROCRYPT 2003, Warsaw, Poland, May 2003.
2. T. Aura, P. Nikander, and J. Leiwo, "DOS-Resistant Authentication with Client Puzzles," Proc. of the 8th Int'l Security Protocols Workshop, Cambridge, U.K., April 2000, LNCS vol. 2133, Springer Verlag, pp. 170-178.
3. D. Bertsekas and R. Gallager, *Data Networks*, second edition, Prentice Hall, 1992.
4. D.W. Bromley, *Making the Commons Work: Theory, Practice and Policy,* ICS Press, San Francisco, 1992 (Part 2, describing case studies; cf. [17], p.22, 272).
5. T. Darmohray and R. Oliver, "Hot Spares for DoS Attacks," *;login:*, Vol. 25, No. 7, July 2000.
6. D. Dean and A. Stubblefield, "Using Client Puzzles to Protect TLS," Proc. of the USENIX Security Symposium, August 2001.
7. C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail," Proc. of Advances in Cryptography: CRYPTO '92, LNCS Vol. 740, Springer Verlag, 1992.
8. V.D. Gligor, "A Note on the Denial-of-Service Problem," Proc. of the IEEE Symposium on Computer Security and Privacy, Oakland, California, April 1983. (also in *IEEE Transactions on Software Engineering*, SE-10, No. 3, May 1984.)
9. V.D. Gligor, "On Denial of Service in Computer Networks," Proc. of Int'l Conference on Data Engineering, Los Angeles, California, February 1986, pp. 608-617.
10. V.D. Gligor and P. Donescu, "Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes," 8th International Workshop on Fast Sofware Encryption, FSE 2001, M. Matsui (ed.), Yokohama, Japan, April 2001, pp. 92-108.
11. T. Hagerup and C. Rub, "A Guided Tour of Chernoff Bounds," *Information Processing Letters*, vol. 33, 1989-90, North-Holland, pp. 305-308.
12. G. Hardin, "The Tragedy of Commons," *Science*, vol 162, (1968): 1243.
13. J. Henessy and D. Patterson, "Computer Architecture: A Quantitative Approach," Morgan-Kaufmann, 1990, pp. 8-9.
14. J. Ioannidis and S. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," Proc. of Network and Distributed Systems Security Symposium," San Diego, California, Feb. 2002, pp. 79-86.
15. A. Juels and J. Brainard, "Client Puzzles: A Cryptographic Defense Against Connection Depletion Attacks," in Proc. of Network and Distributed Systems Symposium, San Diego, CA, Feb. 1999.
16. B. Lampson, "Software Components: Only Giants Survive," in *Computer Systems: Papers for Roger Needham*, A. Herbert and K. Sparck Jones (eds.), Microsoft Research, February 2003, pp. 113-120.
17. L. Lessig, *The Future of Ideas: The Fate of the Commons in a Connected World,* Random House, N.Y., 2001.
18. J. K. Millen, "A Resource Allocation Model for Denial of Service," Proc. of IEEE Symposium on Security and Privacy, Oakland, CA. 1992. (also in the *Journal of Computer Security,* vol. 2, 1993, pp. 89-106).

19. D. Moore, G. Voelker, and S. Savage, “Inferring Internet Denial of Service Activity,” Proc. of 2001 USENIX Security Symposium, Washington D.C, August 2001.
20. *Trusted Computing Platforms – TCPA Technology in Context*, S. Pearson, B. Balacheff, L. Chen, D Palquin, G. Proudler (eds.), Prentice Hall PTR, 2003.
21. J.H. Saltzer, D.P. Reed, and D.D. Clark “End-to-End Arguments in System Design,” *ACM Transactions on Computer Systems*, Vol.2, Nov. 1984.
22. L. Wang, V. Pai, and L. Petersen, “The Effectiveness of Request Redirection on CDN Robustness,” Proc. of the 5th Symp. on OS Design and Implementation (OSDI), Boston, Mass. December 2002.
23. X. Wang and M. Reiter, “Defending Against Denial-of-Service Attacks with Puzzle Auctions,” Proc. of IEEE Symp. on Security and Privacy, Berkeley, CA, May 2003.
24. J. Xu, R. Lipton, and I. Essa, “Hello, Are You Human,” Technical Report, Georgia Institute of Technology, November 2000.
25. C.-F. Yu and V.D. Gligor, ”A Formal Specification and Verification Method for Preventing Denial of Service Attacks,” Proc. of the IEEE Security and Privacy Symposium, Oakland, CA., April 1988, pp. 187-2002 (also in *IEEE Transactions on Software Engineering,* vol. SE-16, June 1990, pp. 581-592).

Guaranteeing Access in Spite of Distributed Service-Flooding Attacks
(Transcript of Discussion)

Virgil D. Gligor

VDG Inc., Maryland 20815, USA

I'd like to be able to offer some guarantees of access in spite of flooding attacks in distributed denial of service instances. This is a topic that has interested me for a number of years, and hopefully I have now got a handle on it.

In large open networks, in which all clients are legitimately authorised access to a particular service – regardless of their speed, or their location, or any other attributes – we'd obviously like to have client registration and authentication become unnecessary for service access. Of course, authentication might be necessary to control access to certain objects within the particular service, but that has nothing to do with whether or not the client can actually address or access the service.

In these large open networks there are obviously some theoretical bounds on the number of clients that could access the service even via the Internet, but that bound is practically unknown. However, there is usually a known bound on the aggregate client request rate at the server. In other words, a server can only get as many bits per second as the router in front of it is going to pump into the server, and we know how many bps that is.

I am interested in a particular kind of attack, namely flooding attacks against application servers. These attacks persist after all other kinds of distributed denial of service attack have ended. In other words, suppose that there is absolutely no denial of service attack at the transport layer or below, suppose that we fixed all the bugs in the server, suppose that the server does the most wonderful resource allocation strategy you can think of, yet you still have flooding attacks which you have to cope with. In general denial of service is not an end-to-end solvable problem, and clearly this form isn't, because generally at least one of the ends, namely the client end, is not particularly cooperative.

So the approach I am going to take is to use a notion of user agreements, or client agreements, a notion that we introduced around 1988 while trying to show that we can prevent denial of service by using this formal specification for concurrent services. These agreements are going to be both client-to-server agreements and client-to-client agreements. They are generally thought of as constraints that we can place on the client. And of course we have to add a verification component to these agreements to enable us to figure out whether or not the clients really satisfy the agreements. The major technical observation here is that we can actually define some simple agreements that can be verified at the line rate. In other words, we can factor out the specific service that deals with these agreements, and use a verifier that can verify these agreements at the line rate, so therefore this service cannot be flooded.

So that's basically what I'm doing. It's simply a reduction to a solvable problem; I am passing this problem to the domain of building a machine for defining and verifying agreements fast (at the line rate). Once I do that I can argue that I've solved the problem.

Jari Arkko: A clarifying question, verifying machines in front of the server or close to the client?

Reply: Verifying machines in front of the server.

Matt Blaze: Line rate could mean two things, the client side line rate or the server side line rate. I understood it to mean the server side from the beginning but now I'm not sure.

Reply: Suppose that we factor out the server that does the agreement definition, and have another server (an FEP) in front of the server, so the FEP verifies the agreements. The servers may have different line rates, but they're going to be network line rates. For example, the line rate in front of the server might be somewhat faster than the one behind it.

Matt Blaze: OK, so we don't care about the client's line rate?

Reply: No.

OK, we have dual goals. We want to protect the server, to make sure the request rate to the server is controlled in some way, so the server doesn't crash, that's reasonable, and that can be accomplished in many ways, all of them very nice and useful. But the second goal is harder to achieve; we want to provide some client guarantees, and typically these guarantees refer to the waiting time that clients will have in accessing the server.

I'm going to be a bit more specific later, but from the time when I put the request, till the time I get the access, ideally I should have some sort of a *maximum* end-to-end waiting time, independent of the attack.

A weaker guarantee might be just to have a finite waiting time. (Of course if the maximum waiting time tends to be very large it goes to a finite waiting time.) So this is *eventual* access: you don't know exactly when you're going to get in, but you will eventually. And finally there is *probabilistic* waiting time, in which you have a given non-zero of probability of access on any request. That's a different kind of guarantee, and there may be some relationship between these three types of guarantee. I haven't looked at this very carefully yet.

Anyway, in our threat definition we have the service flooding I already mentioned. In addition to that, the clients may be scheming zombies; in other words, you have some sort of adversary that can capture the zombie machines, and the code that the adversary installs can establish a protocol among the zombies and then they can coordinate their attacks. For example, if you challenge the zombies with puzzles, they might in fact solve the puzzles at different rates, but then they all wait until everybody's done their solution, before they actually launch an attack.

Protection against different kinds of men in the middle is not an issue here. If you have a man in the middle, that man in the middle can flip any bit he wants in your encrypted message, destroying its integrity, and making that message useless at the far end. He can deny you service simply by flicking bits, and there is no real way to defend against that, so it cannot possibly be a goal of this research. Nor are we interested in quality of service guarantees such as aggregate throughput, or minimising costs, or so on.

Some general observations about the denial of service problem. (These are quite important to actually figure out whether or not this whole approach makes sense at all.) One observation is that, if you have a layered system, denial of service freedom at layer N really does imply, in general, that you have denial of service freedom at level N-1. Now what this means is that denial of service is not really an end-to-end problem, even when the ends cooperate, because you are depending on the entire stack all the way down to layer 0. Of course we know that in general the ends don't cooperate.

The second observation is that the converse is false. If you have denial of service freedom at layer N-1, that does not imply that you have denial of service freedom at level N. I'll come back to this in a minute to show you why this is going to be this way for the foreseeable future, if not forever.

Third observation is that, if you have a solution for denial of service at N-1, that solution may not be always replicatable at layer N. In other words, if you have, for example, push-back at the transport level, the network layer, and below, then that doesn't mean that you can use push back at the application, because you may have to push somebody back who is pretty determined. So solutions at layer N-1 might not be replicatable at layer N.

John Ioannidis: When you say replicate do you mean use a similar strategy, or just that there is a solution at the higher layer?

Reply: I'm making the bold statement that they need not be identical solutions. I'm conceding that the similar strategy doesn't necessarily work, it's likely that at layer N we have to have a distinct solution. The challenge is, assuming layer N-1 is denial of service free, provide a solution that assures denial of service freedom at layer N.

Now coming back to point two is an observation that application service flooding cannot be eliminated by ISPs because ISPs are generally at the layer below the application clients and servers.

Frank Stajano: Is the top layer, the application layer, the one with the highest number? If so, then why is the outer ring the one with the lowest number?

Reply: Yes. The application layer is layer 7, because of the implication. The rings represent Venn diagrams, not a drawing of onion layers! They express implications.

Alright, so the idea is that *we* have to worry about application service flooding since ISPs are not going to do anything for us. They deal with the layers below. So, for example (and by the way, this is a consequence of network economics, it's likely to stay that way for the foreseeable future) lots of people noticed the flooding attacks against CNN, eBay and Yahoo, but the ISPs observed absolutely no unusual traffic, despite all these servers being in fact down.

So obviously we've noticed this in other contexts. For example, Moore, Voelker and Savage published some measurements of denial of service in the Internet in 2001. They listed the maximum request rate at servers that they noticed as 679 kilo packets per second. Now looking at some other research that's been done recently by Larry Peterson's group at Princeton, they actually measured and designed systems that would have a maximum server rate of about 38,000 requests per second, and if we say each request is a packet then we get the maximum right now to be about 40 kilo packets per second. This is at least one order of magnitude slower than what you can pump into those servers. And by the way, these are very sophisticated designs, these are the kind of design in which the server is replicated dynamically. We're talking about at least an order of magnitude of difference, consequently this is something to be concerned with.

The approach I want is one based on the notion of client agreement. I'm going to look at different kinds of client agreements that people have used in the past, and then specify one of possibly many agreements.

The first observation is that whenever we have a shared service you have undesirable dependencies in the service that the service imposes upon the client. This is a fundamental observation, about the tragedy of the commons. If you have a common place to graze, and you have a bunch of selfish shepherds who each has a bunch of sheep, what's going to happen is each shepherd is going to increase the size of his herd until there will be absolutely no grass left on which to graze. The same thing can happen here. In the Internet if you have lots and lots of clients, and lots and lots of them are zombies, all accessing the same service at the same time, then there will be very little service left for anybody else. This is obviously a tragedy we have to cope with.

Now, the second observation is that you can, should and actually in our opinion must, impose agreements on clients' behaviour outside the server. In other words, make these zombies agree to behave. And this of course reminds me of a short story about a CIA guy and a KGB guy at a pyramid in Egypt. They are asked "Is this the tomb of Rameses the Second?" The CIA guy goes in and comes out and says "Insufficient evidence." The KGB guy goes in. An hour later he comes out and he says "Definitely it's Rameses the Second." Everybody asks how he found out. He replies, "Well, I got him to agree." So if the KGB guy can get the mummy of Rameses the Second to agree, I really should think zombies could be more cooperative.

By the way, user agreements have been used in the past in different areas. For example, you have agreements in which you have local state information that is required to write these agreements.

In ethernet, the exponential back off is an agreement that the ethernet card has, and we can guarantee probabilistic waiting time to the shared medium bidding algorithms for collision handling and multi-access protocols; the stuff that Capetanakis discovered in 1976 in his PhD thesis[1]. This applied to ethernet gives you basically finite waiting time guarantees, not just probabilistic waiting times.

Then you have resource request order agreements. If you design your operating system such that each OS process requests resources always in the same order, then you can show that there is no deadlock and so on.

Richard Clayton: These aren't user agreements, these are manufacturer agreements which users are stuck with.

Reply: Well, these are easy to imitate. I am listing these because that's where I actually got the idea. You also have classic examples of agreements, but these are philosophical agreements, not actual user agreements. You have the dining philosophers work. If they agree not to starve each other, starvation won't happen unless they actually break these agreement conditions which are programmed. You can think of this as an inducement not to be selfish. You may have stateless agreements, and I did some of the agreements that you have in push-back in this area. Although they are not such well defined agreements, they are nevertheless global agreements.

John Ioannidis: An example of a user agreement that it is very easy not to exploit is the TCP congestion control agreements. The real TCP works because everybody walks round with the same code, or a code that begins in the same way, and there are tweaks that you can download that will increase your retransmission rate and get you more service.

Reply: Yes. Clearly client puzzles can give you an implicit client request rate control. However, the problem here is they cannot give you finite time guarantees for clients in these sorts of network, so nobody plays with it. This is just an observation.

The good news is that what happens is true for any client, it doesn't matter who the client is. If you're challenging a client, and keep challenging him with puzzles that are stronger and stronger, at some point his request rate is going to drop below the maximum server rate. It turns out you can go further and do this for multiple clients and if you challenge them with higher and higher strength puzzles, then their aggregate will drop below this maximum server rate, which is an even better thing to happen. Yet in spite of that you cannot guarantee a finite waiting time for all clients.

I think this is probably similar to what Nikander and Aura[2] said about using client puzzles for authentication protocols. This makes sense, because you'd like to make sure that your authentication server doesn't crash even though occasionally you don't get access.

[1] J.I. Capetanakis, "Multiple Access Broadcast Channel: Protocol and Capacity considerations", Ph.D. dissertation, MIT, EECS Department, Cambridge, MA., 1977. Also in IEEE Trans. Information Theory, IT-25, 505–515, 1979.

[2] T. Aura, P. Nikander, J. Leiwo, "DOS-Resistant Authentication with Client Puzzles", LNCS 2133, 170–181.

The problem is that you have no finite waiting time guarantees. So let's start with a very simple example. Essentially what you have is some sort of a ticket server (let's forget what the tickets are for the moment) and a ticket verifier. Suppose that we are in a bakery. You go and pick up a sequentially numbered ticket. You can see which request is being handled at this point, and you know your request number, so you know what the maximum waiting time is, because you more or less know what the service rate is in that bakery.

So, start looking at the problem backwards. Don't worry about not crashing the server, worry first about maximum waiting time. That way you can come up with some sort of an algorithm (which is an agreement) that will work like this. All the clients, no matter how many of them there are, will go to this ticket server which generates this sequence of bits very fast, computes a map on them so they cannot be forged, and returns them to the user. The verifier similarly verifies the map and determines whether or not to let the requests go through. These are very simple computations, very trivial agreements, that can actually be generated and enforced. Of course the server and the verifier would operate at the appropriate line rate of their routers and share a key.

So essentially you get guaranteed a maximum waiting time. There are two problems here. One is that the maximum waiting time is going to be very large, because if you are going to have lots of zombies each one of them will bind to a ticket so you are going to have issued all these tickets, and legitimate users will have to wait. Further, suppose that all these zombies don't send their tickets. If they don't send their tickets the server will wait for a little while, and then go to the next ticket, and the next ticket, and so on. That might be a problem in itself. Further notice that on every request you have to take a ticket, you have an interaction between the clients and the ticket server, and this may or may not be unpleasant.

Now let's come back to a question about the maximum waiting time after you get your ticket. The assumption is that since the ticket server operates at line rate, whenever you go to it to get a ticket you'll get one, and since there is no denial of service in the network layer and below, which is our assumption, that means that there cannot be denial of service at that ticket level. So consequently you have approximately an end-to-end client maximum waiting time.

That's not the solution by the way, this is just to give you an idea of what the solution is like.

Pekka Nikander: You mentioned that the tickets are protected by a MAC, but do you need one? Wouldn't it be just enough that you have a random number generator? You could run the random number generator at a constant rate at the verifier, so that in each time window you accept a certain number of those random numbers. Then when you return the ticket to the client it just says the waiting time, and the random number. If you send the ticket at the right time, it will fall into the window where the random numbers are and indexed by the ticket number or something like that, so that you don't need the MAC. Would that be possible?

Reply: I think a MAC might be faster, because they can be generated in parallel. And with a MAC I can make sure that there are no collisions, which there may be with a random number generator.[3]

Richard Clayton: It's not the end of the world if there's a collision anyway.

Ross Anderson: Suppose there are some people (whom we shall call friends) who are actually conducting an international denial of service attack. In these circumstances, a protective mechanism like this wouldn't work. There's millions of kilo packets per second and acres and acres of computer path.

Reply: I think that the problem with friends is more that they are always in the middle.

The point is that this doesn't matter at all. Remember the assumption is that there is no denial, there is no flooding at transport layer and below, so we are not worried about that.

Ross Anderson: Consider 100+ Universities in Britain, each with two gig connectivity. If the spooks wanted to be in the middle for every University they would have to haul 200 gigs. My point is that, if you're really worried about stopping the denial of service attacks allegedly happening at the moment, then this threat model is wrong. Single points of failure, like a cookie server, can cause a fair amount of difficulty, but do not have to worry somebody who's everywhere in the middle all the time.

Reply: Usually what people object to is not that. Many people object to the assumption that at the transport layer and below there is no denial of service. That's an assumption which offends a lot of people, but I have full trust in the talents of the people who are dealing with that problem. If you agree that transport flooding is not an issue, and if you agree that we can build this thing to operate at whatever line rate you want at that point then, under the men-in-the-middle attacks which I'm not willing to consider here, I claim there is nothing to worry about.

Richard Clayton: I was just going to observe that you can have multiple copies of the cookie server. The crucial objection is the one you've already raised; that you can turn up, get millions of tickets and never use them. Doesn't that make this approach worthless?

Reply: Time to put up the next slide. This scheme in the bakery with the ticket works really is just one part of a parallel solution. There are reasons why the other solutions didn't quite work, but I claim this one does.

Imagine that we have a mechanism to control information. In fact the mechanism that I'm suggesting is a reverse Turing test. You run this reverse Turing test that spots the presence of a human behind an applying machine, and you make sure that you cache the decision in a cookie that cannot be forged very

[3] For example MAC = $E_k(n)$ where n is a sequence number.

easily, or be duplicated to good effect. It can be duplicated but duplicating it is not going to do you much good and if it's stolen you can actually tell more or less that it's stolen.

So essentially this will force human users into collusion. The single guy who's starting up the zombies will now have to be very highly placed in the Chinese people's army, the NHS or an employer of a similar size, to have control of a large set of humans in order to make an attack. So that's the first point, control the zombie proliferation with this method.

Once you've done that, you can stop there because in principle that solves the problem. In other words, you can put into the cookie some sort of a time limit and the number of requests one could use, and that's it. We added this second part because this is not a particularly good or sensitive time-slot reservation scheme.

So you actually solve this reverse Turing test to get your cookie. Based on the cookie you're allowed to get a bunch of tickets, and the tickets allow you a number of requests, and you get the tickets on demands that fit exactly what you need. If you do that you can show that there is a maximum waiting time, and you are going to come pretty close to it. I don't know the details about how you would protect cookies from being duplicated, and so on. It is like in any reservation algorithm, except to say that we generalise these agreements across multiple services, not just one service. And finally we can organise this in many ways, like those I mentioned. We can duplicate the ticket service, and so on, and we can have all sorts of combinations.

Matt Blaze: You seem to have created a much worse problem than the existing kind of windows web user interface where it pops up a little window giving the user a warning that they have to slot by clicking on. Now they are going to be presented with something that they can't even just slot away by clicking on it, because it's an actual reverse Turing test. I have this sense that this is going to make the user experience extremely annoying.

Reply: No the user experience is going to be exactly the same. With this notion of CAPTCHAB[4] or reverse Turing test, users would have an identical interface to the one which you mentioned; it will be absolutely no different than trying to set up an account with Yahoo.

Matt Blaze: But that's incredibly annoying. I only have to set up my Yahoo account once.

Reply: This is more like logging into your account. You click on this once if you want to access your account, just like you click on your password.

Richard Clayton: But Yahoo challenges you to read the numbers from the background which is designed to look like a hard puzzle. So maybe you need a human zombie, like a grad student or somebody, to actually do the pattern

[4] L. von Ahn, M. Blum, J. Langford, "How lazy Cryptographers do AI" CACM 47(2), 56–60, Feb 2004.

recognition required. But once the grad student has done that, obviously you're on the net.

Reply: These are not number theoretical problems like factorisation, they are better because if you can break this you've actually solved a very hard problem that everybody can benefit from. In fact there are classes of problems that actually can be used as reverse Turing tests. These have appeared in the last year, and you may want to look at the website at Carnegie Mellon[5].

[5] The CAPTCHA Web Page: http://www.captcha.net. See also L. von Ahn, M. Blum, N. Hopper, J. Langford, "CAPTCHA: Using Hard AI Problems for Security", Advances in Cryptology - Eurocrypt 2003, Warsaw, Poland, 2003, 294–311.

Protocol Codesign*

Hassen Saïdi, Victoria Stavridou, and Bruno Duterte

System Design Laboratory, SRI International
Menlo Park, CA 94025, USA
Tel: (+1) (650) 859-3810, Fax: (+1) (650) 859-2844
{saidi, victoria, bruno}@sdl.sri.com
http://www.sdl.sri.com

Abstract. Security protocols are moving from the network and transport layers into application layers to adapt on one hand to new paradigms in distributed applications, and to achieve on the other hand higher level security properties such as intrusion tolerance. We argue that these new protocols ensuring, for instance, intrusion tolerance can be built with the same building blocks as traditional protocols ensuring confidentiality, authentication, nonrepudiation, fair exchange, and anonymity, but need to integrate additional application-specific requirements. This calls for a new design approach where both application and security requirements are refined simultaneously. Our approach, called *protocol codesign*, achieves this goal by providing a rigorous methodology for designing protocols based on the composition of basic services.

1 Introduction

Cryptographic communication protocols are the basis of security in many distributed systems. It is therefore essential to ensure that these protocols function correctly and do not exhibit vulnerabilities that can be maliciously exploited. For this purpose, protocols achieving authentication, confidentiality, integrity, anonymity, fair exchange, nonrepudiation, and contract signing must be carefully designed, tested, and verified. New emerging computing paradigms in distributed systems are leading to new classes of protocols that are not only used to establish a secure link between two remote systems, but rather to achieve new higher security requirements. Electronic commerce transactions involving multiple parties, secure collaborative environments, virtual private networks, and ad-hoc networks are examples of those new applications, where a higher degree of fault tolerance and intrusion tolerance is required. Parties involved in a protocol are not required to be trustworthy, and the protocol must be secure not only against man-in-the-middle attack, but also against malicious attackers who can have access to parts of the distributed infrastructure such as routers, computing

* This research was partially supported by DARPA/AFRL contract F30602-00-C-0087 and NSF contract CCR-0325274.

nodes, and wires. From the security standpoint, therefore, if the network is providing security, the compromise of a machine on the network compromises the overall system. However, if the application is providing security, the overall system is able to tolerate a significant number of compromised nodes within it. Applications providing security can in fact recognize a compromised fellow application since the application can have a better model of the expected behavior of other applications.

In this paper, we examine the main challenges in designing this new line of protocols. Protocols will have to be tightly integrated with other services such as fault tolerance and routing, and will need to be efficient as well as secure. The design space will include all possible combinations of these services. However, only some combinations will meet all requirements. This calls for a new design approach where both application-specific and security requirements are defined and refined simultaneously. Our approach, called *protocol codesign*, achieves this goal by providing a rigorous methodology for designing protocols. The objective of codesign is to analyze the interplay of security mechanisms with functional and fault tolerance mechanisms to ensure that the resulting protocol satisfies its requirements. There are often subtle interactions between mechanisms that may defeat design intent, for example:

– The analysis of some forms of anonymity properties indicates that ensuring anonymity with a high probability decreases fault tolerance. Shmatikov [Shm02] uncovered two such properties in gossip-based systems, such as the Crowds protocol [RR98] using probabilistic model-checking techniques [KNP01]. For instance, if corrupt nodes can detect that messages along different routing paths originate from the same unknown source, then the probability of identifying the source node increases with the number of possible paths. This indicates that the connectivity of the network should allow only a maximum number of possible paths, therefore maximizing the probability of ensuring anonymity.
– In [Rus95] a simple example of combination of a secure service and a fault tolerant service is shown to provide neither security nor fault tolerance. The example shows that to achieve both the security and fault tolerance requirements, it is necessary to deconstruct both services into smaller and weaker components that must be reassembled and composed to ensure that both requirements are met.

Our thesis is that any security protocol can be decomposed into a set of simple building blocks that are sequentially composed as actions of a single principal. A protocol is therefore defined as the parallel composition of the actions of each principal involved in the protocol. We show that when the goals of a protocol are expressed in a simple logical specification language, then every goal expressed in the logic can be achieved by composing a finite number of building blocks. We identify the basic building blocks of any security protocol as the set of possible cryptographic services that a principal can establish with a single individual or

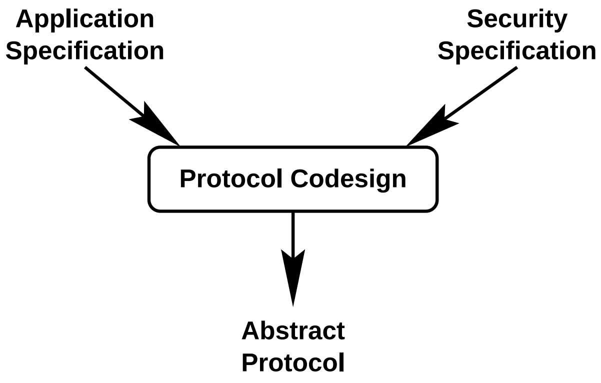


Fig. 1. Protocol Codesign Process

a group of principals through logical communication channels, and we give a formal definition of each service.

2 Protocol Building Blocks

Security protocols provide five major security goals [MvOV96]: confidentiality, data integrity, message authentication, nonrepudiation, and anonymity. If m is a message and A and B are principals, then

- *Confidentiality*, also called privacy or secrecy, ensures that data can be read only by authorized persons, and can be implemented using either symmetric-key or public-key encryption.
- *Data integrity* prevents an unauthorized person from altering a message. Cryptographic hash functions are generally used to provide such a service.
- *Authentication* is considered to be of two types: message authentication and entity authentication. *Message authentication* provides the identity of the author of a message to a given recipient. Message authentication can be implemented using message authentication codes based on keyed cryptographic hash functions. *Entity authentication* provides an identification of an entity in a communication. An important difference between these two notions of authentication is that message authentication is not limited to a certain time period, while entity authentication is limited to the duration of the communication. Entity authentication needs the execution of a protocol. Such a protocol could, for instance, consist in sending some *fresh* information, while assuring message authentication.
- *Nonrepudiation* is the property that binds an entity to a message. A complete nonrepudiation service must ensure both *nonrepudiation of origin* and *non-repudiation of receipt* [Zho96]. Nonrepudiation of origin provides *evidence* to the recipient of a message about the identity of its author. Until today, the only way of providing such evidence has been through digital signatures.

The difference with message authentication is that nonrepudiation provides evidence that can be shown to an adjudicator about the identity of the author, while authentication assures only that the recipient is convinced of the identity of the author. Nonrepudiation of receipt provides an originator of a message with evidence that the recipient received a previously sent message. A complete nonrepudiation service cannot be implemented by a single cryptographic primitive. It needs a nonrepudiation protocol to be run between two entities.

– *Anonymity* ensures that the recipient of a message has no means to derive, from the content or the origin of the message, the identity of the sender. A hierarchy of anonymity properties is defined in [HS03].

Each service can be implemented as a logical channel. These channels can be used as building blocks to design more complex services. For instance, a B-confidential channel provides the service that only the author and B can read the content of m. An A-integrity channel assures that if entity A sends a message m on the network, everyone will receive a nonaltered message, that is, m. An AB-authentic channel provides message authentication between the two entities A and B: whenever a message arrives on an AB-authentic channel, B can be sure that the message was sent by A. Note that the message does not need to be fresh and that B cannot convince anyone else that A is the author of the message. Moreover, authentication does not require the message to be secret. A nonrepudiable channel provides nonrepudiation of origin. When a message arrives on an A-nonrepudiable channel, everyone is convinced that A is the author of the message.

Channels can be implemented using a variety of cryptographic primitives. For instance, a confidential channel established between A and B can be implemented either by using the public key of B, or by using a shared secret key between A and B. Such a choice of implementation may depend on the availability of a public key infrastructure, or adequate software and hardware for generating shared keys. However, all these possibilities guarantee the requirements of a B-confidential channel. We can establish a hierarchy between these channel types. A nonrepudiable channel offers a stronger service than an authentic channel: as an A-nonrepudiable channel convinces everyone that A is the author of the message, it also convinces B in particular. In the same way, an A-integrity channel can be implemented by the stronger AB-authentic channel or an A-nonrepudiable channel, as changing the message m would change the author. A consequence of this hierarchy is that whenever we need a channel offering integrity services, we can implement it by an authentic or a nonrepudiable channel. In practice, more complex services than those provided by the above-defined channels are needed. Therefore, it is important to have the possibility to combine the services of the four basic channels. To do so, we define a message m that can be sent on a channel as either data or a channel in the spirit of the SPI calculus [AG99] through meta-channels. Consider the following example, where

A wants to send a message m to B that has first been digitally signed by A and then been encrypted for B:

$$\mathsf{send}_{B-confidential}(A,\ \mathsf{send}_{A-nonrepudiable}(A, m,))$$

This example shows the expressive power of the basic building blocks and their combination. B is the only one except the sender who can get the information sent on the confidential channel. Receiving a nonrepudiable channel means that B has all the knowledge to build this channel. B can thus decide to send this information to everyone else—and convince everyone else that A is the originator of message m—or decide to keep this information secret. By the same mechanism we can implement nested encryption and other more complex cryptographic mechanisms.

3 Protocol Codesign

Security protocols can be built from simple communication primitives provided by standard communication protocols, and from cryptographic primitives. The rules of the protocol, that is, message content, can be derived from a high-level specification of what the protocol is designed to achieve. In [Saï02], we show that when such a higher-level specification is expressed in a logical formalism to which one can associate a reasoning framework, it is possible to automatically synthesize security protocols from that specification. The codesign approach consists in defining the basic services that need to be used in order to achieve the protocol's goals and the way these basic services need to be composed.

To illustrate the ideas of protocol codesign, we propose the definition of a simple logic L for the description of security goals. The logic is inspired by logics of knowledge with constructs similar to the ones used for the well-known BAN logic [BAN90]. The logic includes a knowledge modality equivalent to the belief modality of BAN. However, we distinguish the cases where the statement $A|\equiv X$ is used to refer to a proposition X in the logic L, and the case where X is data such as a message, a key, or a nonce. Unlike the BAN logic that describes both properties of protocols and the protocols themselves using, for instance, the *sees* and *said* operators, the logic L does not describe protocols. It rather describes the knowledge of principals. Similarly to the BAN logic, we also consider nonces, freshness, and shared secrets.

The semantics of a protocol is represented by all possible runs. A run of a protocol is a sequence of states, and transitions between states represent protocol actions. Following Paulson's modeling approach [Pau98, MR00], we define for a system state $q \in Q$, $trace(q)$ the set of messages that have been exchanged so far and $\underline{trace}(q)$ the message contents that occur in $trace(q)$. Given a set of fields S, the following sets are used: $\mathsf{Parts}(S)$, $\mathsf{Analz}(S)$, and $\mathsf{Synth}(S)$. $\mathsf{Parts}(S)$ is the set of fields and subfields that occur in S. $\mathsf{Analz}(S)$ is the set of fields that can be extracted from elements of S without breaking the cryptosystem. $\mathsf{Synth}(S)$ is the set of fields that can be constructed from elements of S by concatenation and encryption. Formal definitions can be found in [Pau98] or [MR00]. In a state q, the set of fields that G can access is then

$$\mathsf{Know}(G, q) = \mathsf{Analz}(I(G) \cup \underline{trace}(q)).$$

This is the set of fields that G can obtain from its initial knowledge $I(G)$ and the messages seen so far. Our model allows us to characterize the state where a certain formula is valid as a state in a particular sequence of events that corresponds to the messages observed so far. This allows us to construct, for a set of goals expressed in the logic, a trace of events that starts from the state that satisfies the initial configuration of the protocol, and leads to the state where all goals are satisfied.

As a simple example of the use of the logic, we show how an authentication protocol can be designed and generated from its logical specification. The goals of the protocol are as follows:

$$\begin{array}{r} A| \equiv B| \equiv A \overset{N_b}{\longleftrightarrow} B \\ \wedge\, B| \equiv A| \equiv A \overset{N_a}{\longleftrightarrow} B \end{array}$$

The protocol achieves entity authentication between two principals A and B by sharing two secrets, N_a and N_b, generated respectively by A and B. Sharing the secrets A and B becomes bounded common knowledge. The goals of the protocol can be simplified as follows:

$$\begin{array}{r} A| \equiv B| \equiv N_b \in \mathsf{Know}(A, i) \wedge N_b \in \mathsf{Know}(B, i) \wedge N_b \in \mathsf{Know}(Z, i) \\ \wedge\, B| \equiv A| \equiv N_a \in \mathsf{Know}(A, i) \wedge N_b \in \mathsf{Know}(B, i) \wedge N_a \in \mathsf{Know}(Z, i) \end{array}$$

where Z is a malicious attacker, and i is the final state of the protocol run.

$Formula$	$Meaning$
$P\|\equiv X$	P knows X
$P\|\Rightarrow X$	P has jurisdiction over X
$\sharp(X)$	X is fresh
$P \overset{K}{\longleftrightarrow} Q$	K is a symmetric key for P and Q
$P \overset{X}{\longleftrightarrow} Q$	P and Q share secret X
$\overset{+K}{\mapsto} Q$	$+K$ is the public key of Q
$\{X\}_K$	X encrypted with K
$\{X\}_{K^{-1}}$	X encrypted with the private key k^{-1} corresponding to the public key K

Fig. 2. A Simple logic for protocols

The codesign approach generates the following sequence of rules or actions that achieve those goals. These rules are generated following the derivation process defined in [Saï02].

$$\begin{array}{l} A \longrightarrow B : \{N_a, A\}_B \\ B \longrightarrow A : \{N_b, B\}_A \\ A \longrightarrow B : \{N_b, A\}_B \\ B \longrightarrow A : \{N_a, B\}_A \end{array}$$

The rules of the protocol consist of each principal sending its generated secret to the other party. Upon reception, each principal acknowledges the reception of the secret by sending it back to the original sender. The codesign principals indicate that in order to achieve bounded common knowledge of a secret, it is necessary to use confidential and authentic channels that allow on one hand the preservation of the secret and on the other hand the authentication of the origin of the sender. The derivation rules described in [Saï02] can be viewed as a Hoare-triplet defining each service and its implementation as a cryptographic channel. The definition of the confidential channel, for instanc is

$$\{P\} \;\; \mathsf{send}_{B-confidential}(A, m) \;\; \{Q\}$$

where P is the precondition that expresses that A either shares a secret key with B, or that A knows the public key of A. The postcondition Q expresses that B knows the content of m, and that the set of principals who know the content of m is augmented with a single principal B.

References

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. Information and Computation, 148(1):1-70, 10 January 1999.

[BAN90] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. ACM Transactions on Computer Systems, 8(1):18-36, February 1990.

[HS03] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: A modular approach. Journal of Computer Security, 2003. To appear.

[KNP01] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. Technical Report 760/2001, University of Dortmund, September 2001. Also in Proc. PAPM/PROBMIV 2001 Tools Session.

[MR00] J. Millen and H. Ruess. Protocol-independent secrecy. In RSP: 21th IEEE Computer Society Symposium on Research in Security and Privacy, 2000.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. CRC Press series on discrete mathematics and its applications. CRC Press, 1996. ISBN 0-8493-8523-7.

[Pau98] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. Journal of Computer Security, 6:85-128, 1998.

[RR98] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security, 1(1):66-92, 1998.

[Rus95] John Rushby. Combining system properties: A cautionary example and formal examination. Technical report, Computer Science Laboratory, SRI International, June 1995.

[Saï02] Hassen Saïdi. Towards automatic synthesis of security protocols. In Logic-Based Program Synthesis Workshop, AAAI 2002 Spring Symposium., Stanford University, California, March 2002.

[Shm02] Vitaly Shmatikov. Probabilistic analysis of anonymity. In Proc. 15th IEEE Computer Security Foundations Workshop, pages 119-128, 2002.

[Zho96] Jianying Zhou. Non-repudiation. PhD thesis, University of London, December 1996.

Protocol Codesign
(Transcript of Discussion)

Victoria Stavridou

System Design Laboratory, SRI International, USA

This afternoon I'm going to talk to you about some work that we're doing on protocol design. This is actually Hassen Saidi's work; he spoke a little about this at the workshop here a couple of years ago[1] and since then there's been quite a bit of progress.

There are many challenges in the design of protocols; both the ones that we have today, and the ones that we need to evolve in the future. The conversation here this morning clearly identified those challenges, so I don't think I will preach here. But protocols are changing, and they're changing form in a couple of ways. They're changing by moving from the traditional place where one would do a protocol – from the network layer into the application layer – because the applications themselves are changing. Also our expectations of the technology that we develop for security applications has changed. For example, now we talk about intrusion tolerance; it's not enough to build something that is secure, I also want to build it in such a way that even if attackers succeed in penetrating it it will still provide some level of service. In that one might see the influence of fault tolerance: today's applications not only have traditional security requirements, but also things that have not traditionally been thought of as properties security protocols would implement. Nonetheless, now that we're putting them together the protocols have got to do both jobs.

New applications need new protocols and sometimes that happens, but sometimes known protocols get re-engineered (sometimes well, mostly badly), and what tends to happen is that unless one is very, very careful and thoughtful and systematic about the way that protocols are re-engineered or composed, you may end up actually making things worse. Rushby has a good example[2] about putting together two protocols, a fault tolerant protocol and a security protocol, and ending up with something that is neither secure nor fault tolerant. We've been driven not just by changes in the application, but also by this variety of properties that they have to implement, so we need to understand the interactions of the properties, and the subtleties that those interactions entail, and the impact that those subtleties have on the final product: by and large this is a darn hard thing to do.

So how can you make this progress easier and more systematic? One idea that we've been looking at for a while now is this notion of security co-design –

[1] H. Saïdi, "Intrusion Tolerant Group Management in Enclaves", LNCS 2467, 213–216.

[2] John Rushby, "Combining System Properties: A cautionary Example and Formal Examination", Computer Science Laboratory, SRI International, June 1995.

those of you with a background in hardware design or verification will recognise the term. Hardware people understood a long time ago that in design they trade off performance for power, for accuracy, for a whole bunch of things, and they have developed a set of very well understood practices and tools. There are companies out there that produce tools that support exploration of trade-off space and all these different factors. Security is just another constraint, so why not co-design the constraints with the functionality constraints that the mission of the system implies? If you have a good enough understanding of the mission of the system that you're trying to build, you can derive both the functional operational requirements and a bunch of security requirements. One can then define those things into the defence mechanisms in the case of security and the components in the case of functionality, and all the stages will come together in the architecture in a specific way so that the constraints from either side are respected and maintained.

An instance of this programme that we've been working on is what we call protocol co-design. So what are the system constraints that a protocol needs to take into account? Well, the kinds of constraints on policy that it needs to work with and the requirements (what do you want the protocol to do, what does the application want to do). The notion is that we can trade off these different requirements through a systematic design process and thereby produce a protocol.

This is like a microcosm of the co-design area in general. So how can you make this systematic? Imagine if we could decompose security protocols into the same basic building blocks, and that for any kind of protocol one could build there is a set of primitives that you can compose in order to get there. One idea would be to try to identify what those primitives, or basic building blocks, could be. So here we show you a first cut of some of these. I don't for a moment claim that they are complete, but they are an example of how one might think about the building blocks. Hassen thinks of the building blocks as basic services between two principals. The notion is that by describing the desired state of the principals it's possible to derive the combination of building blocks that are needed to achieve this goal. If you have the basic building blocks, the protocol goals, and a set of principals, then the co-design is the process of putting those things together in a way that satisfies both the requirements and the constraints.

So we need a logic of knowledge. Although this looks like BAN it's not quite. Then we need a protocol specification in terms of the knowledge of the principals. If we know what the primitives are then we also need derivation rules that can take us from the goal to the actual protocol, and these derivation rules are expressed in terms of pre- and post-conditions.

The approach then becomes obvious: given the goal and initial states we use the derivation rules to generate a sequence of primitives that satisfies the protocol specification and meets the requirements and the constraints.

Examples of the primitives that we have include: the confidential channel (makes sure that only the author of the message and the recipient of the message can read its contents); the data integrity channel (which guarantees that the recipient will get a message that has not be tampered with); the authentic channel (that guarantees the authenticity of the message, so if a message arrives on the authentic channel, then the recipient has a guarantee of who the author is); the non-repudiable channel (guarantees that the recipient of the message can convince its peers that the message actually was authored by A); and the anonymous channel (guarantees that the recipient of the message cannot observe who the author is).

Now is this complete? I was very interested by the presentation of Asokan[3] the other day, I thought that would be a logical example for us to work on in the coming months, you'll see why in a moment. I think one of the things that would be very interesting to do would be to take Asokan's example and see what one would need in terms of primitives to make it a specification for before and after, and then perform the analysis: that would be a very good way of testing the water.

Virgil Gligor: In relation to co-design, you mentioned at the beginning that you are putting together operational requirements and security requirements to do this codesign. The security roles were quite clear, but I didn't see any operational requirements to be blended in.

Reply: In fact what's missing is not the operational requirements, I would have been a lot happier if this example included fault tolerance.

This is the minimum that we need to be able to do in order to talk about co-design. Different kinds of properties could be functional, could be fault tolerance, and so on. This is just the starting point; we are putting together the machinery.

Virgil Gligor: So the question is, is it possible always to separate in this way? In other words, if you get the machinery for security, and there is some machinery for...

Reply: My thought would be that if the logic of knowledge turns out to be the right way to write this text for the goals, then I would like that same logic to capture everything that I talk about. If it doesn't we're in trouble. So if I want to talk about fault tolerance objectives, for example, that same knowledge logic ought to be able to cover that, and if not then we need to expand it.

Virgil Gligor: There are some examples which show that this can be the synergy. For example, we have this business remit on one hand and authentication and security on the other hand. We put them together and we get a good result, better than what we had before. Is that the kind of thing that you have in mind?

Reply: Right, this is the next step.

Ross Anderson: In this protocol you've got towards the end of the position paper, one thing that does strike me, although I don't see how it could be effective in an attack, is that you have two exchanges, one involving the nonce N_A, and one involving the nonce N_B. These are symmetric, but they are not specifically linked in any way. This leads, for example, to the following observation. Alice

[3] N. Asokan, these proceedings.

is reasonably certain that if Bob is honest, then N_A and N_B correspond to the same run of the protocol, even if there is a bad person in the network, but poor Bob isn't similarly certain. This leaves me with a slight feeling of unease, and I suppose if I were compelled to use something like this I would ensure that Alice sends to Bob some kind of check line, so the dates are encrypted but under the key hash(N_A) and hash(N_B), because that then gives Bob a symmetrical assurance. The fact I won't immediately see whether there's been an attack still leaves me mildly uneasy.

Bruce Christianson: You can combine the second and fourth messages of the protocol by the way.

Richard Clayton: I was going to make another generic remark which is that what we tend to look for in protocols these days are properties that mean you don't have false state if you're attacked by somebody. For example, with this protocol where C fakes a message which appears to come from A, and then B has to hold off an indefinite amount of state. The protocol will fail; both will be sitting there holding a state waiting for the next message to come back and it will never come back because it will just jam. But the other thing you'll tend to be looking for is a lack of amplification, in that there's no way that one message can ever produce two, because again that leads to network related denial of service attacks. It seems to me that you're going to need to put a bit more into your generic logic in order to reflect those sorts of rather more modern ideas about what we want to see in a protocol.

Reply: That's fair. The other concern I have is in relation to derivation rules. We should have some handle on the optimal result. I'm sure there is more than one way of reaching the goals, and creating the final states and the initial states. So how do we tell which is a good protocol as opposed to another? Is there some way of differentiating?

Matt Blaze: My intuition is that when you separate the operational requirements from the security requirements, you would very often discover that they're in conflict. An example of that would be an operational requirement which says a protocol, a system, has to be as efficient as possible. One way of increasing its efficiency is to take out all that security overhead. Obviously that's a trivial example, but I did that in a hurry. [Laughter]

Have you looked into this?

Reply: That kind of problem is exactly what this protocol is trying to solve. The question is, as you go down the path can you care about operations and also care about security and make agreements amongst the functions? Can we create an approximation to both sets of requirements that is good enough?

Matt Blaze: And good enough is determined by people, not by this engine?

Reply: It would be determined by the operational security requirements. One hopes that they would not be discrete points, they would instead be a spectrum, and in that spectrum there would be a set of solutions.

Matt Blaze: Perfect, but a human being does that, OK.

Bruce Christianson: Some security properties are like anti-functional properties; they say the system doesn't exhibit certain functional properties that you don't want it to have, but some are more like performance anti-properties. Denial of service says that there's no efficient way to implement denial of service.

Virgil Gligor: So this system works fine for safety properties, it may work for liveness as well, but denial of service is harder. Can you put any constraint on it such as this particular program cannot get into a state out of which it cannot come, such as this particular program cannot write to that subject and that constraint can be checked?

Reply: Right, but you can't rely on the environment. Somehow things external to the system have to be captured and talked about, but I don't think this can help you. If you are relying on a critical message to arrive some place else, that some place else is not really the core of what we're building.

Richard Clayton: The sort of property you want is, no part of this system acts as an Oracle, because lots of things break that way. It is rather difficult to express in terms of knowing things.

Reply: My sense is that if you have got the time to start talking about other kinds of things, not just security, then this logic will change.

Richard Clayton: And also practical attacks where, if you can generate enough related messages, you can intersect them, or maybe break things that way. Again, any theoretical analysis of these things always says that keys are infinitely long, are never known by anybody else, and so forth, whereas practical security tends to suggest that those assumptions are incorrect.

Reply: I'd welcome other examples of protocols that have been put together with good intentions and have failed horribly because of the Blutack.

Virgil Gligor: One idea, delivered here in 1998[4], was that you take a policy based on the constraints that you're talking about – safety constraints - such as Bel LaPadula type policy – for mandatory access control; and then you take a Biba policy for integrity control, again based on safety constraints. If you combine them, in other words every check has a Biba check and a Bel LaPadula check very carefully combined by conjunction, then you get an extra covert channel that you didn't have before. That shows that two approximations of information flow policies, when composed, do not give you the same approximation.

[4] V. Gligor, S. Gavrila, "Application-Oriented Security Policies and Their Composition", LNCS 1550, 67–82.

Enforcing Security Policies for Distributed Objects Applications

Bogdan C. Popescu, Bruno Crispo, Andrew S. Tanenbaum, and Maas Zeeman

Vrije Universiteit, Amsterdam
{bpopescu, crispo, ast, mmzeeman}@cs.vu.nl

Abstract. In this paper we present the design and the implementation of a policy engine for enforcing security policies for distributed applications. Such policies, represented by using the RBAC model, include both how the distributed, shared and replicated objects are used, by mean of **role certificates** and how these roles are managed by means of **administrative roles**. The policy engine can enforce not only privileges to invoke methods with particular parameters and under specific conditions but also the permissions to execute such methods. The engine is offered as a middleware service such that application developers can concntrate on specify the security policies for their applications and they are realesed from the burden of implementing the mechanisms for the actual enforcement of such policies.

1 Introduction

In this paper we describe the design and implementation of a policy engine for enforcing security policies for distributed object applications. We show how our design can be integrated as part of the Globe [11] system - a middleware for supporting wide-area replicated objects.

While extensive work has been done in the area of security policy languages and policy engines, this paper makes two important contributions: first we identify a number of security policy requirements that arise in the context of replicated applications, more specifically, the need for policy mechanisms to express different amounts of trust one wants to place into different replicas of the same service. Second, we come up with a design that bridges the gap between an abstract security policy description and the actual service implementation. This is consistent to our goal to provide a policy engine at the **middleware level** which would make it simpler for application developers to integrate the policy engine with their applications. Traditional policy engines [2] work at a more abstract level, which in theory makes them very versatile, but in practice means that developers need to write rather complex translators (for passing parameters and environment variables) in order to bridge the gap between the engine and the application.

The rest of the paper is organized as follows: in Section 2 we give an overview of the Globe system, which is the testbed for the policy engine we have developed.

In Section 3 we describe the trust model for Globe applications; our policy language is specifically designed to support this trust model. In the next three sections we describe the policy language constructs, grouped into constructs for supporting administrative policies, access control and method execution policies. Finally, in Section 7 we give an overview of our implementation, in Section 8 we examine related work, and in Section 9 we conclude.

2 An Overview of Globe

Globe is a distributed system based on replicated shared objects. While the idea of encapsulating functionality into objects is not new (systems like Corba [1], Legion [5] or DCOM [4] rely on this paradigm), what makes Globe unique is that objects not only can be used by a large numbers of users on different machines through remote procedure calls, but also can be physically replicated on many hosts at the same time to improve performance.

The central construct in the Globe architecture is the distributed shared object (DSO). As shown in Figure 1 a DSO is built from a number of **replicas** that reside in a single address space and communicate with replicas in other address spaces. All the replicas that are part of a DSO work together to implement the functionality of that DSO. A replica consists of the code for the application (the code that implements the functionality of the DSO that replica is part of), the part of the DSO state the replica stores, and the replication mechanism. A replica can be hosted by any Globe-aware server connected to the Internet. To be Globe-aware, a server needs to run a special daemon program - the **Globe object server** that provides facilities for hosting, remote creation and destruction of replicas. We also provide a special **Globe Location Service** [10] where DSO register the contact points for their replicas, so that their clients can easily find them.

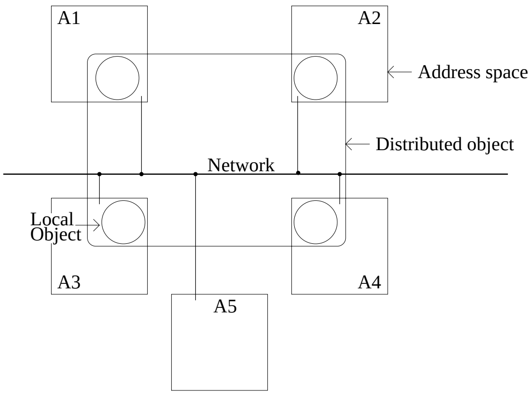


Fig. 1. A Globe DSO replicated across four address spaces

In the context of a Globe, a **client** is an entity that uses a DSO by invoking one of its public methods. Clients and replicas are assumed to operate in different address spaces, thus, in order to use a DSO, the client first has to find a replica part of that DSO, connect to it and then send a remote method invocation request. At first, it may seem strange to also consider the clients of an application (modeled as a DSO) as part of that application. However, as we will see in Section 6, here this makes sense because in Globe the clients of a DSO are also responsible with enforcing (part) of its security policy.

3 Trust Model

The cornerstone of the Globe trust model is that individual DSOs are fully in charge with their security policies. This means a Globe object **does not need** any external trust broker in order to run securely. Because DSOs can be massively replicated across wide-area networks, we have chosen public key cryptography as the basic cryptographic building block for implementing the DSO trust hierarchy. The alternative, namely to use only shared secret keys, has the disadvantage that we need to take special measures to reduce the number of keys, for example, by using a Key Distribution Center. Although public keys introduce their own scalability problems, such as those related to certificate revocation, we have nevertheless decided to associate public/private key pairs with all distinct Globe entities (DSOs, replicas, clients), believing that these are more easy to deploy in a large-scale system.

We require that each DSO has a public/private key pair, which we term as the **object key**. The object key acts as the ultimate source of trust for the object, and any principal that has knowledge of the object's private key can set the security policy for that object (we term such a principal the **object owner**). For a given DSO it is assumed that the object key is known by all entities (clients and replicas) part of that DSO. This can be accomplished through the use of a PKI or through some other out-of-band means.

We also associate a public/private key pair with every DSO replica and client (we call this the **replica key** and the **client key**). Permissions are then associated with these public keys. For example, in the case of a client, we associate the client's public key with the methods that client is allowed to invoke. For replicas, an example of permission that can be associated with its public key, is which of the DSO's methods the replica is allowed to handle.

For a given DSO, the set of all permissions associated with all its clients and replicas form the **security policy** for that DSO. Whenever two entities part of the same DSO interact, they first need to authenticate each other, and then check the DSO's security policy to ensure their interaction follows that policy. For example, when a client invokes a method on a replica, the replica needs to check the security policy to ensure the client is allowed to invoke the method, and the client needs to check the policy to ensure the replica is allowed to handle it.

There are many ways this security policy can be expressed; for example, one can envision a (very large) table storing the public keys of all clients and replicas and listing the permissions associated with each of them. However, such a solution clearly does not scale, since this table would have to be distributed to all entities part of the DSO. Furthermore, updating such a highly replicated table would be a daunting task.

The solution we have envisioned for Globe is inspired by the work done on Role Based Access Control [8]. The idea is to group entities (clients, replicas) with equivalent security properties into **client/replica roles**, and express a DSO's security policy based on these roles. The assumption is that the number of roles is not very large and fairly static (for a given application, one does not have to add a new class of clients every day); We also assume that the set of permissions associated with a role does not change frequently. Based on these assumptions, a security policy expressed in terms of such roles has two big advantages: it is quite compact (because the number of roles is much smaller than the number of entities that are mapped in these roles), and does not change frequently. Thus, it is scalable to distribute this security policy to all entities part of a DSO.

Besides this role-based policy description, we also need to provide the mapping of clients/replicas (identified through their public keys) to the roles they have been assigned. This is accomplished through **role certificates** - digital certificates that bind an entity public key to the role that entity has been assigned as part of the DSO's security policy. Such role certificates are issued by DSO entities (clients or replicas) that have been assigned administrative privileges. For a given DSO, we define an **administrative role** as the set of all administrative entities with equivalent security properties, which in this case means that entities in the same administrative role are allowed to issue the same types of role certificates. Each DSO is allowed to define its own administrative hierarchy; however, the DSO's owner will always be the root of such a hierarchy.

Once we introduce the client, replica and administrative roles, we can conceptually divide a DSO policy expressed in terms of these roles in three parts: the access control policy is mostly concerned with expressing permissions associated with client roles; the invocation policy is mostly concerned with expressing permissions associated with replica roles. Finally, administrative policies deal with describing how administrative roles are allowed to assign other roles. In the following sections we will look at each of these policies in detail.

4 Expressing Administrative Policies

As we explained in the previous section, role certificates are issued by clients or replicas that have been given administrative privileges as part of a given DSO. Each DSO has at least one administrative entity - the object owner - which by default has all the possible administrative privileges (it is allowed to delegate every possible role in the DSO's role hierarchy).

At first, it may seem strange to have both clients and replicas as administrators for an object, since one would usually associate a human with such a role. Administrative replicas come in handy when we deal with massively replicated

DSOs. For such DSOs, a highly dynamic pattern in client requests can be better handled by creating new replicas on the fly, in places where most of the client requests come from. In such a scenario, one client administrator can start a number of administrative replicas, and issue administrative certificates granting them the right to issue replica certificates. These administrative replicas could in turn monitor client requests and create regular replicas in places where they can better handle these requests.

The only type of privileges that can be associated to administrative roles is the right to delegate other roles. Thus, an administrative role can be fully described by listing all the roles it can delegate. An intuitive way to see a DSO's role hierarchy is as a directed graph, with each role corresponding to a node; in such a graph, an edge from node A to node B, implies that role A is an administrative role, and it has the right to delegate role B under the DSO's security policy. Based on what we discussed so far, such a graph needs to have the following properties:

– it has exactly one node of in-degree 0; this is the object owner role, which is implicitly assigned to the principal that has access to the DSO's private key.
– all nodes corresponding to administrative roles (except for the object owner role) have non-zero in-degrees and out-degrees (because an administrative role should be able to delegate at least one role).
– all nodes corresponding to client or replica roles must have a zero out-degree (because they should not be able to delegate any roles).

In addition to this, we would like our role hierarchy to be monotonic, namely a role with less privileges should never be able to issue a role with more privileges. This can be accomplished by enforcing the following extra rules on the role graph:

– the graph should not have any cycles of length greater than one edge. By allowing one-edge cycles we allow administrative roles to replicate themselves.
– the set of zero out-degree children of any node is a subset of the set of zero out-degree children of its parent. This ensures that no administrative entity can get more privileges in issuing client/replica roles by delegating itself to a lower administrative role.

Designing a role graph with the above properties is in fact equivalent to describing the DSO's administrative policy. For Globe DSOs this is done by the object owner who needs to describe the graph in a policy language. One way of un-ambiguously describing a directed graph is by describing all its edges; this can be done using a language construct of the type:

$Role_A$ **canDelegate** $Role_B$

Here, $Role_A$ and $Role_B$ are role names. A DSO's administrative policy is then fully described by a set of such statements. A policy interpreter can then process all these statements and construct the role graph. The graph is first checked to follow the monotonicity properties. After that, all nodes with non-zero in- and out-degrees are interpreted as administrative roles. All the other ones (except for the object owner role) are interpreted as either client or replica roles. How we distinguish between these two, we will describe in the following sections. Once the DSO's administrative policy has been designed and checked for monotonicity, it needs be securely distributed to all DSO's entities. This policy is then used to verify the role certificate chains that DSO entities exchange when they authenticate each other. A certificate chain is considered valid if the signature chaining is correct and the role delegation described by the chain follows a path in the role graph. Thus, each DSO entity needs to be provided with a policy engine, which, after initialized with the DSO's administrative policy can answer queries of the type:

isValidChain($certChain$**)**.

where *certChain* is a chain of digital certificates corresponding to a sequence of DSO entities (clients and replicas) delegating DSO roles to each other. The **isValidChain()** query returns $True$ if the sequence of roles in *certChain* corresponds to a valid root path in the DSO's role graph (this means the chain starts with a certificate signed with the DSO's key, and for each certificate, the role in the certificate **canDelegate** the role in the next certificate in the chain) and $False$ otherwise.

5 Expressing Access Control Policies

Besides administrative privileges, a DSO entity can also have method invocation privileges; such privileges describe which of the DSO's methods that entity is allowed to invoke. Method invocation privileges are not restricted only to the DSO's clients, replicas can also invoke methods on each other; a good example of this is replicas of a DSO that implements a master/slave replication strategy: whenever the DSO's state changes, the master replica invokes a special method *Invalidate*() on all slave replicas. In order to get the new state, the slaves then invoke *StateUpdate*() on the master. Thus, the master replica role needs to have invocation privileges on the *Invalidate*() method, while the slave replica role needs invocation privileges on the *StateUpdate*() method. However, the slave replica role should not be allowed to invoke *Invalidate*(), otherwise, a malicious slave would be able to propagate state changes (it is assumed the master is more trustworthy in this case).

A method invocation privilege can be expressed through a policy statement as the following:

Role **canInvoke** $Method$ **underConditions** *Conditions*

here, *Role* is a non-administrative role previously declared in a **canDelegate** statement; *Method* is the name of one of the DSO's methods; *Conditions* is a boolean expression that puts constraints on the way the $Method$ can be invoked by *Role*; the terms that can appear in *Conditions* are the following:

– the parameters passed to this method; in this way, *Conditions* can impose certain parameter ranges for *Role* invoking *Method*.
– external functions; these have to be separately defined so the policy engine knows how to invoke them; such external functions can impose constraints on the way *Role* can invoke *Method* based on things like the object's state, the resources available on the system running the replica, time of the day, or the location where the request originates. The only requirement here is that such external functions are synchronous - this ensures the policy engine cannot be blocked on an external function.
– additional attributes (expressed as name-value pairs and always interpreted as strings) present either in the certificate chain associated with the caller, or in other certificates provided by the caller.

A DSO's access control policy can then be fully described through a set of such **canInvoke** statements. Again it is the object owner that designs this access control policy and is responsible with securely passing it to all DSO replicas (only replicas need to do access control checks, since clients can only invoke methods, they cannot execute them). Each DSO replica then stores this policy in its policy engine, and whenever receives a request from an (already authenticated) client, it checks the request against the policy through the following query:

$$\textbf{isAllowed}(Role, Method, Parameters)$$

where *Role* is the role the client has authenticated with, *Method* is the method invoked by the client and *Parameters* represent the actual parameters. The **isAllowed()** query then returns either *True* if there is a **canInvoke** statement that allows the invocation of *Method* with the given parameters by *Role*, considering all external conditions that may apply, or *False* otherwise.

6 Expressing Method Execution Policies

The last type of privileges that can be associated with a DSO entity are method execution privileges. The need for expressing such types of privileges is a direct consequence of the fact that Globe DSOs can be replicated over hosts of various degree of trustworthiness. In such a setting, it makes sense for the owner of a DSO to restrict the execution of the most security sensitive operations to the replicas that run on the most trustworthy servers (at least from from the owner's perspective). The owner does this by indicating to clients which replicas can be contacted for each type of method invocation. This type of "reverse access control" can also be seen as a way to label quality of service and/or trust level of replicas to clients.

While the task of enforcing the access control policy relies with the replicas, it is clients that enforce this method execution policy. It is assumed that a client that has been allowed to invoke a method under the DSO's access control policy has all the interest to ensure that invocation is sent to a replica that is allowed to execute it under the DSO's method execution policy.

A method execution privilege can be expressed through a policy statement as the following:

RoleExpr **canExecute** *Method* **underConditions** *Conditions*

Here, *RoleExpr* is an expression of the form:

```
<RoleExpr>:: <PrimaryTarget> | <RoleExpr> "&&" <SecondaryTarget>;
<PrimaryTarget>:: <RoleGroup>;
<RoleGroup>:: <Role> || "Traceable(" <Role> ")" ||
          <PositiveInteger> "*" <Role> ||
          <PositiveInteger> "* Traceable(" <Name> ")";
<SecondaryTarget>:: <RoleGroup> ||
           <PositiveInteger> "\%" <Role>;
```

Such an expression is used to describes a group of replicas in possibly different roles, and the way these replicas need to be contacted by a client that wants to invoke *Method* on the DSO. The reason why more than one replica may need to be contacted when invoking a method is Byzantine fault tolerance; here we envision three basic techniques that can be used to achieve such Byzantine fault tolerance for a DSO consisting of many replicas of various degrees of trustworthiness:

- **replicated invocation**: the same method is invoked on a number of (less trusted) replicas. The result is accepted only when a certain number of them agree on the return value. Thus, in order to make a client accept an incorrect result, a number of malicious replicas would have to collude.
- **traceable results**: a (less trusted) replica executing a method has to sign (with its private key) the invocation request and the return value. The client can then forward this traceable request-result pair to a more trustworthy replica that may audit the result (by executing the method again and comparing the result). Thus, less trusted replicas acting maliciously can be traced and eventually excluded from the DSO.
- **double-checking**: the client first invokes the method on a less trusted replica, and then may double-check the result with a more trustworthy replica. However, in order to avoid overloading the trusted replica, the double-checking is done statistically (for each request there's only a small probability that request will be double-checked).

For example, the expression $3 * Role_1 + 2 * Role_2$ specifies that 3 replicas in role $Role_1$ and 2 replicas in role $Role_2$ have to agree on the result of invoking *Method* before the client accepts the result. As another example, the expression $Traceable(Role_1) + 5\% Role_2$ specifies that *Method* can be invoked on a replica in role $Role_1$, which has to sign the result; furthermore, the result should be double-checked with a replica in role $Role_2$ with a probability of 0.05.

It is important to understand that such mechanisms for achieving Byzantine fault tolerance work only with methods that are idempotent (invoking them multiple times has the same effect as invoking them once). It is the DSO owner's responsibility to ensure that only such idempotent methods are marked as "multiple invocation" in the DSO security policy file.

The other elements in the **canExecute** statement have the same meaning as in the **canInvoke** statement: *Method* is the name of one of the DSO's methods while *Conditions* is a boolean expression that puts constraints on the way the *Method* can be executed by *RoleExpr*; again, conditions can be placed on parameter values (only certain ranges allowed), possibly time of invocation.

A DSO's method execution policy can then be fully described through a set of such **canExecute** statements. Again, it is the object owner that designs this policy and is responsible with securely passing it to all DSO clients and replicas (remember from the previous section that both clients and replicas can invoke the DSO's methods). Each DSO entity then stores this policy in its policy engine, and consults it before issuing a method invocation request by making the following query:

$$\textbf{whoCanDoIt}(Method, Parameters)$$

where *Method* is the method to be invoked and *Parameters* represent the actual parameters. The policy engine then searches through all the **canExecute** statements in the policy and stops whenever it finds one that matches *Method*, and for which the *Parameters* value satisfy the **withConditions** part of the statement. It then returns the *RoleExpr* from the selected statement. The entity that does the invocation then needs to locate replicas in the roles described in *RoleExpr* and send the method invocation request to each of them.

7 Implementation Overview

We have implemented a policy engine based on the design outlined in the previous sections; the idea is to integrate this engine with the Globe middleware.

The policy engine has two separate parts. One part deals with certificate chain validation, the other part with the security checks needed for DSO method invocation.

The certificate chain validation engine has a generic core which is based on the semantic model of trust management engines (TME) defined in [12]. This generic core is an implementation of the proof-check method. The core can be used for any trust management system which can be expressed with the semantic model. This approach allows easy experimentation with new kinds of certificate languages, and new ways to delegate authorizations to principals.

An overall policy constructed by the DSO owner defines the role-hierarchy, and permissions. The implemented trust management system uses the role-hierarchy to check that principals only delegate lower or the same role to other principals.

The other part of the policy engine is responsible with interpreting the **isAllowed** and **canExecute** statements related to method invocation and method execution.

As we described in the previous two sections, permissions contain a condition expression, which allows the DSO owner to express restrictions on the actual parameters passed. The syntax of the condition language is based on the syntax of Java expressions. To be usable for sound access restrictions, the expression language uses a very strict static type checking algorithm, which makes sure that the evaluation of the conditions can never get stuck due to illicitly typed expressions.

The expression language has six base types: *int, long, float, double, char, boolean*, and *string.* When conditions are placed on DSO method parameters declared as complex Java classes, the class-name needs to be declared as a new foreign type and the beginning of the policy file. These foreign types can only be compared using the $=$, and $\neq$ operators. When the foreign class also implements the Jave *Comparable* interface the $<$, $>$, $\geq$, and $\leq$ operators can also be used. To give the policy writer even more control over values of foreign types, it is also possible to define regular Java functions operating on such values. The strict static type checking algorithm makes sure that these function can only be called with parameters of exactly the same types as used in their definitions. This makes the invocation of the methods safe, because it does not allow one to pass parameters constructed from sub-classes not known at the time when the policy is written.

8 Related Work

In this section we look at a number of other trust/certificate management systems: KeyNote [2], RTML [7], XACL [6], X.509 [9], and PGP [3], and compare them with our solution.

The KeyNote system [2] provides a simple language for describing and implementing security policies, and digitally signed credentials. KeyNote unifies the notion of a security policy with that of a credential. This idea is powerful, and allows for sophisticated setups. The drawback of this solution is that it is not possible to just look at the policy to audit the security setup.

KeyNote is more general than the policy language we propose, so it should be possible to express any Globe policy in KeyNote. However, our policy language makes it very easy to express policy constructs related to byzantine fault tolerance and replicated invocation and auditing, which become more complicated with Keynote. Another difference is that we allow tight integration between the policy engine and the secured application. Our policy language also has the possibility to define and call foreign methods from the application. This allows an application programmer to make policy decisions based on application specific values, something which is not possible with KeyNote.

The Role-based Trust-management Markup Language (RTML [7]) has much in common with our solution. It is also role based, and is embedded in a programming language. Just like our solution, the credentials and the policies are encoded in XML. The main difference between our solution and RTML is that we do not allow arbitrary language constructions in credentials. Our credentials can only contain constructs which deal with the delegation of roles, making them simpler. RTML also does not support a tight integration between the security system, and the application.

The XML Access Control Language (XACL see [6]) has a completely different goal as our system. XACL was designed to create access policies for XML documents, while our system was designed to create policies for a distributed object system.

Pretty Good Privacy (PGP [3]) and X.509 [9] are well known certificate management systems. Both of them focus entirely on authentication, while our solution focuses on authorization. X.509 certificates bind so called Distinguished Names (DN) to public keys, while PGP certificates bind e-mail addresses to public keys. On the other hand, for the solution we propose, we bind roles to public keys associated with DSO entities. The main difference in this approach is that all possible roles are known in advance. This is not the case for DNs, and e-mail addresses. Both systems focus only on the authentication problem, and leave the authorization problem entirely to the application programmer.

9 Conclusion

In this paper we have described the design and implementation of a policy engine for enforcing security policies for distributed object applications. In our design, we explicitly take into account object replication, which introduces specific requirements, such as the need for policy mechanisms to express different levels of trust one wants to put on different object replicas. Another distinctive feature of our approach is the aim to provide our policy engine at the middleware level; as a result, our policy language has features that bring it closer to an actual programming language (Java), which should make easier to integrate it with applications. The implementation of Globe can be found at http://www.cs.vu.nl/pub/globe/ and the policy engine at http://www.cs.vu.nl/~bpopescu/globeSecurity.html.

References

1. The Common Object Request Broker: Architecture and Specification. www.omg.org, Oct 2000. Document Formal.
2. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust-Management System, Version 2. RFC 2704, September 1999.
3. J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP Message Format. RFC 2440, November 1998.
4. G. Eddon and H. Eddon. *Inside Distibuted COM.* Microsoft Press, Redmond, WA, 1998.
5. A. Grimsaw and W. Wulf. Legion - a view from 50000 feet. In *Fifth IEEE Int'l Symp. on High Performance Distr. Computing.* IEEE Computer Society Press, Aug 1996.
6. M. Kudo and S. Hada. XML Document Security based on Provisional Authorization. In *Proc. 7th ACM Conf. on Comp. and Comm. Security*, November 2000.

7. N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a Role-based Trust Management Framework. In *Proc. IEEE Symp. on Security and Privacy, Oakland*, May 2002.
8. R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–48, Febr. 1996.
9. I. T. Union. Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks, March 2000.
10. M. van Steen, F. Hauck, P. Homburg, and A. Tanenbaum. Locating Objects in Wide-Area Systems. *IEEE Communications*, pages 104–109, January 1998.
11. M. van Steen, P. Homburg, and A. Tanenbaum. Globe: A Wide-Area Distributed System. *IEEE Concurrency*, pages 70–78, January-March 1999.
12. S. Weeks. Understanding Trust Management Systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 94–105, May 2001.

Enforcing Security Policies for Distributed Objects Applications
(Transcript of Discussion)

Bogdan C. Popescu

Vrije Universiteit, Amsterdam

Birgit Pfitzmann: When you say you have an untrusted replica, you're not thinking of secrets, only of integrity?

Reply: Yes.

Simon Foley: If you're buying certificates that give permissions to clients how do you name the objects?

Reply: There's an actual name that is human readable, which is bound to an object ID. The first line of each object has to be trusted, and you as the user have to decide the object that you worry about. These are just self-certified, basically the object ID is a hash of the object key.

Simon Foley: So if there are two objects, say two entries in a database, and I want to give somebody permission to access just one of them, do you have a fine line of granularity where you can authorise people for particular, individual objects of a certain class?

Reply: If you resemble a database, then yes, that would be part of the object-state based on the record. The object would be the menu of papers, and your record would be part of the state. There are certain methods for this. Let's say you have an enquiry requesting the record of patient X. You can decide whether you want to give that client these records, depending on who the client is and what records he has requested.

Simon Foley: It's done purely on the credentials and certificates that are offered by the client?

Reply: Yes.

Pekka Nikander: How do you synchronise the replicas in practice? I get the impression that one replica calls to the other ones, and they all do the same computation.

Reply: Basically you give them a number of built-in replication algorithms and when you need a service you just select one. Depending on what you select, there is a standard interface between reaching the thing you want and the functionality you want to replicate, and you just plug in whatever you want.

Virgil Gligor: I have a question about the policy that you support. If I find out what accesses you have right now to what set of objects, can I then determine, for example, that that set is different from the set that Bruno has access to right now?

Reply: Well if you know my credentials and if you know the client certificates, then yes.

Virgil Gligor: How do I go about doing it? Suppose I'm a client and I want to find out whether your current accesses are distinct and disjoined from Bruno's, so that it's all per user, or per subject. You have to have a map on clients and also a map on the roles and specific objects, not just policy statements, for all these conditions.

Reply: Well that's two questions. How do you assign roles in the first place? What we do is just specify what roles we can do, we have a lot of simple scenarios for this. I'm a user and I go to a global website, and I just register. I create something and I get back a user certificate. I find some public key to set up permission that I have with regard to that, and then use it to get that thing, and that's it.

Virgil Gligor: So you really don't have any way to trace what specific set of objects you have access to now?

Reply: Yes, everything is more or less based on the objects sent to this node.

Virgil Gligor: OK, but looking at the subject, as the user, you cannot tell what accesses it has now, and whether those accesses are different from anybody else's. One of the fundamental uses for roles is separation of duty. Very simplistically this says these users should have completely separate accesses to completely separate sets of objects, and that's not generally easy to do in most areas of access.

Reply: Yes, I think maybe we used the role in a different way to that. Basically the role is just a set of permissions.

Bruce Christianson: But Virgil's point is that sometimes a role needs prohibitions as well as permissions. If you have a role already, can't you prohibit it from ever using the permissions?

Bruno Crispo: Taking care to define here that it is the usual roles; if role A is made separate from role B.

Reply: How you make sure that once somebody has got role A, he's not going to get role B, would be an applications system responsibility.

Virgil Gligor: No, I mean something less. It should be a fairly trivial question, but in practice it's very hard. The simple question is how I find out all objects that I have access to in the system. Generally this is hard, although it may be easy in your system. It's hard because in order to find out what the number of accesses assigned in such a system are, you'd have to search the denied objects space. Role-based access in those systems has a mapping between a role and the specific objects accessible from that role. If you have that, then you can enforce separation of different policies. If you don't, and you ask the question, "What are my accesses right now?" then you turn on a search engine. They tried that at Maryland, to find out what accesses are, and we had to stop the search after two days.

Partha Das-Chowdhury: So for a single object how do you implement different permission sets for a single object? How do you go on to implement different conditions, and separate what different roles have access to the same object as different functions?

Reply: You just constrain the roles.

Partha Das-Chowdhury: You can do this for the same object?

Bruno Crispo: Yes, because a role is a set permission here.

Reply: You just say that for role A you shouldn't be able to access anything that role B can access.

Pekka Nikander: One of the problems that I have been thinking about for a while now is the boundary between policies and applications.

Reply: There seems to be a gap.

Pekka Nikander: Pretty often you end up having more or less the same logic both on the application side and in the policy. If you think about this from a computer science point of view, it's not very wise to have the same things specified two times in different ways. You sooner or later end up with problems if you do. I think there are a few cases where you really want to have policies. If you want to do a security analysis of the system, it's easier to do with policies, but with some kind of limited security, I don't know. On the other hand, deciding where the policies should start, what should be application specific and what you should be able to do with policies, seems like a hard question.

Virgil Gligor: The real policy tends to be built into the application, because application designers know the requirements of the environment that's used, not this agnostic operating system.

Regular SPKI

Mads Dam[⋆]

LECS/IMIT, Royal Institute of Technology (KTH),
KTH Electrum 229, SE–164 40 Kista, Sweden
mfd@kth.se

Abstract. SPKI is a certificate-based framework for authorisation in distributed systems. The SPKI framework is extended by an iteration construct, essentially Kleene star, to express constraints on delegation chains. Other possible applications, not explored in the paper, include multidomain network routing path constraints. The main decision problems for the extended language are shown to correspond to regular language membership and containment respectively. To support an efficient decision algorithm in both cases we give a sound and complete inference system for a fragment of the language which is decidable in polynomial time. We finally show how to use the extended syntax to represent constrained delegation in SPKI.

1 Introduction

SPKI (Simple Public-Key Infrastructure) [EFL+99] is a framework for authorisation based on the idea that authorisations and names are bound to public keys by signed certificates. SPKI uses LISP-like S-expressions [Riv97] to express authorisations. As an example, the S-expression

```
(object document (attributes (name doc1) (loc EU))
                    (op read) (subject (users orgA)))       (1)
```

might express the authority of orgA users to read objects of type document which have name doc1 and are located in the EU[1].

The treatment of delegation in SPKI is rudimentary. Authorisations are equipped with a flag which, when set, enable holders of authorisations unconstrained delegation rights for that authority. In [BDF02] we argued that such unconstrained delegation right are not always desirable, and we proposed a mechanism, *constrained delegation*, which uses regular expressions to control the way authority is propagated between principals along a delegation chain. It is the objective of the present paper to examine how the constrained delegation approach can be handled within the SPKI framework.

⋆ Work supported by Vinnova, the Swedish Agency for Innovation Systems, project "policy-based network management".

[1] In fact, proper SPKI treats such authorisations using a specialised 5-tuple syntax. This, though, is of little consequence for the present discussion.

In [BDF02], delegation chain constraints are expressed by regular expressions. As a simple example, the chain constraint `admin`*`users` expresses the set of delegation chains which have a prefix in `admin` and terminate in `users`. This chain constraint would capture the situation where a group of administrators are given authority to create some management structure within the group for administering, say, the access rights of users.

Our proposal for handling this within the SPKI framework is to extend the SPKI syntax by a new primitive, `(* path ...)`, denoting, roughly, Kleene star. To see how this might be done, consider the following S-expression:

```
(object document (attributes (name doc1) (loc EU))
              (op read) (delegation (* path (admin)) (users))  (2)
```

In this example[2] the `subject` component of (1) is replaced by a component `delegation` that represents delegation chain constraints. Here delegation is licensed through `admin` in any number of steps, but the read permissions that are ultimately granted must be under users control.

Built in to SPKI is the idea of possibly refining, at each step of delegation, the authority received in the previous step. The basic mechanism for refinement is to recursively append further constraints to each list. In the presence of `path` expressions, unfolding must be added to this mechanism. As an example, (2) could be used to justify:

```
(object document (attributes (name doc1)
   (loc EU France)) (op read) (delegation (admin domain1)
                           (* path (admin domain2)) (users)))
```

restricting documents to the location France, and allowing administrators in domain1 to delegate through administrators in domain2 an arbitrary number of steps. In turn, this S-expression can be further refined, and the delegated authority discharged, to ultimately result in authorisations similar to (1) above. Below, in section 7, we give an example showing this process of delegation in more detail.

In the paper we discuss mechanisms which could be used for such a regular language extension to SPKI. At the basic level two extensions are required: S-expression concatenation, and Kleene star (`path`). We show how this can be done, and how basic questions concerning S-expressions can then be reduced to corresponding questions for regular languages. This is sufficient to efficiently answer simple authorisation queries. Refinement, however, corresponding to regular language containment, will require exponential time. Chain discovery, then, will also be exponential. To address this we introduce a restricted syntax for which refinement is decidable in polynomial time. We present an inference system for entailment (containment) for this fragment which is based on fixed point induction. The inference system is shown to be sound and complete. The completeness

[2] Which is actually wrong: Using the notation introduced in the paper, the delegation component should properly be written `(delegation (* path (admin));((users)))` where `;` is list concatenation.

proof, in particular, allows the decision procedure to be extracted. We then show how to use the regular expression facility to represent constrained delegation, and how name resolution and delegation can be handled in this framework.

2 S-Expressions

A SPKI expression denotes a set of S-expressions [Riv97]. Let A be a denumerable set of "atomic" elements ranged over by a of one or several data types such as strings, octets, or integers. The set S of *S-expressions*, ranged over by s, is determined by the following BNF style grammar:

$$s \ ::= \ a \ \mid \ (s_1 \ \cdots \ s_n)$$

where $n \geq 0$. In other words, S is the tuple algebra over A, and, e.g., $(a_1 \ a_2)$ is the tuple with left hand component $a_1 \in A$ and right hand component $a_2 \in A$. To account for authorisation we introduce a partial ordering $\leq$ on S. Let $s_1, s_2 \in S$.

1. If $s_1 \in A$ or $s_2 \in A$ then $s_1 \leq s_2$ if and only if $s_1 = s_2$.
2. If $s_1 = (s_{1,1} \ \cdots \ s_{1,m}) \in S$ and $s_2 = (s_{2,1} \ \ldots \ s_{2,n}) \in S$, then $s_1 \leq s_2$ if and only if $m \geq n$ and $s_{1,i} \leq s_{2,i}$ for $i = 1, \ldots, n$.

That is, $s_1 \leq s_2$ just in case s_1 is more specific (more constrained, or authorised by) s_2, and that the process of becoming "more specific" is by appending more information to the end of sublists.
Example 1.

```
(object document (attributes (name doc1) (loc EU France)
                                                (author NN)))
  ≤ (object document (attributes (name doc1) (loc EU)))
```

If s represents a policy authorised for a principal x, and s' represents a request of x such that $s' \leq s$, then and only then should the request s' be granted. In the SPKI literature this idea is usually treated not using the partial order $\leq$, but through the associated operation of glb, or intersection, such that $s' \leq s$ iff $s \cap s' = s'$ (cf. [BD02]).

In SPKI, S-expressions are usually required to begin with an atom. The leading atom, which we refer to as a "tag" below, serves as a type indicator. That is, the type of an element s_i of a list $(a \ s_1 \ \cdots \ s_n)$ will in standard SPKI be determined by the tag a and the position i. These type associations are determined by some external means; here it suffices to assume some fixed such binding, when it applies.

3 Syntax of Regular SPKI Expressions

S-expressions provide a basic syntax for expressing constrained authorisation, but the notation is not really versatile enough for practical use. For this reason the SPKI authorisation syntax extends S-expressions with the following features:

- Constructs to denote sets of atoms (the SPKI prefix and range constructs). These constructs are left out of the present treatment, but they can be added without substantial complications.
- Constructs to denote sets of lists (the wildcard `(*)` and the `* set` construct).

To this constructs we add the following two:

- An iterator, `* path`, basically Kleene star.
- Composition of S-expressions, denoted by semicolon.

Definition 1 (Regular SPKI Expressions). The set $\mathcal{S}$ of *regular SPKI expressions*, ranged over by σ, is determined as follows:

$$\sigma ::= \texttt{(*)} \mid a \mid \texttt{(}\sigma_1 \cdots \sigma_n\texttt{)} \mid \sigma_1\texttt{;}\sigma_2 \mid \\ \texttt{(* set } \sigma_1 \cdots \sigma_n\texttt{)} \mid \texttt{(* path } \sigma\texttt{)}$$

where $a \in A$, $b \in B$, and $n \geq 0$.

Essentially, SPKI expressions can be regarded as abbreviations of sets of S-expressions. This is brought out by the semantics, fig. 1.

$$\begin{aligned}
&\|\texttt{(*)}\| = S \\
&\|a\| = \{a\} \quad \text{for all } a \in A \\
&\|\texttt{(}\sigma_1 \cdots \sigma_n\texttt{)}\| = \{\texttt{(}s_1 \cdots s_n\texttt{)} \mid \forall i : 1 \ldots n . s_i \in \|\sigma_i\|\} \\
&\|\sigma_1\texttt{;}\sigma_2\| = \\
&\quad \{\texttt{(}s_1\texttt{,}\ldots\texttt{,}s_n\texttt{)} \mid \exists i : 1 \ldots n . \texttt{(}s_1 \cdots s_i\texttt{)} \in \|\sigma_1\|, \texttt{(}s_{i+1} \cdots s_n\texttt{)} \in \|\sigma_2\|\} \\
&\|\texttt{(* set } \sigma_1 \cdots \sigma_m\texttt{)}\| = \|\sigma_1\| \cup \ldots \cup \|\sigma_m\| \\
&\|\texttt{(* path } \sigma\texttt{)}\| = \{\texttt{(}s_1 \cdots s_n\texttt{)} \mid \forall i : 1 \ldots n . s_i \in \|\sigma\|\}
\end{aligned}$$

Fig. 1. Regular SPKI expressions, semantics

Let $\sigma_1 \cong \sigma_2$ iff $\|\sigma_1\| = \|\sigma_2\|$. General lists are definable in terms of composition (`;`) and singleton lists, since

$$\texttt{(}\sigma_1 \cdots \sigma_n\texttt{)} \cong \texttt{(}\sigma_1\texttt{)} \;\texttt{;}\; \cdots \;\texttt{;}\; \texttt{(}\sigma_n\texttt{)}. \tag{3}$$

Definability the other direction does not hold. This is easily seen, as the list constructor strictly increases depth of nesting which composition does not. As a consequence we only need to consider the empty list (`()`) and singletons (`(`σ`)`) as primitive. This is exploited heavily below. It is important, however, to bear equation (3) in mind, since it will allow the composition operator to be eliminated in favour of the more standard list syntax in all "reasonable" cases, except those that specifically involve path expressions.

Example 2. Let $\sigma = \texttt{(}a\ \texttt{(* path}\ b\texttt{)}\ c\texttt{)}$. We compute:

$$\begin{aligned}\|\sigma\| &= \{\texttt{(}s_1\ s_2\ s_3\texttt{)} \mid s_1 \in \|a\|, s_2 \in \|\texttt{(* path}\ b\texttt{)}\|, s_3 \in \|c\|\} \\ &= \{\texttt{(}a\ \texttt{(}s_1\ \cdots\ s_n\texttt{)}\ c\texttt{)} \mid \forall i.s_i \in \|b\|\} \\ &= \{\texttt{(}a\ \texttt{(}b\ \cdots\ b\texttt{)}\ c\texttt{)}\}\end{aligned}$$

Compare with $\sigma' = \texttt{(}a\texttt{);(* path}\ b\texttt{);(}c\texttt{)}$:

$$\begin{aligned}\|\sigma'\| = \{\texttt{(}s_1\ \cdots\ s_n\texttt{)} \mid n \geq 2, s_1 \in \|a\|, \\ \texttt{(}s_2\texttt{,}\ldots\texttt{,}s_{n-1}\texttt{)} \in \|\texttt{(* path}\ b\texttt{)}\|, s_n \in \|c\|\} \\ = \{\texttt{(}a\ s_2\ \cdots\ s_{n-1}\ c\texttt{)} \mid n \geq 2, \forall i : 1 \ldots n.s_i \in \|b\|\} \\ = \{\texttt{(}a\ b\ \cdots\ b\ c\texttt{)}\} \quad .\end{aligned}$$

The semantics of fig. 1 is not the only one possible. A different semantics, $\|\sigma\|'$, would introduce ; as concatenation of s-expression lists, and then define $\|\sigma\|'$ as in fig. 1 except that:

$$\|\texttt{(* path}\ \sigma\texttt{)}\|' = \{s_1\texttt{;}\cdots\texttt{;}s_n \mid \forall i : 1 \leq i \leq n.s_i \in \|\sigma\|\}. \tag{4}$$

The two semantics are easily related, since obviously

$$\|\texttt{(* path}\ \sigma\texttt{)}\| = \|\texttt{(* path (}\sigma\texttt{))}\|'.$$

We prefer the semantics of fig. 1 since the notation in the latter case seems to contribute not much more than the need to add a few extra parentheses.

The partial ordering $\leq$ on S-expressions is extended to regular SPKI expressions in the following way:

$$\sigma_1 \leq \sigma_2 \text{ iff } \forall s_1 \in \|\sigma_1\|.\exists s_2 \in \|\sigma_2\|.s_1 \leq s_2 \tag{5}$$

To see that this definition makes sense, let

$$\downarrow\sigma = \{s \mid \exists s' \in \|\sigma\|.s \leq s'\} \quad .$$

The set $\downarrow\sigma$ is the "downwards closure" of $\|\sigma\|$ according to $\leq$. In the intuitive sense of section 2 it is the set of all S-expressions which are authorised by some element in $\|\sigma\|$. The following is standard:

Proposition 1. $\sigma_1 \leq \sigma_2$ *iff* $\downarrow\sigma_1 \subseteq \downarrow\sigma_2$ $\square$

In other words, $\sigma_1 \leq \sigma_2$ just in case every S-expression authorised by σ_2 is also authorised by σ_2.

Two problems are of central interest:

1. P1, membership: Given a request formulated as an S-expression s and an authorisation policy σ, is $s \in \downarrow\sigma$?
2. P2, entailment: Given authorisation policies σ_1, σ_2, is σ_1 authorised by σ_2 (i.e. does $\sigma_1 \leq \sigma_2$ hold)?

It is not very difficult to cast these problems in terms of regular languages. Define an ancillary ordering on S by $s_1 \sqsubseteq s_2$ iff either $\exists a \in A.s_1 = a = s_2$ or else $s_1 = (s_{1,1} \ \cdots \ s_{1,m})$, $s_2 = (s_{2,1} \ \cdots \ s_{2,m})$, and $s_{1,i} \leq s_{2,i}$ for all $i : 1 \leq i \leq m$. That is, $\sqsubseteq$ acts just as $\leq$ except that appending rightmost list elements to the outermost list is not permitted. Let then

$$\Downarrow\sigma = \{s \mid \exists s' \in \|\sigma\|.s \sqsubseteq s'\}$$

Now, consider S-expressions as given in the form $(s_1)\,;\cdots;(s_m)$ instead of $(s_1 \ \cdots \ s_m)$, and consider $(;)$ and Kleene star closure $(\cdot)^*$ as operations on sets C of S-expressions as follows:

$$\begin{aligned} C_1\,;C_2 &= \{s_1\,;s_2 \mid s_1 \in C_1, s_2 \in C_2\} \\ C^* &= \{s_1\,;\cdots;s_n \mid \forall i : 1 \leq i \leq n.s_i \in C\} \end{aligned}$$

We obtain the following characterisation of closure sets (proof is given in the appendix):

Proposition 2.

1. $\Downarrow\texttt{(*)} = \|\texttt{(*)}\|$
2. $\Downarrow\texttt{()} = \texttt{()}$
3. $\Downarrow(\sigma) = \{(s) \mid s \in \downarrow\sigma\}$
4. $\Downarrow\sigma_1\,;\sigma_2 = \Downarrow\sigma_1\,;\Downarrow\sigma_2$
5. $\Downarrow\texttt{(* set } \sigma_1 \ \cdots \ \sigma_m\texttt{)} = \Downarrow\sigma_1 \cup \cdots \cup \Downarrow\sigma_m$
6. $\Downarrow\texttt{(* path } \sigma\texttt{)} = (\Downarrow\sigma)^*$
7. $\downarrow\sigma = \Downarrow\sigma\,;\Sigma^*$ □

This proposition provides a direct representation of regular SPKI expressions as "ordinary" regular expressions, and so we obtain:

- $\downarrow\sigma$ is a regular language
- P1 is regular language membership. Thus P1 is decidable in time $\mathcal{O}(|s||\sigma|)$. Moreover, since there is a trivial logspace reduction of regular language membership to P1, P1 is also complete for NLOGSPACE.
- P2 is regular language containment. This follows directly from fact 1. Thus P2 is in EXPTIME and complete for PSPACE.

4 EOL Markers

There is a basic tension between the introduction of path expressions and the basic S-expression syntax. In particular, S-expressions are intended to be positional in the sense explained in section 2. But this positionality breaks down in the context of path expressions. Consider the following example:

$$\sigma = \texttt{(mysequence);((start));(* path (hop));((end))} \qquad (6)$$

An S-expression in $\downarrow\sigma$ can have a shape like

$$s = \texttt{(mysequence (start here) (hop there) (hop and-there)} \\ \texttt{(end over-here) (unintended bit)).} \qquad (7)$$

The SPKI authorisation discipline will admit s as authorised by σ, since the extra component `(unintended bit)` is appended to the right of the list and so just represents one further constraint. But in the context of path expressions this is counter-intuitive, since we may not have a preconceived idea of what the last element of a path is, and so we may not know whether `(end over-here)` or `(Unintended bit)` represents that element. For instance, were the `((end))` item to be missing from σ, an attacker could insert new hops at the end of σ at will. The example also gives away the solution: Simply assume elements with some given tag to represent the end of the list. Any application-dependent choice will do, but we may also introduce a general-purpose atom `EOL` to represent the end of the list. In this manner we will represent σ as

$$\sigma' = \texttt{(mysequence);((start));(* path (hop));((end));(EOL).} \qquad (8)$$

Observe that the addition of the `EOL` atom does not interfere with the semantics in any way.

5 An Efficient Syntax for Entailment

Through the characterisation of SPKI expression in terms of regular languages we obtain a reasonably efficient procedure for deciding the problem P1, is a given S-expression s authorised by the regular SPKI expression σ. The problem P2, however, remains intractable. It may be argued, as is sometimes done in the SPKI literature, that requests will in practice not need to involve the problematic constructions (in the absence of path expressions this means set expressions), but a closer examination of this issue reveals this to be false in many applications (cf. [BD02] for a brief discussion). At any rate the entailment problem is in our view of independent interest, for instance to allow users to efficiently determine the effects of their policy decisions. To address this problem we introduce in this section a restricted syntax for which an efficient decision procedure also for P2 is possible.

In [BD02] we addressed this issue for the basic SPKI authorisation syntax, and obtained an $n \log n$ asymptotic complexity for P2 in this case. The idea was to restrict occurrences of `set` expressions to those of the form

$$\sigma = \texttt{(* set } a_1\texttt{;}\sigma_1 \cdots a_n\texttt{;}\sigma_n\texttt{)}$$

where all a_i are required to be distinct atoms. This allows queries of the form $a;\sigma' \leq \sigma$ to be directly reduced to the query $\sigma' \leq \sigma_i$ where $a = a_i$, if such an a_i exists, and if it does not, the query is rejected. This syntactical restriction is just a formalisation of existing SPKI practice (to the extent such a thing exists): It does not in any way reduce the expressiveness of the basic SPKI authorisation syntax.

To extend this approach also to path expressions the idea is simply to tag path expressions as well as sets in such a manner that it becomes immediate how to match a path expression with its unfoldings. For instance, the presence of the `hop` atom makes it trivial to determine that s in (7) is authorised by σ of (6), as is the regular SPKI expression

```
(mysequence);((start));((hop));(* path (hop));((end))      (9)
```

whereas an expression such as

```
(mysequence);((start));(* path (hop));(* path (hop));((end)) (10)
```

would be more difficult to accomodate in principle. This solution we propose is to replace uniqueness at the level of tags with uniqueness at the level of initial segments, as in the following expression:

```
(mysequence);((start));(* path (hop domain1));
          (* path (hop domain2));((end)) .                   (11)
```

We proceed to introduce the restricted syntax which makes such a tagging discipline enforceable.

Definition 2 (Restricted Expressions). The set R of restricted expressions, ranged over by r, is given as follows:

$$\begin{array}{rcl} r & ::= & (a)\texttt{;}p \mid \texttt{(* set } r^{a_1} \cdots r^{a_m}\texttt{)} \\ r^a & ::= & (a)\texttt{;}p \\ p & ::= & \texttt{()} \mid q\texttt{;}p \\ q & ::= & (a) \mid (r) \mid \texttt{(* path } r\texttt{)} \end{array}$$

where $a \in A$, $m \geq 1$, and all a_i, $1 \leq i \leq m$, are distinct.

We generally let (a) abbreviate $(a)\texttt{;()}$.

Example 3. Keep in mind the definition of list expressions, def. 3. The following regular SPKI expressions are restricted:

- $(a\ (b\ c)) = (a)\texttt{;}((b)\texttt{;}(c))$
- $\texttt{(* set }(a\ \texttt{foo})\ (b)\texttt{;(bar))} = \texttt{(* set }(a)\texttt{;(foo) }(b)\texttt{;(bar))}$
- $(a)\texttt{;(* path }(b)\texttt{;(* path }(c)\texttt{))}$

The following regular SPKI expressions are *not* restricted:

- a, b, `(*)`
- $((a\ b)) = ((a)\texttt{;}(b))$
- $\texttt{(* set }(a\ b)\ (a\ c)\texttt{)} = \texttt{(* set }(a)\texttt{;}(b)\ (a)\texttt{;}(c)\texttt{)}$
- $(a\ \texttt{(* path }b\texttt{)}) = (a)\texttt{;((* path }b\texttt{))}$
- $(a)\texttt{;(* path (* path }b\texttt{))}$

The function *tags* computes the set of tags of expressions r and q respectively:

– $\mathit{tags}\ (\texttt{(}a_1\texttt{)}\,\texttt{;}\cdots\texttt{;}\,\texttt{(}a_m\texttt{)}\,\texttt{;}q_1\texttt{;}\ldots\texttt{;}q_n) = \{\texttt{(}a_1 \ \cdots \ a_m\texttt{)}\}$, where q_1 is not of the shape $\texttt{(}a\texttt{)}$ for any a
– $\mathit{tags}\ (\texttt{(* set } r^{a_1}\texttt{,}\ldots\texttt{,}r^{a_m}\texttt{)}) = \mathit{tags}\ (r^{a_1}) \cup \cdots \cup r^{a_m}$
– $\mathit{tags}\ (\texttt{(}\texttt{(}a\texttt{)}\texttt{)}) = \emptyset$
– $\mathit{tags}\ (\texttt{(}r\texttt{)}) = \mathit{tags}\ (r)$
– $\mathit{tags}\ (\texttt{(* path r)}) = \mathit{tags}\ (r)$.

Definition 3 (Well-formed Restricted Expressions). The restricted expression r is *well-formed* if whenever r contains a subexpression of the shape $r' = \texttt{(}a\texttt{)}\,\texttt{;}q_1\texttt{;}\cdots\texttt{;}q_n$ and q_i has the shape $\texttt{(* path } r''\texttt{)}$ then for all $j > i$,

$$\mathit{tags}\ (q_j) \cap \mathit{tags}\ (r'') = \emptyset.$$

Example 4. The expressions (9) and (11) are well-formed. The expression (10) is ill-formed, as is the expression

$$\texttt{(mysequence);((start));(* path (hop));((hop));((end))}. \qquad (12)$$

6 Inference System

We proceed to give, in fig. 2, an inference system for proving entailments of the form $r_1 \preceq r_2$, intended as syntactical correlates of the entailment relation $r_1 \leq r_2$. Call an expression e of one of the forms $r_1 \preceq r_2$ or $p_1 \preceq p_2$ an *entailment expression.* Judgments have the shape $\Gamma \vdash e$ where e is an entailment expression and Γ is a set of entailment expressions. The proof system implements a form of fixed point induction, in the style of Kozen [Koz83]. It is designed to be used in a bottom-up fashion, and can in fact be read just as a logic program. To show the proof system in action we give a couple of example derivations.

Proposition 3. *The following entailments are derivable:*

1. $\vdash \texttt{(* path (}a\ b\texttt{))} \preceq \texttt{(* path (}a\texttt{))}$.
2. $\vdash \texttt{(* path (}a\ b\texttt{));(* path (}a\ c\texttt{))} \preceq \texttt{(* path (}a\texttt{))}$.

Proof. 1. Reduce first using IX to obtain the subgoals

$$\vdash \texttt{()} \preceq \texttt{(* path (}a\texttt{))} \qquad (13)$$

$$\texttt{(* path (}a\ b\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \vdash \texttt{((}a\ b\texttt{));(* path (}a\ b\texttt{))} \preceq \texttt{(* path (}a\texttt{))}. \qquad (14)$$

Subgoal (13) is resolved using VII and II. Subgoal (14) is resolved by VIII first to

$$\texttt{(* path (}a\ b\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \vdash \texttt{((}a\ b\texttt{));(* path (}a\ b\texttt{))} \preceq \texttt{((}a\texttt{));(* path (}a\texttt{))} \qquad (15)$$

$$\text{I}\quad \frac{\cdot}{\Gamma, p_1 \preceq p_2 \vdash p_1 \preceq p_2}$$

$$\text{II}\quad \frac{\cdot}{\Gamma \vdash p \preceq \texttt{()}}$$

$$\text{III}\quad \frac{\Gamma \vdash p_1 \preceq p_2}{\Gamma \vdash \texttt{(}a\texttt{)}\texttt{;}p_1 \preceq \texttt{(}a\texttt{)}\texttt{;}p_2}$$

$$\text{IV}\quad \frac{\Gamma \vdash \texttt{(}a\texttt{)}\texttt{;}p \preceq r^{a_i}}{\Gamma \vdash \texttt{(}a\texttt{)}\texttt{;}p \preceq \texttt{(* set } r^{a_1}\texttt{,}\ldots\texttt{,}r^{a_n}\texttt{)}}\ 1 \le i \le n$$

$$\text{V}\quad \frac{\Gamma \vdash r^{a_1} \preceq r \quad \cdots \quad \Gamma \vdash r^{a_n} \preceq r}{\Gamma \vdash \texttt{(* set } r^{a_1}\texttt{,}\ldots\texttt{,}r^{a_n}\texttt{)} \preceq r}$$

$$\text{VI}\quad \frac{\Gamma \vdash r_1 \preceq r_2 \qquad \Gamma \vdash p_1 \preceq p_2}{\Gamma \vdash \texttt{(}r_1\texttt{)}\texttt{;}p_1 \preceq \texttt{(}r_2\texttt{)}\texttt{;}p_2}$$

$$\text{VII}\quad \frac{\Gamma \vdash p_1 \preceq p_2}{\Gamma \vdash p_1 \preceq \texttt{(* path } r\texttt{)}\texttt{;}p_2}$$

$$\text{VIII}\quad \frac{\Gamma \vdash p_1 \preceq (r)\texttt{;}\texttt{(* path } r\texttt{)}\texttt{;}p_2}{\Gamma \vdash p_1 \preceq \texttt{(* path } r\texttt{)}\texttt{;}p_2}$$

$$\text{IX}\quad \frac{\Gamma \vdash p_1 \preceq p_2 \qquad \Gamma, \texttt{(* path } r\texttt{)}\texttt{;}p_1 \preceq p_2 \vdash \texttt{(}r\texttt{)}\texttt{;}\texttt{(* path } r\texttt{)}\texttt{;}p_1 \preceq p_2}{\Gamma \vdash \texttt{(* path } r\texttt{)}\texttt{;}p_1 \preceq p_2}$$

Fig. 2. Inference system

then VI to

$$\texttt{(* path (}a\ b\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \vdash \texttt{(}a\ b\texttt{)} \preceq \texttt{(}a\texttt{)} \tag{16}$$

$$\texttt{(* path (}a\ b\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \vdash \\ \texttt{(* path (}a\ b\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \tag{17}$$

which are resolved using III and II, respectively I.
2. The proof reduces, using IX, to

$$\vdash \texttt{(* path (}a\ c\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \tag{18}$$

$$\texttt{(* path (}a\ b\texttt{));(* path (}a\ c\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \vdash \\ \texttt{((}a\ b\texttt{));(* path (}a\ b\texttt{));(* path (}a\ c\texttt{))} \preceq \texttt{(* path (}a\texttt{))} \tag{19}$$

Subgoal (18) is an instance of 1. The proof of (19) follows that of (14). □

The proof system is well-behaved with respect to restricted syntax in the sense that, when used in a bottom-up fashion, if the initial judgment is well-formed then subsequent judgments will be well-formed as well.

Proposition 4. *Let*

$$\frac{\Gamma_1 \vdash e_1 \qquad \cdots \qquad \Gamma_n \vdash e_n}{\Gamma \vdash e}$$

be any instance of one of the proof rules I–IX*. If $\Gamma \vdash e$ is well-formed then so are all $\Gamma_i \vdash e_i$, $1 \leq i \leq n$.* □

We proceed to consider soundness and completeness. Say that Γ is *valid* if $r_1 \leq r_2$ whenever $r_1 \preceq r_2 \in \Gamma$ ($p_1 \preceq p_2 \in \Gamma$), and say that $\Gamma \vdash r_1 \preceq r_2$ is *valid*, written $\Gamma \models r_1 \preceq r_2$, if Γ valid implies $r_1 \leq r_2$. Similar definitions apply to terms of the form p_1, p_2. Soundness holds for arbitrary regular SPKI expressions, not only for restricted ones. This is readily apparent from the soundness proof given in the appendix.

Theorem 1 (Soundness). *If $\Gamma \vdash r_1 \preceq r_2$ then $\Gamma \models r_1 \leq r_2$* □

Completeness, however, holds only for well-formed, restricted expressions. Problematic cases are the rules for sets (rule IV) and paths (rule VII and VIII) which make use of the restricted format in an essential way.

Theorem 2 (Completeness). *Suppose that r_1, r_2 are well-formed restricted expressions. If $r_1 \leq r_2$ then $r_1 \preceq r_2$.* □

The completeness proof is constructive, and provides an algorithm which can be used to decide entailments. If used as-is this algorithm is quadratic: In the worst case, at each step as the input expressions r_1 and r_2 are scanned, set expressions must be scanned against each other. This can easily be brought down to $\mathcal{O}(n \log n)$ if set expressions are sorted according to their tags. So we obtain:

Theorem 3 (Worst-Case Complexity). *The relation $r_1 \leq r_2$ is decidable in time $\mathcal{O}(n \log n)$ where n is the sum of the lengths of r_1 and r_2.*

Proof The completeness proof provides an $\mathcal{O}(n \log n)$ algorithm, provided the expressions are sorted. If the input expressions are unsorted, a preprocessing phase of $\mathcal{O}(n \log n)$ brings them into sorted form first. □

7 Constrained Delegation in SPKI

In this section we discuss how path expressions can be used to represent constrained delegation in SPKI.

SPKI has both a naming and an authorisation component. If we ignore validity checking issues we can, for the purpose of the present discussion, view a SPKI name certificate as a triple

$$\texttt{(}k\texttt{,}n\texttt{,}s\texttt{)} \tag{20}$$

where k is an S-expression representing a key, n is a string atom, and s is an S-expression representing a key or a SDSI name, an S-expression of the form

$$\texttt{(name } k_1 \; n_1 \; \cdots \; n_m\texttt{)} \; .$$

For instance, if n is the atom `personnel-dept` and s is the SDSI name `(name` k_1 `head-office personnel-dept section1)` then the certificate (20) should be read as

"k's `personnel-dept` is k_1's `head-office`'s `personnel-dept`'s `section1`".

Delegation chains will refer to principals, as keys or as SDSI names. Hence entailment must be extended to take name resolution into account. Curiously, this aspect is ignored in standard SPKI. There, authorisation expressions [3] are taken as primitive S-expressions, and there is no defined mechanism for resolving a name appearing as part of a standard SPKI authorisation expression.

Entailment is easily extended to take name resolution into account, using a rewriting approach akin to that of [CEE+01]. One rule and a rule schema needs to be added:

$$\text{X}\quad \frac{\cdot}{\Gamma \vdash s\texttt{;(EOL)} \preceq \texttt{(name } k\ n\texttt{ EOL)}}\ \texttt{(}k\texttt{,}n\texttt{,}s\texttt{) is valid}$$

$$\text{XI}\quad \frac{\Gamma \vdash s_1\texttt{;(EOL)} \preceq s'_1\texttt{;(EOL)} \qquad \Gamma \vdash s_1 \preceq s_1\texttt{;}s'_2}{\Gamma \vdash s_2 \preceq s'_1\texttt{;}s'_2}$$

Observe that rule `(XI)`, with the rules of fig. 2 but in the absence of `(X)`, is admissible, because of completeness. Thus, the only new entailments provable are those arising because of naming (schema `(X)`).

A SPKI authorisation certificate (auth cert) can be viewed as a 4-tuple (again we ignore validity checking):

$$\texttt{(}k\texttt{,}s\texttt{,}d\texttt{,}t\texttt{)} \tag{21}$$

where k is a key, s is a SDSI name, d is a delegation flag, and t is a SPKI tag, a SPKI authorisation expression. There are several ways to adapt the SPKI certificate format to constrained delegation. In this section we consider the case of replacing the d flag with a regular SPKI expression determining a delegation constraint. An alternative would be to include the delegation constraint in the authorisation tag t, as indicated in section 1. This would gain some backwards compatibility at the expense of some notational clarity.

An *extended auth cert* would thus be a certificate as (21) except that d and t would both be regular SPKI expressions, and d, in particular, would represent lists of a form such as

$$\texttt{(delegation } \tau_1\texttt{;(}s_1\texttt{)} \cdots \tau_n\texttt{;(}s_n\texttt{))} \tag{22}$$

where $\tau_1, \ldots, \tau_n$ are tags (in the sense of section 5) and $s_1, \ldots, s_n$ are SDSI names, or possibly the empty list `()` in case no further delegation is possible.

The latter situation arises when k authorises s for t directly, and the former applies when s receives from k the power to pass an authorisation down the delegation chain. This chaining relation can be accounted for in terms of a rewrite relation $\rightarrow$ on auth certs such that if

$$\texttt{(}k\texttt{,}s\texttt{,}d\texttt{,}t\texttt{)} \rightarrow \texttt{(}k'\texttt{,}s'\texttt{,}d'\texttt{,}t'\texttt{)}$$

[3] Tags, in SPKI terminology, not to be confused with tags as used above.

then the validity of (k', s', d', t') follows from the validity of (k, s, d, t) (if they have been issued). The single rule governing delegation chaining will be the following:

$$\frac{\vdash k'\texttt{;(EOL)} \preceq s\texttt{;(EOL)} \qquad \vdash \texttt{(delegation } \tau\texttt{;(}s'\texttt{));}d' \preceq d \qquad \vdash t' \preceq t}{(k, s, d, t) \rightarrow (k', s', d', t')}$$

where τ is a tag.

Example 5. We give an example based on delegated facility access administration. Two organisations are involved, `orgA` and `orgB` with associated keys K_A and K_B. The task is for `orgA` to grant its administrator the right to engage `orgB` to perform some access management on behalf of `orgA` staff. Assume the following name certs:

$$\begin{gathered}(K_A\texttt{,orgB,}K_B)\\(K_A\texttt{,admin,}K_{A,1})\\(K_A\texttt{,staff,}K_{A,2})\\(K_B\texttt{,staff,}K_{B,1})\\(K_{A,2}\texttt{,somebody,}K_{A,3})\end{gathered}$$

Let

$$\begin{aligned}d &= \texttt{(delegation (* path (contractor (name } K_A \texttt{ orgB)));}\\ &\qquad\qquad\qquad\qquad\qquad \texttt{((target (name } K_A \texttt{ staff))))}\\ d' &= \texttt{(delegation (* path (contractor (name } K_B \texttt{ staff)));}\\ &\qquad\qquad\qquad\qquad\qquad \texttt{((target (name } K_A \texttt{ staff))))}\end{aligned}$$

Assume the initial auth cert

$$(K_A\texttt{,(name } K_A \texttt{ admin),}d\texttt{,(tag access))}$$

Then the following is a valid certificate chain authorising access for $K_{A,3}$:

$$\begin{gathered}(K_{A,1}\texttt{,(name } K_B\texttt{),}d\texttt{,(tag access))}\\(K_B\texttt{,(name } K_B \texttt{ staff),}d'\texttt{,}t)\\(K_{B,1}\texttt{,}K_{A,2}\texttt{,(),(tag access)).}\end{gathered}$$

8 Concluding Remarks

We have suggested extending the basic syntax of SPKI with a facility for expressing regular languages, and explored its application in the context of constrained delegation. Constrained delegation is by no means the only conceivable application of such a regular language extension. Another example could be constraints on multidomain routing paths. Also, the extension may open up for more complex authorisation schemes, useful, for instance, in the context of web services orchestration. An example is sequential constraints on authorisation whereby one authorisation (to enter some facility, say) can be made subject to another authorisation having been previously enacted (say, to have been granted some ticket).

The question remains if the extension is too rich, and whether there are other, equally valid ways of achieving the same ends. We doubt, for instance, whether there is much use for nested path expressions. Also, some of the effects which can be obtained with constrained delegation can be obtained equally well by threshold certificates. For instance, for example 5, a similar effect (to avoid `orgB` delegating outside `orgA`) could be achieved by `orgA` initially issuing a threshold cert ensuring that final authorisations can only be applied to `orgA staff`. A systematic investigation of this issue is left for future work.

We proposed also a restricted syntax for which chaining can be decided in polynomial time, as opposed to the exponential worst case running time obtained by a straightforward reduction to regular language containment. Most examples above remain within the restricted fragment, most notably example 5. The algorithm given in the paper does not take name resolution into account. We expect that the techniques of either [CEE+01] or [JR02] can be applied to address the more general problem without substantial problems.

References

[BD02] O. Bandmann and M. Dam. A note on SPKI's authorisation syntax. In *Proc. 1st Annual PKI Research Workshop*, 2002.

[BDF02] O. Bandmann, M. Dam, and B. Sadighi Firozabadi. Constrained delegation. In *Proc. 23rd Annual Symp. on Security and Privacy*, 2002.

[CEE+01] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in spki/sdsi. *Journal of Computer Security*, 9:285–322, 2001.

[EFL+99] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. SPKI Certificate Theory, May 1999. RFC 2693, expired. URL: ftp://ftp.isi.edu/in-notes/rfc2693.txt.

[JR02] S. Jha and T. Reps. Analysis of SPKI/SDSI certificates using model checking. In *Proc. IEEE Computer Security Foundations Workshop*, pages 129–146, 2002.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Riv97] R. Rivest. S-expressions, May 1997. Internet Draft, expired. URL: http://theory.lcs.mit.edu/ rivest/sexp.txt.

Appendix

Proposition 5.

1. $\Downarrow\texttt{(*)} = \|\texttt{(*)}\|$
2. $\Downarrow\texttt{()} = \texttt{()}$
3. $\Downarrow\texttt{(}\sigma\texttt{)} = \{\texttt{(}s\texttt{)} \mid s \in \downarrow\sigma\}$
4. $\Downarrow\sigma_1\,\texttt{;}\,\sigma_2 = \Downarrow\sigma_1\,\texttt{;}\Downarrow\sigma_2$
5. $\Downarrow\texttt{(* set } \sigma_1 \ \cdots \ \sigma_m\texttt{)} = \Downarrow\sigma_1 \cup \cdots \cup \Downarrow\sigma_m$
6. $\Downarrow\texttt{(* path } \sigma\texttt{)} = (\Downarrow\sigma)^*$
7. $\downarrow\sigma = \Downarrow\sigma\,\texttt{;}\,\Sigma^*$

Proof. 1. Trivial
2. $s \in \Downarrow\texttt{()}$ iff $s \leq \texttt{()}$ iff $s \in \Sigma^*$.
3. $s \in \Downarrow\texttt{(}\sigma\texttt{)}$ iff $\exists s_1 \in \|\texttt{(}\sigma\texttt{)}\|$ such that $s \leq s_1$ iff $\exists s_2.s \leq \texttt{(}s_2\texttt{)}$ and $s_2 \in \|\sigma\|$ iff $\exists s_2', s_3.s = \texttt{(}s_2'\texttt{)}\,\texttt{;}\,s_3$, $s_2' \in \downarrow\sigma$, and $s_3 \in \Sigma^*$ iff $s \in \{\texttt{(}s'\texttt{)} \mid s' \in \downarrow\sigma\}\,\texttt{;}\,\Sigma^*$.
4. $s \in \Downarrow(\sigma_1\,\texttt{;}\,\sigma_2)$ iff $\exists s_1 \in \|\sigma_1\|, s_2 \in \|\sigma_2\|.s \leq s_1\,\texttt{;}\,s_2$ iff $\exists s_1 \in \Downarrow\sigma_1, s_2 \in \Downarrow\sigma_2.s = s_1\,\texttt{;}\,s_2$ iff $s \in \Downarrow\sigma_1\,\texttt{;}\Downarrow\sigma_2$
5. $s \in \Downarrow\texttt{(* set } \sigma_1 \ \cdots \ \sigma_m\texttt{)}$ iff $s \in \Downarrow\sigma_1 \cup \cdots \cup \Downarrow\sigma_m$
6. $s \in \Downarrow\texttt{(* path } \sigma\texttt{)}$ iff $s = \texttt{(}s_1\texttt{)}\,\texttt{;}\cdots\texttt{;}\,\texttt{(}s_n\texttt{)}\,\texttt{;}\,s', \forall i : 1 \leq i \leq n.s_i \in \Downarrow\sigma, s' \in \Sigma^*$ iff $s \in (\Downarrow\sigma)^*\,\texttt{;}\,\Sigma^*$
7. Immediate by the definition. □

Theorem 4 (Soundness). *If $\Gamma \vdash r_1 \preceq r_2$ then $\Gamma \models r_1 \leq r_2$*

Proof. We prove the statement for both sequents of the form $\Gamma \vdash r_1 \preceq r_2$ and $\Gamma \vdash p_1 \preceq p_2$ by induction on the structure of proof. All cases except IX are routine. We go through the non-trivial cases one by one.

III. Suppose Γ and $\Gamma \vdash p_1 \preceq p_2$ are both valid. Let $s \in \downarrow(\texttt{(}a\texttt{)}\,\texttt{;}\,p_1)$. By proposition 2, $s = \texttt{(}a \ \ s_1 \ \cdots \ s_n\texttt{)}$, and $s' = \texttt{(}s_1 \ \cdots \ s_n\texttt{)} \in \downarrow p_1$. By the assumption and fact 1, $s' \in \downarrow p_2$, so $s \in \downarrow\texttt{(}a\texttt{)}\,\texttt{;}\,p_2$, as desired.

VI. Suppose that Γ, $\Gamma \vdash r_1 \preceq r_2$ and $\Gamma \vdash p_1 \preceq p_2$ are all valid, and assume that $s \in \downarrow(\texttt{(}r_1\texttt{)}\,\texttt{;}\,p_1)$. By proposition 2, s has the shape $\texttt{(}s_1 \ \ s_2 \ \cdots \ s_n\texttt{)}$ such that $s_1 \in \downarrow r_1$ and $\texttt{(}s_2 \ \cdots \ s_n\texttt{)} \in \downarrow p_1$. But then $s_1 \in \downarrow r_2$ and $\texttt{(}s_2 \ \cdots \ s_n\texttt{)} \in \downarrow p_2$ as well, so $s \in \downarrow(\texttt{(}r_2\texttt{)}\,\texttt{;}\,p_2)$ as we wanted.

VII. Suppose that Γ and $\Gamma \vdash p_1 \preceq p_2$ are both valid. Assume also that $s \in \downarrow p_1$. Then $s \in \downarrow(\texttt{(* path } r\texttt{)}\,\texttt{;}\,p_2)$ as well, since $\texttt{()} \in \downarrow\texttt{(* path } r\texttt{)}$ for any r and k.

VIII. Assume that Γ and $\Gamma \vdash p_1 \preceq_m \texttt{(}r\texttt{)}\,\texttt{;}\,\texttt{(* path } r\texttt{)}\,\texttt{;}\,p_2$ are both valid, and that $s \in \downarrow p_1$. Then $s = \texttt{(}s_1 \ \cdots \ s_h \ \ s_{h+1} \ \cdots \ s_k \ \ s_{k+1} \ \cdots \ s_l\texttt{)}$ such that $\texttt{(}s_1 \ \cdots \ s_h\texttt{)} \in \downarrow r$, $\texttt{(}s_{h+1} \ \cdots \ s_k\texttt{)} \in \downarrow(\texttt{(* path } r\texttt{)}\,\texttt{;}\,p_2)$, and $\texttt{(}s_{k+1} \ \cdots \ s_l\texttt{)} \in p_2$. It follows, by proposition 2, that $\texttt{(}s_1 \ \cdots \ s_h \ \ s_{h+1} \ \cdots \ s_k\texttt{)} \in \downarrow(\texttt{(* path } r\texttt{)}\,\texttt{;}\,p_2)$, so $s \in \texttt{(* path } r\texttt{)}\,\texttt{;}\,p_2$ as desired.

IX. This is the only slightly tricky case. We now assume that a proof is given, but that the conclusion of the proof is false. ¿From these assumptions a contradiction is derived. Define

$$C^0 = \texttt{()}$$
$$C^{n+1} = C^n \cup C^n\,\texttt{;}\,C$$

Obviously, $C^* = \bigcup_{n\in\omega} C^n$. We use these n's to annotate path expressions in the proof, in this way deriving the contradiction. The annotation uses expressions $\texttt{(* path } r\texttt{)}^n$ which denote $(\downarrow r)^n$.

Assume now that $\Gamma \vdash p_1 \preceq p_2$ and

$$\Gamma, \texttt{(* path } r\texttt{);}p_1 \preceq p_2 \vdash \texttt{(}r\texttt{);(* path } r\texttt{);}p_1 \preceq p_2$$

are all valid, but $\Gamma \vdash \texttt{(* path } r\texttt{);}p_1 \preceq p_2$ is not. Then Γ must valid and $\texttt{(* path } r\texttt{);}p_1 \not\leq p_2$, i.e. $(\downarrow r)^*\texttt{;}\downarrow p_1 \not\subseteq \downarrow p_2$. Then we find an $n \in \omega$ such that $(\downarrow r)^n\texttt{;}\downarrow p_1 \subseteq \downarrow p_2$ but not $(\downarrow r)^{n+1}\texttt{;}\downarrow p_1 \subseteq \downarrow p_2$. The application of IX we are considering is now annotated as follows:

$$\frac{\Gamma \vdash p_1 \preceq p_2 \quad \Gamma, \texttt{(* path } r\texttt{)}^n\texttt{;}p_1 \preceq p_2 \vdash \texttt{(}r\texttt{);(* path } r\texttt{)}^n\texttt{;}p_1 \preceq p_2}{\Gamma \vdash \texttt{(* path } r\texttt{)}^{n+1}\texttt{;}p_1 \preceq p_2}$$

The annotation of the proof is completed simply by letting annotations propagate, using the annotated version of IX in place of IX proper. Now, for the annotated proof, except possibly for instances of I, if the parent (the conclusion) of a rule instance is invalid, then so is one of the children. As a consequence we can trace a path from the invalid proof node $\Gamma \vdash \texttt{(* path } r\texttt{);}p_1 \preceq p_2$ to a leaf, an instance of I, using only invalid sequents. We may assume that there are no further applications of rule IX along this path (otherwise it suffices to consider a proper suffix). It follows that the invalid leaf node must have the shape $\Gamma, \texttt{(* path } r\texttt{)}^n\texttt{;}p_1 \preceq p_2 \vdash \texttt{(* path } r\texttt{)}^n\texttt{;}p_1 \preceq p_2$, but this node is valid, a contradiction. It follows that no proof can lead to a false conclusion, which is what we had to show. □

Theorem 5 (Completeness). *Suppose that r_1, r_2 are well-formed restricted expressions. If $r_1 \leq r_2$ then $r_1 \preceq r_2$*

Proof. We assume that $r_1 \leq r_2$ and give a bottom-up strategy for building a proof of $r_1 \preceq r_2$.

$r_1 = \texttt{()}$. Refine using II.

$r_2 = \texttt{()}$. Any other case than $r_1 = \texttt{()}$ is a contradiction.

$r_1 = \texttt{(}a\texttt{);}p_1$, and $r_2 = \texttt{(}a'\texttt{);}p_2$. The case where $a \neq a'$ is a contradiction. Otherwise we must have $p_1 \leq p_2$, and we refine using III.

$r_1 = \texttt{(}a\texttt{);}p$, $r_2 = \texttt{(* set } r^{a_1} \cdots r^{a_n}\texttt{)}$. We must have $a = a_i$ for exactly one i, and $r_1 \leq r^a$. Refine using IV.

$r_1 = \texttt{(* set } r^{a_1} \cdots r^{a_n}\texttt{)}$, $r_2 = \texttt{(}a\texttt{);}p_2$. This case is not very interesting. We must have $n = 1$, $a_1 = a$ and $r^a \leq r_2$, and we refine according to V.

$r_1 = \texttt{(* set } r_1^{a_{1,1}} \cdots r_1^{a_{1,n_1}}\texttt{)}$, $r_2 = \texttt{(* set } r_2^{a_{2,1}} \cdots r_2^{a_{2,n_2}}\texttt{)}$. In this case let $\alpha_i = \{a_{i,1}, \cdots, a_{i,n_i}\}$, $i \in \{1,2\}$. We must have $\alpha_1 \subseteq \alpha_2$ and for each $a \in \alpha_1$ it will be the case that $r_1^a \leq r_2^a$. Consequently we refine using first V, then IV.

We then proceed by assuming $p_1 \leq p_2$. Consider first the cases where p_1 has the shape $\texttt{(}a\texttt{);}p_1'$, and p_2 has one of the shapes $\texttt{(}r\texttt{);}p_1'$ or $\texttt{(* path } r\texttt{);}p_1'$, or vice versa. These cases are contradictions and so cannot occur. We proceed:

$p_1 = \texttt{(}r_1\texttt{)}\,\texttt{;}p'_1$, $p_2 = \texttt{(}r_2\texttt{)}\,\texttt{;}p'_2$. In this case we obtain $r_1 \leq r_2$ and $p'_1 \leq p'_2$, and we refine using VI.

$p_1 = \texttt{(}r_1\texttt{)}\,\texttt{;}p'_1$, $p_2 = \texttt{(* path }r_2\texttt{)}\,\texttt{;}p'_2$. Since $p_1 \leq p_2$, either $\mathit{tags}\ (r_1) \cap \mathit{tags}\ (r_2) = \emptyset$ or else $\mathit{tags}\ (r_1) \subseteq \mathit{tags}\ (r_2)$. In the first case we obtain that $p_1 \leq p'_2$ directly, and refine using VII. In the second case note first that we may assume that $\mathit{tags}\ (r_1) \neq \emptyset$ since otherwise the first subcase applies. Assume that $p'_2 = q_1\,\texttt{;}\cdots\texttt{;}q_n$. If $n = 0$, i.e. $p'_2 = \texttt{()}$, then we obtain directly that $p_1 \leq \texttt{(}r_2\texttt{)}\,\texttt{;}\,\texttt{(* path }r_2\texttt{)}\,\texttt{;}p'_2$ and so refine by VIII. Thus we can assume that $n > 0$. By the properties of Kleene star we know that $\downarrow p_1 \subseteq (\downarrow p'_2) \cup ((\downarrow r_2)\,\texttt{;}\,(\downarrow r_2)^*\,\texttt{;}\,(\downarrow p'_2))$. By well-formedness we know that $\mathit{tags}\ (r_2) \cap \mathit{tags}\ (q_1) = \emptyset$. Then $\mathit{tags}\ (r_1) \cap \mathit{tags}\ (q_1) = \emptyset$ too, so $\downarrow p_1 \cap \downarrow p'_2 = \emptyset$, whence we may conclude that $p_1 \leq \texttt{(}r_2\texttt{)}\,\texttt{;}\,\texttt{(* path }r_2\texttt{)}\,\texttt{;}p'_2$ and refine by VIII.

$p_1 = \texttt{(* path }r_1\texttt{)}\,\texttt{;}p'_1$ and either $p_2 = \texttt{(}r_2\texttt{)}\,\texttt{;}p'_2$ or $p_2 = \texttt{(* path }r_2\texttt{)}\,\texttt{;}p'_2$. If we find that $p_1 \preceq p_2$ is in the current set Γ proof construction terminates. Otherwise, by $p_1 \leq p_2$ we obtain directly that $p'_1 \leq p_2$ and that $\texttt{(}r_1\texttt{)}\,\texttt{;}\,\texttt{(* path }r_1\texttt{)}\,\texttt{;}p'_1 \leq p_2$, and so refine by IX.

We have thus defined a procedure which constructs a proof from a valid sequent. Our task is to show that the procedure terminates. This is not difficult to see. Let a proof structure (i.e. a tree, possibly infinite, rooted in a sequent, say, $\Gamma \vdash r_1 \preceq r_2$, and constructed according to the proof rules) be given. Let N be the set of all expressions r and p appearing somewhere in this proof structure. This set will be finite. Now, if the procedure fails to terminate it will be possible to trace a path through the (infinite) proof structure starting from the root which infinitely often visits a sequent $\Gamma \vdash p_1 \preceq p_2$ for fixed choices of p_1 and p_2. We can assume that p_1 has the shape $p_1 = \texttt{(* path }r_1\texttt{)}\,\texttt{;}p'_1$, and that in each case the next rule applied is rule IX. But then, already at the second occurrence of this sequent, I will be applicable, a contradiction. $\square$

Regular SPKI
(Transcript of Discussion)

Mads Dam

Royal Institute of Technology (KTH), Sweden

Pekka Nikander: I don't really understand the power of your restricted delegation. You said you also have the possibility of using threshold certificates. Can't you solve this particular problem at least just by using threshold certificates?

Reply: I think that there are various reasons why you could not solve this problem using mechanisms already there. We don't have negative permissions, if you did you could certainly combine positive and negative permissions to address this. I considered using threshold certificates here for this reason, but I don't think it's a particularly elegant solution for this problem.

Pekka Nikander: Actually I think there is a very simple solution which basically says that you need to have a threshold certificate where one part is from A and one is from B. If you say that you need two certificates, where one is rooted at A_0 and the other one is rooted from B_0, then A is just saying somebody belongs to the set of A and B is delegating the authority.

Reply: I think this is not really the essential message here. The essential issue is how we need to constrain the delegation tree in general.

Matt Blaze: Even though SPKI and other related things have been around for quite a while, I have no intuition of what these certificates actually look like in practice. I mean, you're drawing a hypothetical graph but I have no sense at all as to whether or not this graph has been referred to practice; whether or not graphs that have a thousand times more elements in the path like this might exist in practice. Partly that's because in reality there are asymptotically zero certificates out there in the world. Are there convincing arguments about what these graphs will look like?

Pekka Nikander: We need some strong analysis for this[1]. There are some analyses of what the graph would look like in some cases, as well on how to search the graphs, and how to actually find the solution in practice.

Reply: Well I would say the deeper issue is really about matching the structure to the organisational problem of how authority is actually delegated.

[1] Pekka Nikander, An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems, Ph.D. Dissertation, Helsinki University of Technology, March 1999.

Matt Blaze: This brings us back to the X.509 PKI role. We have this very straightforward hierarchy, that X.509 implements, that is very difficult to understand. When we move this into the commercial world, the companies that actually issue the certificates in fact do everything in their power to make the hierarchy completely flat so that they get paid for every certificate. So, you know, our intuition about what the graphs will look like is often quite wrong. I completely agree that it's a different game now, I'm just arguing that we still don't know what the game actually looks like.

Federated Identity-Management Protocols

Birgit Pfitzmann and Michael Waidner

IBM Zurich Research Lab
{bpf,wmi}@zurich.ibm.com

Abstract. For authentication, one answer to the workshop question "where have all the protocols gone?" is "into federated identity management". At least this is what many influential industrial players are currently striving for. The best-known examples are Microsoft Passport, the Liberty Alliance's proposals, and WS-Federation. While there have been many political discussions about Passport, in particular its privacy, and some technical studies of operational risks, there is almost no public literature about the actual protocols and their security.

We start with an overview of the driving factors in this space, the security properties desirable and achievable under the given design constraints, and the protocols proposed so far. We present a new protocol, BBAE, with better privacy and scalability, i.e., absence of single points of control, than prior proposals. We also discuss particular difficulties of rigorously treating a protocol that can be a profile in current standardization efforts.[1]

1 Introduction

While cryptography and security research has devised security protocols for a wide range of functionality, such as payment systems and fair exchange, the majority of security protocols used in real networks and products only aim at user authentication and secure channel establishment. Research has also produced many good protocols achieving these goals; they are even the main focus of the use of formal methods in security. Nevertheless, a surprising number of new such protocols are continuously being designed in practice, and typically not by adopting well-investigated proposals. While lack of knowledge and a marketing-driven desire for novelty certainly play a role, new applications truly have new requirements, and it seems to require researchers to extend the existing basic ideas appropriately to these new requirements.

Federated identity management is a case in point. The high-level goal of enterprises is to simplify user management in an increasingly dynamic world. In particular, they want to benefit from user registration done in other places for their own user management. The market demand is mostly for business-to-business scenarios, less than the much-discussed initial scenario of Microsoft Passport where one enterprise essentially authenticates the world population. Current demand is even mostly for simplifications within one enterprise, and the

[1] This paper reflects the view of the authors, which is not necessarily shared by IBM.

next steps are scenarios like access control for employees of supply-chain partners and customer-relationship management in a federation of travel agencies, airlines, and hotels.

The main requirement on current federated identity-management protocols is to work if the user agent is only a standard browser. This is called *browser-based* or *zero-footprint*. The reason is that a large percentage of potential users has proved to be unwilling to download specific software. Further, the solutions should work without active content, because many users turn that off for security reasons. Optimally, they should even work without cookies for the same reason; however, not all solutions do, and it is definitely accepted to take advantage of cookies if a browser returns them. Enterprises even require that the solution can be *browser-stateless*. This means that the user uses multiple browsers on different machines, up to working only from Internet cafes. While the latter is not advisable for security, nobody is forced to use Internet cafes at least in developed countries, and we simply accept this requirement.

In research terms, this amounts to three-party authentication, possibly under pseudonyms, combined with an exchange of user attributes. We call the third party *wallet* and the recipient of the authentication and attributes *destination site*. However, no state-of-the-art protocol applies because they all assume special user agents. In particular, a user agent in three-party authentication typically uses a newly generated or received cryptographic key, while a standard browser will not do this. We have to work with browser redirects and other standard HTTP constructs only. More precisely, HTTP, HTML, and secure SSL/TLS channels triggered by HTTPS addresses are assumed. Normal authentication protocols are known to be very prone to design errors, and the browser constraint can only make things worse. Hence we believe that detailed security considerations for such protocols will become very important. Further, the transfer of additional attributes and the possibly quasi-automatic usage adds privacy as an important dimension.

In this paper, we start with an overview of the security and privacy properties desirable and achievable under the zero-footprint and browser-stateless constraints, and the protocols proposed so far. We then present a new protocol, BBAE, with better privacy and scalability, i.e., absence of single points of control, than prior proposals. Between this workshop and these proceedings, scalability has been addressed in similar ways in standards proposals, while similar privacy is still not achieved.

We designed BBAE to be suitable as a profile in existing standardization efforts such as SAML, Liberty or WS-Federation, as this is the best chance of adoption. While this paper contains a more precise and mathematical definition of BBAE, this point of view had two consequences: First, a few parameters are used in specific ways to fit a particular standard. Secondly, the definition is more modular than usual in research definitions of security protocols. All current standardization efforts aim for such modularity, e.g., independence of underlying transport protocols, cryptographic mechanisms, and policies, and even relative independence of concrete message formats and message flows, the so-called profiles. (We discuss this tendency a bit more in the outlook.) Concretely, we have specified this version of BBAE completely as a SAML profile, i.e., with SAML message formats and general constraints. This specification was prototyped by Stephen Levy in the IBM Privacy Services prototype (for an earlier version see [3]). At the time of this workshop, SAML was the only choice, containing the only message standard with general attributes. The only real restriction by this choice was in the inclusion of privacy policies, see below. In the meantime, we also specified BBAE as a WS-Federation Passive Requestor profile with very minor changes together with Daniel Lutz, ETH Zurich, who also prototyped that version. The existence of these profile specifications should not be misunderstood as IBM input to standardization.

A particular reason to present profiles of standards like SAML more formally is that each participant has several names in such profiles, e.g., in the certificate in an underlying SSL channel, in the certificate for an XML signature, and in metadata; in text representations it is hard to say precisely which name is used where. Additionally, there can be several channel and protocol identifiers.

Two other challenges in precisely describing a browser-based protocol are independent of a relation to standards: One has to describe the user and the browser. We are not aware of prior protocol definitions that include user rules. Providing them is important for a later security analysis, because in a zero-footprint, browser-stateless protocol the user is an important protocol principal, and security can only be defined with respect to certain user actions and shown under certain assumptions about the user behavior. It is vital for implementations and documentation of such protocols to be aware of this semantics of the users actions. We even recommend to standardize the corresponding user syntax, i.e., the graphical user interface, to a large extent.

2 Existing Proposals and Design Goals

Figure 1 gives an overview of browser-based federated identity-management protocols. The first such protocol was Microsoft Passport. It is not published, but guessable from existing publications [11,9]. The only open standard so far is SAML [17]. Shibboleth [18] is a more elaborated SAML application to a university federation. The Liberty Alliance, also building on SAML, makes public proposals without an open standardization process [10]. WS-Federation belongs to a larger web-services roadmap by IBM and Microsoft; its passive requestor profile is the browser-based case [20]. BBAE is our proposal of this paper; it precedes Liberty 2 and WS-Federation. WebSEAL is a product from the enterprise space [8], its 2002 version is shown for comparison with standardization efforts.

The round icons are browser-based protocols. For comparison, the square icons show other protocol classes for web authentication and attribute exchange, but not zero-footprint and browser-stateless. "Form filler" denotes local wallet products like [7,16,12,23] and more recently browser personalization. "Proxy wallets" denotes similar products working as remote proxies. As they see the user's entire web traffic, they are much worse for privacy than the protocol class discussed here where the wallets only participate in authentication and attribute

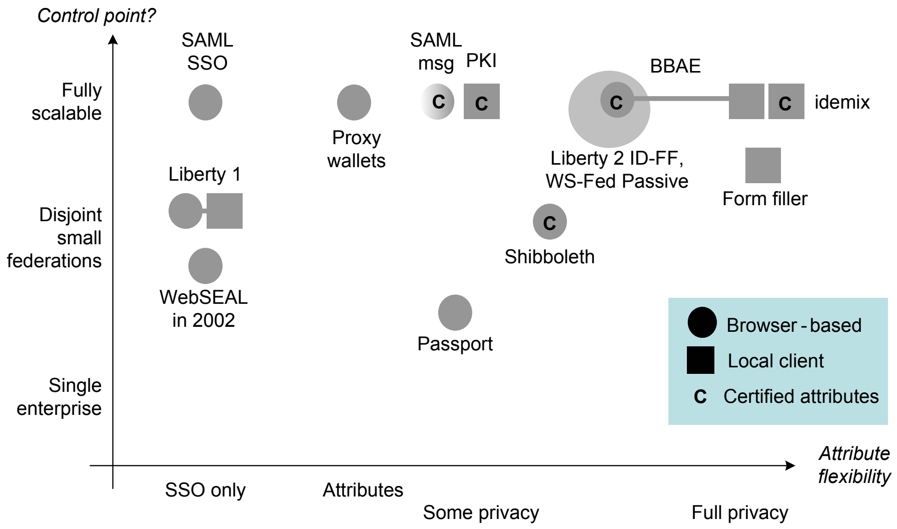


Fig. 1. Overview of browser-based protocols and some related techniques

exchange. "PKI" means public-key infrastructure, and "idemix" (Identity Mixer) is an IBM prototype of unlinkable credentials [5].

Vertically we show to what extent a single *control point* is inherent in the protocol design. We call the opposite *scalable* because a single control point may be acceptable in one enterprise or a small federation, but not beyond. Control points occur for different reasons in protocols: the wallet addressing may be fixed, at least per installation, as in Passport; all wallets may have to trust each other; there may be no room for certificate chains in messages; or there may be a symmetric-key distribution center that can impersonate all other parties. A common cookie domain, as proposed by Liberty, is at least dangerous. However, we accept the lack of a fully specified set-up phase as long as such a phase could be added.

Horizontally, we first show whether only single signon is offered or multiple user *attributes* can be exchanged. We joined this with the *privacy* dimension because multiple attributes without privacy are simply a very long identity. Privacy on this protocol layer means that all information exchange can be governed by policy, and arbitrary policies are possible. In other words, no information is exchanged automatically. We include here that users are not forced to give attributes to third-party wallets, i.e., they can choose between the zero-footprint version and *local wallets* in the same protocol. One exception from privacy seems unavoidable for browser-based protocols: A wallet learns the identities of the destination sites that a user authenticates to. Detailed privacy goals and which protocol features enable them are discussed in [14]. Note that privacy in the overall system additionally requires a well-designed policy-management system and privacy on lower layers.

The "C" in some icons means that attributes may be certified, i.e., confirmed by the wallet or other parties. In most protocols certified attributes allow less privacy than uncertified ones (such as the user's book preferences). Combining certified attributes and strong anonymity is impossible with browser-based protocols. Where one needs that, one has to give up the zero-footprint restriction and upgrade to cryptographic unlinkable credentials [4], here exemplified by idemix [5].

Looking at browser-based protocols (round icons) only, and omitting the shaded "SAML msg" icon, which denotes that SAML already defines message formats for this case, but no protocols yet, one sees that no other protocol addresses full privacy, and no protocol prior to BBAE addressed attribute exchange for the fully federated case.

3 Overview of the BBAE Protocol

In this section, we present a browser-based attribute-exchange protocol, BBAE.

3.1 Message Flow

Figure 2 shows the message flow of the longest execution path of the BBAE protocol, and when no error occurs. Steps 0 and 13 show that the user is assumed to browse at the destination site before the protocol and to get some application-level response after the protocol. Steps 1-2 locate the wallet, Steps 3-4 redirect the browser to it, Step 5 authenticates the user to the wallet, Steps 6-10 transport the actual request and response on a backchannel, where Step 8 allows the real-time release of attributes, and Steps 11-12 redirect the browser back to the destination site with a handle that links this browser to the response.

The figure contains all the exchanged parameters with hopefully self-explanatory names (*adr* denotes addresses and *dsid* a session identifier), except that all messages also contain a protocol identifier and a message number within the protocol. If not stated otherwise, these identifiers are not protected except by the secure channels.

The submodules used are defined in Section 4, the detailed protocol steps in Section 5, and Section 6 gives additional explanations when and why these steps are necessary.

3.2 Notation

We use a straight font for `constants`, including `constant sets`, `functions`, and `submodules`, and italics for *variables.* Assignment by possibly probabilistic functions is written $\leftarrow$. An input *in* to an asynchronous submodule `module` is written $\texttt{module!}(in)$ and an output *out* from it $\texttt{module?}(out)$. Most of these submodules are distributed, i.e., they get inputs and make outputs for several participants of the main protocol. Simple sending of a message is shown as $—m \rightarrow$ or $\leftarrow m—$ between participants; concretely it stands for HTTP messages.

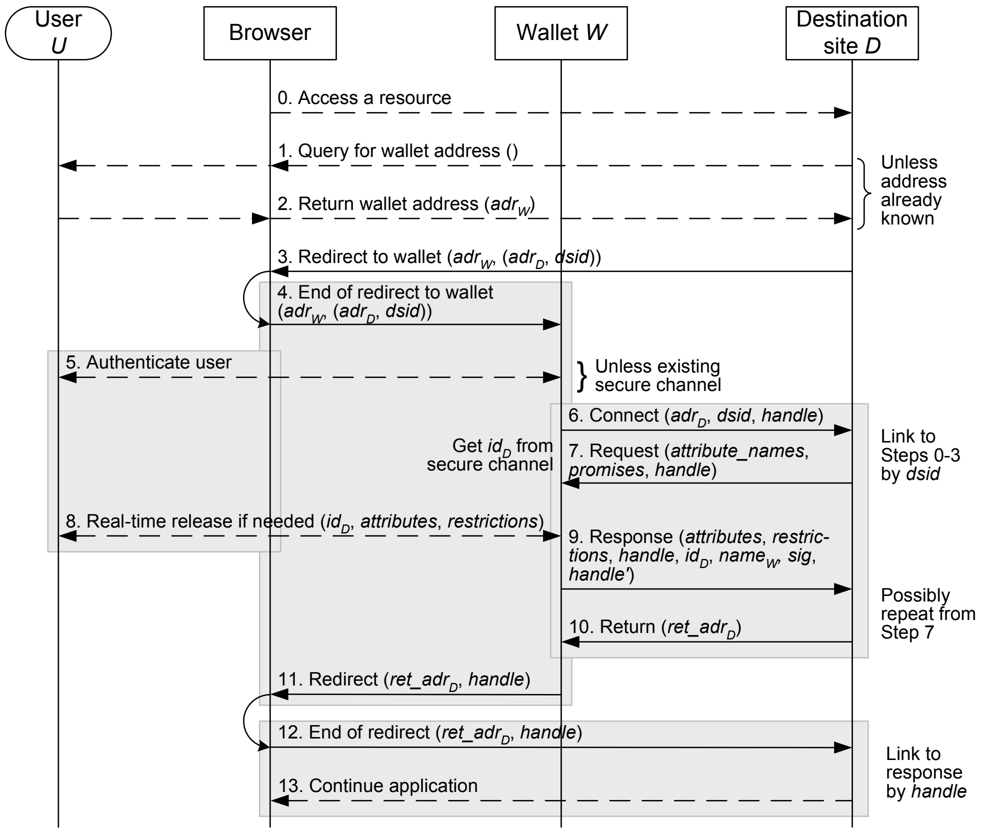

User U
Browser
Wallet W
Destination site D
0. Access a resource
1. Query for wallet address ()
2. Return wallet address (adr_W)
Unless address already known
3. Redirect to wallet (adr_W, (adr_D, $dsid$))
4. End of redirect to wallet (adr_W, (adr_D, $dsid$))
5. Authenticate user
Unless existing secure channel
6. Connect (adr_D, $dsid$, $handle$)
Get id_D from secure channel
7. Request ($attribute\_names$, $promises$, $handle$)
Link to Steps 0-3 by $dsid$
8. Real-time release if needed (id_D, $attributes$, $restrictions$)
9. Response ($attributes$, $restrictions$, $handle$, id_D, $name_W$, sig, $handle'$)
Possibly repeat from Step 7
10. Return (ret_adr_D)
11. Redirect (ret_adr_D, $handle$)
12. End of redirect (ret_adr_D, $handle$)
13. Continue application
Link to response by $handle$

Fig. 2. BBAE protocol with abstract parameters. Steps with dashed lines are only needed in certain cases. The boxes denote secure channels.

4 Required Submodules and Set-Up

Browser-based protocols, at least if one keeps close to standardization, use several submodules whose implementation is left open. Some submodules require that certain data is exchanged in advance; this corresponds to constraints on their global initial states. In this section, we describe these submodules and constraints. BBAE does not require parameter exchange except for its submodules, and in the concrete version this only means SSL server certificates and XML signature capabilities. This makes BBAE easier to bootstrap than all prior proposals. Negotiation of common submodules among multiple possible ones is not part of the subsequent protocol description. Further, we do not present details of how a user's authentication information, attributes, and privacy policies are managed over time.

4.1 Browser Channels

Every wallet and every destination site must be able to maintain secure channels with standard browsers. We define this to mean that message confidentiality and integrity hold with respect to the same principals for all messages sent over the channel. The easiest implementation is SSL/TLS channels, triggered by HTTPS addresses [21]. Servers that release SSL channels after each HTTP response may use another secure implementation, but should have at least one zero-footprint version.

Submodule "`secchan`*".* We denote the submodule as `secchan` and the possible actions as follows:

	Browser		Server
1	$\texttt{secchan!}(\texttt{new}, \mathit{adr})$	$\rightarrow$	$\texttt{secchan?}(\texttt{new}, \mathit{cid}, \mathit{adr})$
2	$\texttt{secchan?}(\texttt{accepted}, \mathit{cid}, \mathit{adr}, \mathit{id})$	$\leftarrow$	$\texttt{secchan!}(\texttt{accept}, \mathit{cid}, \mathit{id})$
3	$\texttt{secchan!}(\texttt{send}, \mathit{cid}, m)$	$\rightarrow$	$\texttt{secchan?}(\texttt{receive}, \mathit{cid}, m)$
4	$\texttt{secchan?}(\texttt{receive}, \mathit{cid}, m')$	$\leftarrow$	$\texttt{secchan!}(\texttt{send}, \mathit{cid}, m')$

Line 1 shows that the browser initiates a secure channel to an address $\mathit{adr} \in$ `URLHost`. This set denotes the set of URL host names, with `localhost` $\in$ `URLHost`. Recall that "!" denotes that this is an input to an asynchronous module. The server is notified with a channel identifier *cid*. Line 2 shows that the server may accept the channel and identify itself under an identity *id*. We denote anonymity by $\mathit{id} = \epsilon$. The browser is notified of the acceptance and of *id* and *cid*. Then both parties may send. This is shown in Lines 3-4 with messages m and m'.

The abstract message

$$\texttt{redirectS}(\mathit{adr}, \mathit{path}, \mathit{query})$$

models a redirect (HTTP 302 or 303) to `https://adr/path?querystring`, where *querystring* is an encoding of the abstract *query*. Its consequences are that the browser establishes a secure channel to the address *adr* and then sends *path* and *query* over that channel [21]. Modeling the channel establishment within the redirect explicitly is important for a security analysis.

Certificates needed. In BBAE, wallets must be identified in secure browser sessions. Concretely, they need SSL/TLS certificates acceptable to browsers. Abstractly, we require that each wallet W has an identity id_W such that, if W uses $\mathit{id} := \mathit{id}_W$ in Line 2, a correct browser gets the output $\texttt{secchan?}(\texttt{accepted}, \mathit{cid}, \mathit{adr}, \mathit{id}_W)$. Nobody else must be able to achieve this, i.e., to impersonate the wallet under id_W.

User involvement. Browsers must reliably present secure channels and the partner identity *id* to their users. The typical implementation is browser icons for windows with secure channels and the possibility to look up certificates. Wallets may support this by personalizing the window content. We denote the outputs a user receives, and inputs that he makes, by

$$\texttt{secchan?}(\texttt{receive}, \textit{id}, \text{m'});^{2}$$

$$\texttt{secchan!}(\texttt{send}, \textit{id}, \text{m}).$$

4.2 Backchannel Sessions

We also use secure channels between wallets and destination sites. We model these secure channels by the same module `secchan` as the browser channels, because they will typically also be SSL/TLS. However, as no party is a browser here, this implementation is easier to change. Again the initiating party need not identify itself; hence the in- and outputs are exactly as above. Destination sites must typically be identified; otherwise they only obtain the attributes that the users' privacy policies permit for anonymous recipients. Let id_D be the identity under which destination site D can securely identify itself in secure channels, in the same sense as for wallet identities id_W above. If a real organization has several such identities, we treat it as several destination sites because the identity choice can be made before a BBAE execution.

4.3 Response Authentication

A wallet must typically authenticate responses for a destination site. With SAML messages, this means XML signature capabilities [22]. They comprise both digital signatures in the cryptographic sense and authentication codes and allow many types of certification and key distribution. We discuss typical usages in Section 9. Generally, we write such authentication as a function

$$m' \leftarrow \texttt{auth}(id, m, strength)$$

for authenticating a message m under an identity id with a certain strength, and

$$(id, m, strength) \leftarrow \texttt{test}(m')$$

for verifying a received message and extracting an identity id, a payload m, and a strength. We denote failure by $(id, m) = (\epsilon, \epsilon)$ and the unauthenticated case by $strength = \epsilon$ and $m' = (id, m)$. We assume that a domain `Strengths` for the parameters *strength* is given; for implementations without this notion we define `Strengths` to have one element only. The Liberty Alliance made detailed proposals [10]. The functions denote authentication including all necessary exchange and verification of keys and certificates.

[2] We only model that the user sees the partner identity *id*, not a channel identifier, because he will not notice if a channel is interrupted. Usually, however, he can distinguish different channels with one partner by different windows.

4.4 User Registration

A user U must register with at least one wallet. Later, $Wallets_U$ denotes the set of wallets that U registered with, and $Users_W$ the set of user identities registered at a wallet W.

Installation or address. Let $local_W \in$ `Bool` denote whether a wallet W is local. (This can be fixed for all users of W.) If W is local, it must be installed; this should be combined with a local firewall that only allows local access to this wallet. If W is remote, the user obtains a contact address $adr_W \in$ `URLHost` of it (else let $adr_W := \epsilon$). This address must be fixed for all users of W unless a user consented to a different privacy policy. We assume a fixed path `BBAEpath` by which these addresses are extended to address the wallet services.

User authentication module "`uauth`*".* The user and the wallet set up a method for later user authentication via a browser. We explicitly model passwords, in order to analyze the security of the zero-footprint and browser-stateless case. Users who do not insist on these properties may set up higher-quality authentication in particular with remote wallets. The overall method must comprise means to protect the user from fake-wallet attacks, because the browser arrives at the wallet (or an adversary) by redirection from an untrusted site. This means at least user education about verifying the wallet certificate.[3] We denote the submodule as `uauth` and the possible actions as follows:

	User		Wallet
1	$\texttt{uauth?}(\texttt{start}, id_W)$	$\leftarrow$	$\texttt{uauth!}(\texttt{start}, cid)$
2	$\texttt{uauth!}(\texttt{do}, id_W, login)$	$\rightarrow$	$\texttt{uauth?}(\texttt{done}, cid, id_U)$

Line 1 denotes that the wallet initializes user authentication over a secure channel with identifier *cid*. Thus the module `uauth` uses the module `secchan`. At the user, this leads to an output that asks for authentication. This output contains the wallet's identity id_W that the browser obtained in the set-up of channel *cid*. With current concrete implementations this happens automatically by the browser window. The user inputs login information *login* into the same window (Line 2), and the wallet derives a user identity $id_U \in Users_W \cup \{\epsilon\}$, where ϵ denotes failure.

Exchanged parameters. After registration of user U at wallet W, the user knows an identity id_W that the wallet can use for secure browser channels (compare Section 4.1), and the wallet knows an identity id_U of the user.[4] Further, they share login information $login_{U,W}$. We have to assume that the entropy of $login_{U,W}$

[3] A dangerous feature in Passport and Liberty is "inline single signon", where the wallet uses a part of the destination site's window, because it disables such methods.

[4] If the name id_U is used for anything beyond recognition of U by W, it is considered an attribute and treated in Section 4.5, with examples in Section 9. Different roles that U wants to play with destination sites, but manage in one wallet, also belong to those sections. A person wishing to interact in different, unlinked roles with one wallet is treated as multiple users.

is large enough and the protocol uauth good enough that, as long as U only uses correct browsers, an attacker cannot achieve that W obtains an output $\texttt{uauth?}(\texttt{done}, cid, id_U)$ for a channel cid where the attacker is the other principal. Note that this assumption is not always fulfilled in practice; then every browser-based protocol fails.

4.5 Attributes and Privacy Policies

Vocabulary. If a wallet and a destination site interact, they need a common vocabulary for user attributes. We simplify this as sets Attribute_Names and Attributes of attribute names and name-value pairs. Elements of Attribute_Names may contain further information, e.g., whether an attribute is mandatory in the answer or what degree of certainty or even liability of this attribute is needed. Similarly Attributes may contain such degrees of certainty. With SAML this would be easier if the attribute names could be of arbitrary types, not only strings. We assume a function $\texttt{auth\_strength}\colon \texttt{Attributes}^* \rightarrow \texttt{Strengths}$ that computes the necessary authentication strength for a list of attributes with their desired degree of certainty.

Privacy exchange language. Wallets and destination sites need a common privacy language because user attributes can typically only be forwarded for certain purposes and with certain obligations to the recipient. We assume fixed sets Promises and Restrictions for this. For simplicity, we let Promises comprise all additional information a destination site adds about itself, e.g., certified group memberships.

Privacy evaluation functions. Wallets need an internal privacy-management system (see, e.g., [3]). The languages Promises and Restrictions will typically be derived from the internal policy language by additional bindings to actual attribute names and by elements that characterize policies as promises or restrictions. The internal privacy policies may be three-valued, i.e., the set of decisions is $\{\texttt{allow}, \texttt{deny}, \texttt{askme}\}$. Here allow and deny characterize situations where the user knows what he wants, while askme is for situations that the user does not want to think about before. Such an approach corresponds well to user studies that people do not want to start with abstract concepts like policies, but with examples. For the evaluation of privacy policies in a wallet, we therefore assume a function

$$(att, name_W, restr, rtr) \leftarrow \texttt{priv\_eval}(DB, id_U, id_D, att\_n, prom).$$

The inputs are the wallet's current datastore DB including privacy policies, the identity id_U of a registered user, the identity id_D of a destination site, a list $att\_n \in \texttt{Attribute\_Names}^*$, and promises $prom \in \texttt{Promises}$ by id_D. The outputs are

- a list $att \in (\texttt{Attributes} \times \{\texttt{allow}, \texttt{askme}\})^*$ containing the potentially permitted answers to att_n,
- a name $name_W$ under which the wallet should authenticate the attributes (see Section 9),

– their restrictions $restr \in$ `Restrictions`,
– and $rtr \in$ `Bool` denoting if a real-time release is necessary.

The evaluation function may have special properties like setting $att := ()$ if a requested mandatory attribute is denied.

Users also take privacy decisions, in particular for attributes where `priv_eval` outputs `askme`. To show what parameters the user needs and decides on, we write this as a function for each user U,

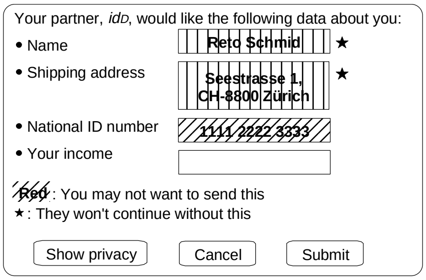
$$(att', name'_W, restr') \leftarrow \texttt{priv\_eval}_U(id_D, att, name_W, restr^*, ctxt).$$

This defines that the user sees the identity id_D of the destination site, a proposed attribute list att and name $name_W$ as output by `priv_eval` with restrictions $restr^*$, and has a context $ctxt$ in which this data request occurs. For instance, U may release more attributes to D when buying something than when only browsing. The wallet should derive overall restrictions $restr^*$ as a suitable combination of the promises and the original restrictions,

$$restr^* \leftarrow \texttt{priv\_combine}(restr, prom).$$

The user possibly modifies the attributes (in particular resolves the `askme` decisions), the name, and the restrictions.

Figure 3 shows how a wallet can present a privacy decision to the user. Where the wallet knows the attribute, it distinguishes the policy decisions `allow` (green, vertical stripes) and `askme` (red, diagonal stripes). In the example the name and shipping address are pre-authorized, while the ID number is not. The privacy restrictions are initially hidden under the button "Show privacy". The user might delete the ID number and not add an income because the fields are not mandatory. If they were, he might cancel. We define that registering for BBAE covers consent by U for W to use the BBAE protocol at all. The privacy policy must also govern all storage and usage of received information by W, but we need no notation for this because it concerns actions after the protocol. Unless the policy allows otherwise all values that W receives in a BBAE execution must be deleted at the end of this execution.



Your partner, *id*$_D$, would like the following data about you:

• Name	Reto Schmid	★
• Shipping address	Seestrasse 1, CH-8800 Zürich	★
• National ID number	1111 2222 3333	
• Your income		

Red: You may not want to send this
★: They won't continue without this

Fig. 3. Simple example form for a user privacy decision (for the real-time release, Step 8).

Concrete user attributes and privacy policies. The user deposits attributes together with privacy policies, or consents to the use of already present attributes according to certain privacy policies. We denote the data set associated with user U in wallet W by $DB_{U,W}$.

Attribute verifications. If the wallet is to confirm certain attributes beyond just storing them for the user, it has to verify them. For names, this is discussed in Section 9.

4.6 Miscellaneous Submodules

Nonces. We also need nonce generation, written

$$n \leftarrow \texttt{nonce\_gen},$$

where we omit the dependency on a security parameter k. It must have the usual cryptographic properties, and is implemented most easily as generation of a k-bit random string. SAML uses typed nonces (artifacts), but we do not distinguish such types here.

Destination site addresses. Each destination site D chooses two addresses adr_D, $ret\_adr_D \in \texttt{URLHostPath}$. They must be fixed for all executions of the BBAE protocol, at least in a certain time period.

Session administration. For a destination site D, we need a set SID_D of current BBAE session identifiers, initially empty. It is important to define how D links the three parts of a BBAE execution that have no common channels from its point of view, Step 0-3, Step 6-10, and Steps 12-13. A similar set in wallets can remain implicit.

Location query form. Each destination site D needs a location query form

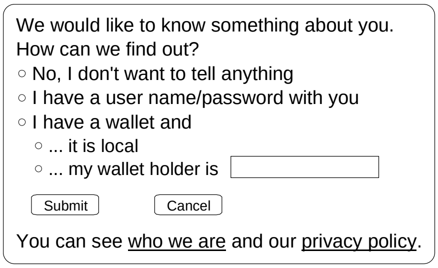
$$form \leftarrow \texttt{loc\_form}_D(ctxt)$$

for Step 1, in case it has to ask a user in person for the address of his wallet. It can be a function of D's context. An example form is shown in Figure 4. The mandatory parts are at least one “no” or “cancel”, a radio button $local \in \texttt{Bool}$ where the text “it is local” represents `true` and “my wallet holder is” represents `false`, and a text field adr after the latter button. These mandatory parts enable a consistent user experience over multiple destination sites and allow local proxies to handle location forms instead of the user. We represent the user's choices for the mandatory fields as a triple

$$(ok, local, adr) \in \texttt{Bool} \times \texttt{Bool} \times (\texttt{URLHost} \cup \{\epsilon\}),$$

where $ok = \texttt{true}$ if no “no” or “cancel” was chosen and the buttons $local$ were enabled, and $adr = \epsilon$ if the address field is empty or not in `URLHost`. The user should be able to verify whether a form $form$ has the mandatory parts, written as

$$ok' := \texttt{verify\_form}(form)$$


We would like to know something about you.
How can we find out?

- ○ No, I don't want to tell anything
- ○ I have a user name/password with you
- ○ I have a wallet and
  - ○ ... it is local
  - ○ ... my wallet holder is

You can see who we are and our privacy policy.

Fig. 4. Example form for a wallet location query (Step 1)

with $ok' \in$ `Bool`. The user will decide whether he agrees to start BBAE and which wallet to use depending on his context. We write this

$$(ok, W) \leftarrow \texttt{decideBBAE}_U(ctxt_U)$$

with $W \in Wallets_U \cup \{\epsilon\}$ and $(ok = \texttt{false} \Rightarrow W = \epsilon)$.

Request derivation. For Step 7, the destination site D has to construct an attribute-name list $att\_n \in \texttt{Attribute\_Names}^*$ that it asks for and promises $prom \in$ `Promises` that it can make. This will be based on D's context, e.g., whether U is buying something from D or trying to access employee-only web pages. We write it as

$$(att\_n, prom) \leftarrow \texttt{make\_request}_D(ctxt_D).$$

5 BBAE Step by Step

We now define the individual steps of BBAE, including parameter generation and tests. In contrast to Figure 2, which showed the error-free case, we now use different variables for values that can be different under attack, e.g., adr_W at the browser in Steps 2 and 4. We only do this per participant; variables of different participants are implicitly qualified with the participant name. Further, some variables get shorter names than in the figure for readability in protocol tables.

In addition to the participants' long-term parameters, let $ctxt_D$ and $ctxt_U$ denote the contexts of D and U in which the current BBAE execution starts.

5.1 Find Appropriate Wallet

The wallet-finding phase, Steps 1-2, provides the destination site D with the host part adr_W of the wallet address for the subsequent redirection, where $adr_W =$ `localhost` for local wallets. Possibilities for omitting this step are discussed in Section 6. The only general protocol version is that D asks the user for the wallet location. Thus these steps are (compare Section 4.6 for more explanations):

	User U		Destination site D
1a			$\mathit{dsid} \leftarrow \texttt{nonce\_gen}$; $\mathit{ctxt}_{\mathit{dsid}} := \mathit{ctxt}_D$
1b	**if** $\texttt{verify\_form}(\mathit{form}) = \texttt{false}$ **abort**;	$\leftarrow \mathit{form}$—	$\mathit{form} := \texttt{loc\_form}_D(\mathit{ctxt}_{\mathit{dsid}})$
2a	$(\mathit{ok}, W) \leftarrow \texttt{decideBBAE}_U(\mathit{ctxt}_U)$;		
2b	$l := \mathit{local}_W$; $a := \mathit{adr}_W$	—$(\mathit{ok}, l, a) \rightarrow$	**if** $\mathit{ok} = \texttt{false}$ **abort**; **if** $l = \texttt{true}$ **then** $\mathit{adr}_{\mathit{dsid}} := \texttt{localhost}$ **else** $\mathit{adr}_{\mathit{dsid}} := a$

In Step 1a, D generates a session identifier *dsid* ("D's session id") for this execution of the BBAE protocol. It is used as an index for storing the context and the obtained address. The best way for D to link Steps 0 to 3 is to already use a secure channel, but we do not prescribe this because it only influences availability. Similarly, U is only sure about its context ctxt_U if a secure channel is already used, but no serious damage can result if not.

5.2 Redirect to Wallet

In Steps 3-4, the destination site D redirects the browser to the obtained address. This is a secure redirect as defined in Section 4.1, and the address is extended with the fixed path from Section 4.4. The query string transports the destination site's backchannel address and session identifier. Thus Step 3 works as follows:

	Browser		Destination site D
3a			$a' := \mathit{adr}_{\mathit{dsid}}$; $\mathit{path} := \texttt{BBAEpath}$; $\mathit{SID}_D := \mathit{SID}_D \cup \{\mathit{dsid}\}$
3b		$\leftarrow \texttt{redirectS}(a', \mathit{path}, \mathit{query})$—	$\mathit{query} := (\mathit{adr}_D, \mathit{dsid})$

The browser reacts by establishing a secure channel and sending the path and querystring over it. We call the channel identifier *bwid* for browser-wallet id. A message with $\mathit{path} = \texttt{BBAEpath}$ triggers a BBAE execution at the wallet. As we only specify BBAE here, not the dispatching, we only abort if the path is wrong. The wallet chooses a local session identifier *wsid* ("W's session id") if BBAE really starts.

	Browser		Wallet W
4a	$\texttt{secchan!}(\texttt{new}, a')$	$\rightarrow$	$\texttt{secchan?}(\texttt{new}, \mathit{bwid}, a')$
4b	$\texttt{secchan?}(\texttt{accepted}, \mathit{bwid}, a', \mathit{idw})$	$\leftarrow$	$\texttt{secchan!}(\texttt{accept}, \mathit{bwid}, \mathit{id}_W)$
4c	$\texttt{secchan!}(\texttt{send}, \mathit{bwid}, (\mathit{path}, \mathit{query}))$	$\rightarrow$	$\texttt{secchan?}(\texttt{receive}, \mathit{bwid}, (\mathit{path}, \mathit{query}))$
4d			**if** $\mathit{path} \neq \texttt{BBAEpath}$ **abort**; $\mathit{wsid} \leftarrow \texttt{nonce\_gen}$; $(a_{\mathit{wsid}}, \mathit{dsid}_{\mathit{wsid}}) := \mathit{query}$; **if** this fails **or** $a_{\mathit{wsid}} \notin \texttt{URLHostPath}$ **abort**

The notation *idw* indicates that this is a wallet identity, but not yet known to be that of a specific wallet W known as id_W to U.

5.3 Authenticate User

After successful execution of Step 4, the wallet authenticates a user over the established secure channel. With the notation from Section 4.4, Step 5 is defined as follows:

	User U		Wallet W
5a	$\texttt{uauth?}(\texttt{start}, \mathit{idw})$;	$\leftarrow$	$\texttt{uauth!}(\texttt{start}, \mathit{bwid})$
	if $\mathit{idw} \neq \mathit{id}_W$ **abort**		
5b	$\texttt{uauth!}(\texttt{do}, \mathit{id}_W, \mathit{login}_{U,W})$	$\rightarrow$	$\texttt{uauth?}(\texttt{done}, \mathit{bwid}, \mathit{idu}_{wsid})$;
			if $\mathit{idu}_{wsid} = \epsilon$ **abort**

In Step 5a, *idw* is the identity that the browser obtained in Step 4b. The user remembers from Step 2a which wallet W he wanted to use and aborts if the secure channel is with an identity other than id_W.[5] In Step 5b, the user sends login information, and the wallet derives a registered identity or aborts.

5.4 Request

The wallet sets up a backchannel by contacting the destination site at the address obtained in Step 4d, and transmits the destination site's session identifier. In the concrete version, this is done in the query string of the HTTPS request that triggers the SSL/TLS channel. The wallet also transmits a fresh nonce handle_{wsid} that is mainly needed to link Steps 9 and 12. This is due to a SAML idiosyncrasy. In SAML, this nonce is called an artifact.

	Wallet W		Destination site D
6a	$\texttt{secchan!}(\texttt{new}, a_{wsid})$	$\rightarrow$	$\texttt{secchan?}(\texttt{new}, \mathit{wdid}, \mathit{adr}_D)$;
6b	$\texttt{secchan?}(\texttt{accepted}, \mathit{wdid}, a_{wsid}, \mathit{idd}_{wsid})$	$\leftarrow$	$\texttt{secchan!}(\texttt{accept}, \mathit{wdid}, \mathit{id}_D)$
6c	$\mathit{handle}_{wsid} \leftarrow \texttt{nonce\_gen}$		
6d	$\texttt{secchan!}(\texttt{send}, \mathit{wdid}, (\mathit{dsid}_{wsid}, \mathit{handle}_{wsid}))$	$\rightarrow$	$\texttt{secchan!}(\texttt{receive}, \mathit{wdid}, m)$;
			$(\mathit{dsid}, \mathit{handle}_{dsid}) := m$;
			if this fails **or** $\mathit{dsid} \notin \mathit{SID}_D$ **abort**

The destination site D only reacts (at least with Step-6 behavior) if the address a_{wsid} is adr_D, as denoted implicitly in Step 6a; then it identifies itself under id_D. If a message then arrives over this channel, its first parameter must be the session identifier *dsid* of a current BBAE execution. Then D can use *dsid* to retrieve information from this protocol execution. It first derives what attributes it wants to ask for, and then sends the request together with the received nonce handle_{dsid} over the secure channel.

	Wallet W		Destination site D
7a			$(\mathit{att\_n}, \mathit{prom}) \leftarrow \texttt{make\_request}_D(\mathit{ctxt}_{dsid})$
7b	$\texttt{secchan!}(\texttt{receive}, \mathit{wdid}, (\mathit{att\_n},$	$\leftarrow$	$\texttt{secchan!}(\texttt{send}, \mathit{wdid}, (\mathit{att\_n},$
7b	$\mathit{prom}, \mathit{handle\_d}))$		$\mathit{prom}, \mathit{handle}_{dsid}))$
7c	**if** $\mathit{handle\_d} \neq \mathit{handle}_{wsid}$ **abort**		

[5] Recall that unless we have a secure channel already, there is no unambiguous link to Step 2a for the user, but only availability is affected if the user assumes another wallet W' here.

In the SAML version, the message in Step 7b is a SAML request containing exactly one SAML attribute query, and $handle_{dsid}$ is the artifact in that. The wallet aborts if this is not the handle it expects on the channel *wdid*. The wallet may now mark $handle_{wsid}$ as used so that Step 7c will abort if repeated as a backup security measure.

5.5 Deriving a Response

If Step 7c was passed, the wallet tries to derive a response. It first uses the privacy evaluation function from Section 4.5; it has all the necessary inputs in the current BBAE execution. Then, if necessary, it continues with a real-time release as explained in Section 4.5.

	User U		Wallet W
8a			$(att, name_W, restr, rtr) \leftarrow \texttt{priv\_eval}(DB, idu_{wsid}, idd_{wsid}, att\_n, prom)$
8b			**if** rtr **then**
			$restr^* \leftarrow \texttt{priv\_combine}(restr, prom)$;
	$\texttt{secchan?}(\texttt{receive}, id_W, (idd, att, name_W, restr^*))$	$\leftarrow$	$\texttt{secchan!}(\texttt{send}, bwid, (idd_{wsid}, att, name_W, restr^*))$
8c	$(att', name'_W, restr') \leftarrow \texttt{priv\_eval}_U(idd, att, name_W, restr^*, ctxt_U)$		
8d	$\texttt{secchan!}(\texttt{send}, id_W, (att', name'_W, restr'))$	$\rightarrow$	$\texttt{secchan?}(\texttt{receive}, bwid, (att', name'_W, restr'))$
			else $(att', name'_W, restr') := (att, name_W, restr)$

The wallet uses the attributes *att'* and restrictions *restr'* as the main part of its response to the destination site. It adds the name idd_{wsid} of the destination site as a measure against man-in-the-middle attacks and the nonce $handle_{wsid}$. It authenticates these elements under $name'_W$ with the appropriate strength. (Recall Section 4.5 and that typical choices of this name are discussed in Section 9.) Similar to Step 6, we include a new fresh nonce $handle'_{wsid}$ in case Steps 7 and 9 will be repeated. The destination site makes the natural verifications. If it decides not to repeat Step 7, it tells the wallet the return address ret_adr_D for the browser.

	Wallet W		Destination site D
9a	$str \leftarrow \texttt{auth\_strength}(att')$; $sig \leftarrow \texttt{auth}(name'_W, (att', restr', idd_{wsid}, handle_{wsid}), str)$; $handle'_{wsid} \leftarrow \texttt{nonce\_gen}$		
9b	$\texttt{secchan!}(\texttt{send}, wdid, (sig, handle'_{wsid}))$	$\rightarrow$	$\texttt{secchan?}(\texttt{receive}, wdid, (sig, handle'_{dsid}))$
9c			$(name_W, m, str) \leftarrow \texttt{test}(m')$; **if** $(name_W, m) = (\epsilon, \epsilon)$ **abort**; $(att, restr, idd, handle^*) := m$; **if** this fails **or** $idd \neq id_D$ **or** $handle^* \neq handle_{dsid}$ **abort**;
10	$\texttt{secchan?}(\texttt{receive}, wdid, ret\_adr)$	$\leftarrow$	$\texttt{secchan!}(\texttt{send}, wdid, ret\_adr_D)$; $SID_D := SID_D \setminus \{dsid\}$

We did not formalize how D decides if it wants to repeat Step 7, e.g., because it is not satisfied with the attributes, and how *handle'* then becomes *handle*. All other decisions that D bases on the attributes have no consequence in BBAE and are thus not formalized either, but D must delete attributes if it cannot fulfill the restrictions for them. The channel *wdid* can now be released.

In the SAML version, Step 9 is a POST of a SAML response, where $handle_{wsid}$ is a SAML artifact included as SAML subject confirmation, idd_{wsid} is the SAML recipient element, $name'_W$ the issuer element, and the strength *str* is only implicit in the presence or absence of an XML signature and its type.

5.6 Redirect Back

If the wallet obtains a Step-10 message, it redirects the browser back to the destination site with $handle_{wsid}$ as a parameter. The redirect is sent within the secure channel with the identifier *bwid*, and the redirect must also trigger a secure channel.

	Browser		Wallet W
11			$(ra, path) := ret\_adr$;
	$\texttt{secchan?}(\texttt{receive}, bwid,$ $(\texttt{redirectS}(ra, path, handle_{wsid})))$	$\leftarrow$	$\texttt{secchan!}(\texttt{send}, bwid,$ $(\texttt{redirectS}(ra, path, handle_{wsid})))$

This has the following effect:

	Browser		Destination site D
12a	$\texttt{secchan!}(\texttt{new}, ra)$	$\rightarrow$	$\texttt{secchan?}(\texttt{new}, bdid, ra)$
12b	$\texttt{secchan?}(\texttt{accepted}, bdid, ra, id_D)$	$\leftarrow$	$\texttt{secchan!}(\texttt{accept}, bdid, id_D)$
12c	$\texttt{secchan!}(\texttt{send}, bdid, (path, handle_{wsid}))$	$\rightarrow$	$\texttt{secchan?}(\texttt{receive}, bdid, (path, handleb))$
12d			**find** *dsid* **with** $handle_{dsid} = handleb$

That D inputs id_D for this channel helps the user keep track of the channel after the BBAE protocol. D uses *handleb* to retrieve the corresponding session identifier *dsid* and thus also the context $ctxt_{dsid}$ and the response from Step 9 of the protocol execution with *dsid*. This finishes the BBAE protocol. Typically D will internally redirect the browser to its original target URL, stored in $ctxt_{dsid}$, with parameters from the response.

6 Optimizing the Number of Steps

In this section, we sketch why we believe the steps in BBAE are optimal for the general case, and in which cases steps can be omitted. As in the entire paper, we consider the situation where the user is initially browsing at the destination site. A portal scenario, where the user starts at the wallet and addresses destination sites via special links, only starts with Step 4.

The two redirects, Steps 3-4 and 11-12, are common to all known browser-based protocols. They seem unavoidable if the user and the browser are unknown at the destination site and the browser contains no user-related information.

Steps 1-2, i.e., asking the user for the wallet location, is optional in BBAE. It is necessary in the browser-stateless case, i.e., when neither the destination site nor the browser knows anything about the user. It is also needed if a user has several wallets without clear separation of responsibilities that could be evaluated automatically. Multiple unlinked wallets are, e.g., Microsoft's proposal for handling multiple roles; only in Passport they are all at the same address. With a standard browser and free choice of wallets, and without an overall cookie domain, the steps are also needed at least once per user and destination-site domain. For the other cases, implementers may choose any state-of-the-art techniques of state keeping, such as cookies or browser extensions. The steps then either disappear completely into Step 0, or do not go to the user.

Step 5 is clearly needed if no prior secure channel is available. Further, reusing an existing channel can be dangerous if only cookies are used for session maintenance and in the mobility scenario; compare attacks on Passport [19]. Step 5 cannot be joined into the real-time release (Step 8) if that becomes necessary, although one can postpone it until after Step 7: Before the real-time release, a remote wallet must know whose attributes it is looking up, and every wallet must authenticate its user before showing privacy-critical attributes on his or her screen for releasing.

The backchannel (Steps 6-10) can be omitted if both request and response are short enough to fit into a URL. As a protocol extension, the destination site could indicate this in Step 3, i.e., immediately add a request there and even omit the address adr_D if it does not expect a long response. However, if one adheres to the HTTP recommendation of at most 255 bytes, this does not often work except in small federations with a-priori exchanged symmetric keys, because signatures and certificates are too long. An alternative to backchannels for long data are POSTs. However, this requires user interaction between Step 3 and 4 and between Step 11 and 12, or active content for cross-domain posting, which is not zero-footprint. The inconvenience of user interaction can only be avoided by integrating it with something meaningful for the user. We do not see such a possibility for the first redirection. The second one can be integrated with the real-time release where that is present, but this greatly reduces flexibility, e.g., for multi-step interaction and for making privacy choices also for future similar requests. Hence we believe a backchannel is the better solution. It may also often be faster.

Steps 6 and 10 are needed because we build the backchannel from the wallet side. This is done to enable anonymous local wallets. Those can be addressed as "localhost" by the destination site in Steps 3-4, but the destination site cannot build a backchannel to them.

7 Multi-wallet Extensions

Deriving a response between Steps 7 and 9 may involve a second wallet that holds or confirms certain attributes. We sketch the four main options. (No prior proposal contains confirmed attributes.) If the second wallet has to confirm a long-term attribute for a certain name, e.g., the year of birth, it can give the first wallet a confirmation token, e.g., a SAML attribute assertion, with an appropriate validity period. This becomes an attribute in the first wallet. If the confirmation must be fresh and the second wallet's policy allows release to the first wallet, the first wallet can fetch the confirmation on a backchannel. For a real-time release at the second wallet, the first wallet must redirect the browser to it. If the wallets are not linked, the first wallet only answers what it can and the destination site has to restart the protocol, asking the user for the appropriate wallet for the remaining attributes.

8 Security

The main issue with a security protocol should be security. Nevertheless, all prior protocols for federated identity management come with almost no published security justification. (The most detailed one is that of SAML, but it is still an attack-by-attack listing of countermeasures taken, not an overall analysis.) Further, they do not visibly follow robust-design principles [1,2]. Indeed, the first version of Passport was vulnerable to a man-in-the-middle attack, and the current version is not as secure against that as it could be; vulnerabilities were also found in one of the original Liberty protocols (but not a browser-based one) and in permitted SAML instantiations [15,6]. Earlier attacks on Passport were found in [9,19]. They are mostly weaknesses of the environment, not of the protocol itself. Some of them apply to all browser-based protocols, and one has to keep in mind that this is a pretty vulnerable environment. In particular, online password guessing always works for remote wallets, fake wallet screens are a danger, and the protocol cannot get more secure than the underlying operating system and the browser. A brief security proof sketch for BBAE can be found in the preliminary version [13], but for a far less rigorous definition of BBAE. While one of our original motivations of the current detailed definition was to enable a more rigorous proof, we have not done this so far.

9 Names, Keys, and Certificates

We now recommend how to use identifiers (names) of users and names, keys, and certificates of wallets in the most important scenarios. Our abstract arbitrary authentication under an arbitrary name $name_W$ leaves that open, and so do concrete SAML responses, which have an arbitrary issuer $name_W$ and optional XML signatures. We concentrate on the general consumer scenario with many wallets and destination sites. Additionally, a wallet can use id_D to first look up whether it has a closer relationship with the destination site, e.g., a joint secret key, for more efficient authentication.

Form-filling case. The most usual case for initial e-commerce does not require authentication, because it replaces current forms that the user fills in without confirmation, e.g., with demographics and preferences. Even shipping addresses and payment information belong to this case, because the payment information is verified with existing payment systems. Then no wallet name and authentication are necessary. For a really anonymous response, a local wallet should choose a nonce as $name_W$ and make no signature.

Real identity. For authenticity under a name id_U^* with prior meaning, e.g., a legal name or an email address, this name must be verified in registration. Authenticity cannot get better than this verification, and all parties trusted by a destination site to confirm this name must be honest. (This is the same tradeoff between security and convenience as with PKIs; it has nothing to do with whether only a password is exchanged in registration or a public key.) Users with a local wallet have to get a public key *pk* of the wallet certified for id_U^*. The local wallet thus becomes a leaf in a tree of certification authorities for a specific, externally managed name space, where it may only confirm one or a few names.[6]

Long-term role. If a wallet generates a role name *role* for repeated use (e.g., with a certain group of enterprises), but without verified prior meaning or with the explicit desire to keep this role unlinkable to other roles, it generates a new name. (SAML explicitly allows this.) This is like SPKI/SDSI compared with classical PKIs. A remote wallet holder can issue responses for *role* under its own fixed name $name_W$ and with a fixed key *pk* without endangering unlinkability if its user community is large enough to offer anonymity. If the wallet is local and the role should really be unlinkable, it should generate a fresh name $name_{role}$ and key pk_{role} for this role. For XML signatures, the wallet can include this key in the KeyValue element when first using it with a destination site. The name *role* should be considered to be in a name space governed by *pk* or pk_{role}, respectively. Then only the wallet that issued *role* must be trusted for authenticity in this role.

To distinguish these cases precisely for the destination site, one should apply authentication context definitions similar to [10] to individual attributes in a response. For instance, one could represent the Passport case by stating that the user ID is a single long-term role, that names and payment information have not been verified, while control (but not ownership) of the email address has been verified.

10 Conclusion and Outlook

We have shown that browser-based attribute exchange and even the simpler single signon, although a type of three-party authentication, requires new protocols and poses new challenges for secure design. We presented a new protocol BBAE which, for the first time, combines all important privacy properties that can be achieved under the design constraints. Its efficiency is essentially the same as of other browser-based protocols.

A general trend in the protocol design for influential standards and products in this area is a multi-layered approach: One starts with a very general message format, then extends and refines it, then designs a small core protocol, and

[6] The wallet may use id_U also as its name *issuer*; this simplifies the use of X509 certificates. New XML tokens as certificates can be more flexible.

then extends that again. SAML already takes this approach to a certain extent. Liberty adds about one layer of extension between each layer of SAML. The WS-Federation proposals take this approach even further. The attractiveness for quick standardization is obvious. However, the challenges for finally getting secure protocols are large: First there is the simple practical issue that "general processing constraints" are spread over all layers, and both a security analyst and a later implementer may miss some of them or get them wrong. Secondly, the initial general message formats have no clear semantics, and even the core protocols are not always full protocols. Hence from a security point of view, the standards cannot currently be considered modular, because the first layer for which one can specify and prove usual security goals is typically the highest one. It will be a challenge to the security-research community to adapt to this trend.

Acknowledgements

We have benefited from discussions with many colleagues, in particular Kathy Bohrer, Peter Buhler, Peter Capek, Chris Giblin, Thomas Groß, Heather Hinton, John Hind, Stephen Levy, Matthias Schunter, and Jay Unger, and with Daniel Lutz.

References

1. M. Abadi, R. Needham: Prudent Engineering Practice for Cryptographic Protocols; IEEE Transactions on Software Engineering 22/1 (1996) 6–15
2. R. Anderson, R. Needham: Robustness Principles for Public Key Protocols; Crypto 95, Springer-Verlag, Berlin 1995, 236–247
3. K. Bohrer, X. Liu, D. Kesdogan, E. Schonberg, M. Singh, S. Spraragen: Personal Information Management and Distribution; 4th Intern. Conf. on Electronic Commerce Research (ICECR-4), Dallas, 2001
4. D. Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030–1044
5. J. Camenisch, E. Van Herreweghen: Design and Implementation of the Idemix Anonymous Credential System; 9th ACM Conference on Computer and Communications Security (CCS), 2002, 21–30
6. Thomas Groß: Security Analysis of the SAML Single Sign-on Browser/Artifact Profile; 19th Annual Computer Security Applications Conference (ACSAC 2003), IEEE Computer Society Press, December 2003
7. IBM Consumer Wallet; White Paper, 1999 (first release 1997), `http://www-3.ibm.com/software/webservers/commerce/payment/wallet.pdf`
8. IBM: Enterprise Security Architecture using IBM Tivoli Security Solutions; April 2002, `http://www.redbooks.ibm.com/abstracts/sg246014.html`
9. D. P. Kormann, A. D. Rubin: Risks of the Passport Single Signon Protocol; Computer Networks 33 (2000) 51–58
10. Liberty Alliance Project: Liberty Phase 2 Final Specifications, November 2003, http://www.projectliberty.org/specs/lap-phase2-final.zip (v1.0 July 2002).
11. Microsoft Corporation: .NET Passport documentation, in particular Technical Overview, Sept. 2001, and SDK 2.1 Documentation (started 1999); `http://www.passport.com` and `http://msdn.microsoft.com/downloads`

12. Passlogix: v-Go Single Signon; White Paper, 2000 (first release 1999), http://www.passlogix.com/media/pdfs/usable_security.pdf
13. B. Pfitzmann, M. Waidner: BBAE – A General Protocol for Browser-based Attribute Exchange; IBM Research Report RZ 3455 (#93800) 09/09/02, http://www.zurich.ibm.com/security/publications/2002/
14. B. Pfitzmann, M. Waidner: Privacy in Browser-Based Attribute Exchange; ACM Workshop on Privacy in the Electronic Society (WPES) 2002, ACM Press 2003, 52–62
15. B. Pfitzmann, M. Waidner: Analysis of Liberty Single-Signon with Enabled Clients; IEEE Internet Computing 7(6) 2003, 38–44
16. Roboform: Free Web Form Filler and Password Manager; first release 1999, http://www.siber.com/roboform/.
17. Security Assertion Markup Language (SAML); OASIS Standard, Nov. 2002, http://www.oasis-open.org/committees/security/docs/
18. Shibboleth-Architecture Draft v05; May 2002 (v01 in 2001), http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-arch-v05.pdf
19. M. Slemko: Microsoft Passport to Trouble; Rev. 1.18, Nov. 2001 http://alive.znep.com/ marcs/passport/
20. BEA, IBM, Microsoft, RSA Security, VeriSign: WS-Federation: Passive Requestor Profile; Draft, Version 1.0, July 2003, http://www-106.ibm.com/developerworks/webservices/
21. HTTP Over TLS; Internet RFC 2818, 2000
22. XML-Signature Syntax and Processing; W3C Recommendation, Feb. 2002, http://www.w3.org/TR/xmldsig-core/
23. Zeroknowledge: Freedom Personal Firewall; first release 1999, http://www.freedom.net/products/firewall/index.html

Federated Identity-Management Protocols
(Transcript of Discussion)

Birgit Pfitzmann

IBM Zurich Research Lab

The question was, where have all the protocols gone? One answer, at least where authentication protocols might go, is what's called federated identity management protocols.

This is a fairly vague term, and essentially I mean protocols like Microsoft Passport[1] and the Liberty Alliance[2] protocols. I have in the paper a complete protocol proposal which I think is better than those ones, called BBAE.

You want to look at privacy, security, and access control points, and analyse these protocols, but they have quite a lot of limits. They contain useful things like operations security, browsers and passwords, and operating systems. And from the privacy point of view, you can actually have very good privacy with them for the typical case where there's no certification. If you want them together with the certifier, that can't be done without something that makes credentials, and as I said these sorts of problems raise analysis challenges.

Pekka Nikander: This reminds me of the design of the IKE exchange. This sounds like the same kind of approach, where people are reluctant to commit themselves into a specific protocol.

Michael Roe: Related to that, in ISO you had layers upon layers of encrypted bits. This didn't actually work, so we had yet another layer that says, there's some packet identifier and an opaque bitstring, but the Liberty Alliance design was going to be yet another identifier with Bits in it. This just progresses up the stack, and the stack gets taller and taller, but...

Bruce Christianson: It's going to get worse and worse. You talked about the problems, but some of them are features. If someone using the browser doesn't want their actions to be linked, for example if they don't want market research companies to be able to correlate their actions, then you have pointed out very clearly what barriers there are to that work in the current type of architecture you propose. What happens if we wanted to design an architecture that didn't have those limits, can we do it, or do we have to start again?

Reply: Well, I think it really depends on what is certified. If they don't have certified attributes then this protocol can do perfect privacy, with one exception which is that this wallet learns the names of the destination sites when users go in. But the protocol allows users to keep their work local. So you could say you

[1] See http://www.passport.com.
[2] See http://www.projectliberty.org.

use the same protocol as these other people who insist on a zero footprint and don't want to download something into their own wallet, and that way you can keep it anonymous. This is one of the things that is different. If we use a bad channel we connect this way round because we want to get to the destination site anyway, despite the man in the middle attacking these tickets. In this way if the wallet is open then you don't need to give out an address, so you can use it over again. Whatever anonymity the analyser gives you, you need to have destination sites that won't analyse the addresses.

Bruce Christianson: Yes, but then what does certification really mean in this context? I guess that's what it comes down to, what's the nature of certification?

Reply: At some point this site asks some requests, and here is an answer. Now, even with a single packet of Passport, they're not actually guaranteeing anything in this answer except that the person controls that address.

If this is an open wallet, an open wallet doesn't sign, or signs under a pseudonym or something. If it's something like a credit rating, for example, then you need a signed response, and you can't handle this from the open wallet. If you have an open wallet then you need some other wallets for those attributes that need to be certified. The user chooses which one is used. There is now some transformation in the browser so these switch points would go over and they would know that they had opened the same person. And that's what cryptographic credentials can do, the bank just puts out some cryptographic credentials on some credit rating and it transforms the pseudonyms into fact lines. But this is the only thing you can't do with an open wallet.

Bruce Christianson: But there's no reason why you couldn't have different semantics which allow the bank to give a credit rating to an anonymous entity.

Reply: Yes, it doesn't need to be a real name, but they would need to use the wallet at that moment.

In principle I agree, but I think there are not that many cases where you want use this sort of protocol for both anonymity and certified transactions at the same time. So for most cases, if you only want anonymity, or you only want certified attributes, then you can always use this protocol.

Ross Anderson: I still don't see the advantages for a destination site. I can't in particular see any attraction in lending money to pseudonyms because that would be illegal in the EC under the Money Laundering Regulations. Otherwise I could guarantee a loan for ten million pounds to someone who turns out to be a drug dealer.

What benefit do you get in general from federated identity management protocols if you're a merchant? If you're a merchant and you're dealing with Microsoft you have access to the growing network of customer profile data being collected, but if you have federated identity management then as a merchant you're probably better off simply collecting customer profile data yourself.

Reply: I don't think the business case is in the shopping scenario, I fully agree. Anyway only two percent of the actual password users use Passport because they like it, so on the user side it's definitely not a market. But in the business scenario I have been working on, it's simply the cost of registration. For example if you merge two companies, or if you have a supplier change, then given the goals they have, that is the costly part of identity management. They would like to say, "OK, this is another company's database, these people are already registered there for some purpose, and they're better registered than we can register them right now." It might be the travel agent that IBM lets me use that has the database. So at the moment we've lots of concerns there, like can they get my internal password? The travel agents seem to have my employee number, my phone number, and all sorts of things. So there it might actually help to only do a case-by-case transfer of the evidence that is actually needed, instead of sharing databases.

Ross Anderson: I think this gets us to the point. I suppose that the wallet is acting as a kind of broker. For example I, as the author of a book[3], would like to know the names and email addresses of everybody that's bought a copy of my book, so that I can go and ask them whether it's the right time to produce a second edition. It would be convenient for me if there was some service that would manage that for me, so that I could go to the service and say, here's $100, please give me a random selection of a hundred email addresses from people who bought my book, so that I can ask this question. Providing privacy in such a circumstance doesn't seem to make very much sense at all because the identity management system is essentially being used as a marketing tool.

Reply: In your application yes, I think it is. But in general, it depends. I think they would all like to exchange even more attributes than they do at the moment. But, in things like supply chains, the companies also have a certain interest, so the destination sites might not want to do the registration themselves. And then of course there are things like regulations, and they have heard all these bad statistics. I must admit that I don't know any easy ways where anonymous protocols have a business case. Whenever you interact with somebody, you could get a specific pseudonym, and whenever you contact that destination site again you're automatically identified under the same pseudonym, but it's different for different destination sites. So that's the Liberty standard policy, you just have to add identification. In contrast, with Passport they have a global identifier that appears in all the chains.

[3] Ross J. Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems", Wiley 2001, by the way.

Enforcing the Unenforceable

Peter Y.A. Ryan

University of Newcastle upon Tyne, Newcastle upon Tyne, UK

Abstract. A security policy is intended to regulate the behaviour of a socio-technical system (computers, networks and humans) in such a way as to ensure that certain properties are maintained or goals achieved. Two problems arise here: regulating the behaviour of humans is non-trivial and, secondly, many security goals are not "enforceable" in the Schneider sense, [1]. Thus, security policy mechanisms inevitably involve approximations and trade-offs.

We discuss the theoretical and practical limitations on what is technically enforceable and argue for the need for models that encompass social as well as technical enforcement mechanisms.

1 Introduction

In [1], Fred Schneider introduces the notion of an enforceable policy. Intuitively, such a policy is one that can be enforced by blocking offending actions. More formally, it is a policy that can be enforced by an Execution Monitor (EM), a conceptual device that runs in parallel with the system and observes relevant actions of the target system and blocks any that would violate the property or policy. An EM thus simulates the policy and has no look-ahead or look-sideways capability (in particular, it has no model of the target system). Essentially, an EM can enforce trace (or safety) properties, in particular, access control policies.

The EM concept can be expressed neatly in process algebra terms, for example in CSP. Given any safety property ϕ and system S we can find an execution monitor EM_ϕ such that:

$$EM_\phi \;\; sat \;\; \phi$$

and so, automatically:

$$S||EM_\phi \;\; sat \;\; \phi$$

where || is the parallel (lock-step) composition operator of CSP.

A trace/safety property defines those behaviours that are regarded as acceptable or safe, and so can be expressed as a set of traces. To verify that a given system S satisfies a given safety property, it suffices to determine that the trace set $\tau(S)$ of S is a subset of the set of safe traces. This of course assumes that we have an accurate model of the system.

To check that a system maintains a safety property at run time, we can simply monitor its actions and check that the trace remains in the set of safe traces. As soon as an action a occurs that takes the trace outside the set of allowed traces, the property will have been violated and the action a will be regarded as a failure with respect to the policy.

Many security and privacy requirements are not enforceable in this sense. That is to say, they cannot be stated as predicates over behaviours (traces) of the target system. Consequently, it is not possible to find an EM such that, composing it in parallel with a system will guarantee that the resulting composed system will satisfy the policy.

Besides such theoretical limitations on what is enforceable, there may be practical limitations. Some goals may be theoretically enforceable and yet may be more appropriately or effectively enforced by social rather than technical mechanisms, e.g., dealing with exceptions. The policy and the technical enforcement mechanisms can easily become unwieldy if we try to enforce too much technically.

2 Theoretical Limits to Technical Enforcement

2.1 Availability

Availability (at least in the eventually sense) is unenforceable, in 2 senses:

– There will be no point in a single execution at which the system can be deemed to have failed.
– It requires the occurrence of actions- an EM can only prevent the occurrence of actions and so cannot force the system to perform actions.

If we observe an execution of S, there will never be a point in the execution at which we can say that the availability property has been violated. We cannot determine that a system will never perform a certain action by observing it for some finite time: there will always be the possibility that the required action will be performed at some later time.

Consequently, for an availability property ψ we cannot find an EM that will convert a system that does not satisfy ψ into one that does.

On the other hand, if we adopt a time-limited notion of availability, we can express this as a trace property and detect a violation at run time: if during some execution a service is requested but not provided within the requisite time limit then, as soon as the time limit is reached, the availability property will be deemed to have been violated.

Note that even though time-limited availability can be characterised as a trace property, it still cannot be enforced by an execution monitor. An execution monitor, by definition, can only block actions and cannot force the target system to perform an action.

We can characterise an availability failure with respect to a given execution if we assume a god-like ability to view the possible future unfoldings of the execution. In this case we can determine that a certain action will never be available. More realistically, if we assume a detailed understanding of the construction of the system we may be able to determine that, having reached a certain state, it will never be able to execute a certain action. This however would require an EM with a look ahead capability, specifically excluded by definition.

2.2 Confidentiality and Information Flow

Confidentiality and (some flavours of) integrity are also not enforceable, but for different reasons: information flow is a property of the ensemble of possible executions of the target system. It can for example be expressed as a predicate on $\mathcal{P}(traces)$ or in terms of process equivalence. As a result, we cannot detect a violation from observation of a single execution of the target system. This would be a little like trying to determine whether a given number is "random".

In modelling terms we can characterise a failure of confidentiality in trace terms as the appearance of sensitive data in the adversary state but not purely in terms of the state of the target system. Similarly, certain classes of integrity failure can be thought of as a mismatch of the data with reality, i.e., the state of the environment. This too cannot be captured purely in terms of observations of the target system.

For information flow properties, even the ability to foresee the future is insufficient to characterise a failure given only the ability to monitor the actions of the target system. To establish whether a system will maintain confidentiality we must look at the set of all its possible behaviours and establish whether this set as a whole has certain characteristics.

To make this clearer, consider a simple example: a system that takes in sensitive plaintext and outputs this encrypted with a one-time-pad over an open channel. This will be perfectly secure as long as this really does implement a one-time-pad, i.e., is able to generate any of the possible key streams (unpredictably and with equal probability). If the system malfunctions in such a way that it will in fact only generate one of a small number of the possible key streams, this would constitute a security failure but would not be detectable by observing any single execution.

Again, it is useful to turn to the concept of an execution monitor. We cannot, in general, find an execution monitor that can constrain a given system in such a way as to ensure that it will satisfy a given information flow property. An information flow property has to be designed in at the outset. To return to the stream cipher example, the algorithm has to be designed and verified to be effectively indistinguishable from a random process. No execution monitor will convert a flawed crypto device into a secure one.

This observation perhaps goes some way to explain the emphasis that has traditionally been placed on verification in the security community, as contrasted with the emphasis on fault tolerance in the dependability community.

The difficulty with verifying that a system satisfies a given security property is that it is heavily dependent on our having an accurate model of the system (and perhaps of its environment). Typically this is extremely difficult to achieve: in order to be tractable, models are necessarily abstractions of the real system. This abstraction process harbors possibilities of losing significant aspects of the system behaviour that might undermine the security requirement (e.g. EM radiation, timing, power consumption, characteristics of underlying crypto primitives that have been abstracted away etc). Furthermore, even supposing that our system model starts off faithful to the security properties, there is always the possibility that with time, bit-rot, subversion, upgrades etc, that the system will evolve away from the initial design description.

This means that we cannot rely on an initial, compile-time verification to be confident that a system maintains a security requirement. Verification must be augmented be monitoring, but with the observation that it may be very difficult to detect certain violations of a security property.

3 The Human Element

The above discussion applies mainly to the purely technical aspects of a system. Typically, humans will also have an essential role to play in the behaviour of any reasonably complex system. A security policy must address the behaviour of the humans as well as that of the technical component of the system. To some extent the behaviour of the humans can be controlled by technical means: where action of the humans need the cooperation of the machine, e.g., the sending of a sensitive e-mail. Other aspects cannot be so controlled, e.g., pillow talk.

Where the human behaviour cannot be regulated by technical mechanisms we must resort to social mechanisms: trust in professional conduct, perhaps backed up by audit and random checks etc. Often a mix of technical and social mechanisms will be used. Note also that the envelope of what is technically enforced can be widened, for example by enforcing n-person rules (or, to take a rather different context: by employing eunuchs in the harem). Here too we have to be careful: our technical enforcement mechanisms can quickly become completely unweildy and unusable. In this case it might be counterproductive, simply encouraging workarounds etc. A further point is that there may be situations in which human judgement and intiative may be essential to achieving the goals.

To date, there seems to be very little research into the relative effectiveness of technical and social enforcement mechanisms, nor into their interaction. Thus we have no systematic way of evaluating the relative merits of various trade-offs between technical and social mechanisms and, consequently, no well-founded way of determining optimal balances and mixes of technical and social enforcement mechanisms.

4 The Execution Monitor Approximation

Nonetheless the EM concept and its physical embodiment as a security kernel enforcing access controls are very appealing: easy to understand, implement and analyse. The EM need only simulate the policy so, assuming that the policy is reasonably simple, the EM can be a fairly simple and easily verifiable piece of code.

Thus, for exmaple, integrity requirements are often replaced by access control mechanisms: this shifts the problem of defining what valid means to depending on the honesty and competence of authorised users. In other words, we use social mechanisms such as trust and accountability to fill in the gap between the goals and what is technically enforceable.

5 Conclusions

Many security and privacy goals are not enforceable (at least in the Schneider sense). Furthermore, many security properties have the additional characteristic that failures (violations of the property) during run-time may be very difficult to detect, let alone enforce. As a result security failures may be very difficult to contain and recover from.

Typically we try to technically enforce security goals as far as is feasible and fill in the gaps with social mechanisms. Often technical and social mechanisms are combined, e.g. N-person rules. The arguments that the technical and social mechanisms uphold the goals involve assumptions and approximations and how exactly we bridge the gap between the enforceable and the unenforceable is still poorly understood. Furthermore, we do not have good models of how social mechanisms (trust, accountability, separation of duty) operate and interact with the technical mechanisms. Consequently, the problem of striking the optimal balance (and mix), from the point of view of security, usability etc, between technical and social enforcement mechanisms remains an open area for research.

Reference

1. Schneider, F. B.: Enforceable Security Policies. ACM Transactions on Information and System Security, 3 (1), pp.30-50, February 2000.

Is the Verification Problem for Cryptographic Protocols Solved?

Giampaolo Bella[1,2], Cristiano Longo[2], and Lawrence C. Paulson[1]

[1] Computer Laboratory, University of Cambridge,
15 JJ Thomson Avenue, Cambridge CB3 0FD (UK)
{gb221, lcp}@cl.cam.ac.uk
[2] Dipartimento di Matematica e Informatica, Università di Catania,
Viale A. Doria 6, I-95125 Catania, Italy
{giamp, longo}@dmi.unict.it

Abstract. Experience gained in the formal verification of cryptographic protocols is discussed. Many protocols have been analysed successfully thus far, but emerging goals may raise new challenges. However, existing formal approaches appear to be sufficiently scalable.

1 Overview

Needham and Schroeder's 1978 paper on authentication [15] introduced the first security protocols and called for methods of proving them correct. Research on protocol verification has progressed rapidly, especially since the BAN paper [10]. Today, complex real world security protocols can be formally analysed in detail. Kerberos [8], TLS [18], and even SET [7] have undergone formal verification. Our proofs, using Isabelle and the inductive method [17], require a considerable amount of human intervention. Other researchers have made great progress in making verification automatic. Cohen, for example, can reproduce nearly all of our examples. His tool, TAPS [11], takes a protocol description, guesses a suitable invariant and proves it using a resolution prover. Finally, TAPS proves the desired protocol goals from the invariant, again using resolution.

Innumerable other researchers have used automatic tools to verify or break a great variety of protocols. We are well beyond the stage of analysing authentication protocols. Electronic commerce, fair exchange, and non-repudiation protocols are routinely examined. An important finding is that most of these esoteric protocols do not require new types of tools. Non-repudiation, for example, can be expressed using existing specification languages [9]. It requires proving that certain messages guarantee thatcertain events have occurred, which is little different fromauthentication. Protocols that make use of smart cards make theirverification harder but still feasible [5]. Evenprobabilistic protocols have been analysed [16],though they do appear to require a new type of tool: a probabilistic modelchecker.

2 What's Next?

With this record of progress, what more remains to be done? The most obvious limitation of nearly all work is that encryption is treated as a black box. Real world encryption methods have their quirks and vulnerabilities. RSA encryption satisfies multiplicative laws that some protocols intentionally exploit. Exclusive-OR is self-cancelling, which is obviously untrue of encryption in general. Recent work justifies the use of black box encryption in certain cases [2,4]. Backes et al. [4] describe an abstract model, which looks little different from the traditional Dolev-Yao framework, but a simulation theorem relates the high level model to a low level one. The main problem with this work is the limited choice of primitives allowed, and even this limited choice has required the authors to undertake proofs of great complexity.

However, the hardest unsolved problems are not technical ones. For big protocols like TLS and SET, the labour required to understand the informal specification is the single biggest obstacle to undertaking a formal verification. One problem, of course, is the huge size of such protocols. Stripping out the unnecessary parts takes time and introduces the risk of over-simplification. This could hide serious flaws, or conversely could introduce spurious ones. (At least the latter case guarantees a publication.) Protocol design teams should give a higher priority to simplicity. They should also ensure that the documentation is complete. The SET development team devoted great efforts to producing readable documentation, but their so-called formal protocol description contains nothing but the message formats, with no explanation of what they are for. Perhaps worse, it presents no precise definition of SET's objectives or operating assumptions. Since the capitals handled by e-commerce protocols are increasing, we may even wonder whether cyberlaw should regulate the design of a protocol that aims at worldwide trust.

3 Real Flaws?

Many protocol flaws are not flaws in the protocol but, rather, disagreements about the operating environment. The popular attack on the asymmetric Needham-Schroeder protocol [13] succeeds because the attacker is an insider, but the designers apparently meant to protect the principals from outsiders only. Along the same lines, the TMN protocol was found significantly vulnerable [14] by an active attacker who shuffles the intercepted messages appropriately. It is no oversight that the protocol messages are not protected by external encryption, so each component is visible. This design would still do against a passive attacker, who can just overhear the traffic, as the protocol designer had presupposed.

Another example comes from the Zhou-Gollmann fair non-repudiation protocol [19]. Its principals are Alice and Bob, who are negotiating a contract, and a trusted third party (TTP). Fairness is the protocol's main objective. A successful run gives both Alice and Bob sufficient evidence to prove the other's participation. It is also acceptable that neither of them should obtain this evidence.

Gürgens and Rudolph recently demonstrated an attack on this protocol [12]. Alice reuses a session identifier and presents the evidence she obtained in the first run to confirm Bob's participation in the second run. Bob will not notice that he is being attacked unless he keeps track of session identifiers, which the protocol does not ask him to do. The TTP does not even take part in the second run. The attack only works because a real-world TTP cannot keep records of all past runs on-line. In the second run, Bob will seek evidence from the TTP, but as the session identifier relates to a very old run, the TTP will not find the corresponding record.

We analyzed this protocol [9] and did not find the attack because our formal model gave the TTP unlimited storage — as formal models typically do. Gürgens and Rudolph made their TTP discard session identifiers immediately after each run, ensuring that the attack would succeed. It is not clear whether their attack is realistic. If contract-signing protocols are ever to gain public acceptance, it is obvious that any deployed TTP must maintain a full audit trail, even if most of it is off-line. Whether this attack can succeed or not therefore depends on a detailed description of the dispute resolution mechanism. Dispute resolution should therefore be part of the protocol description. Recall again that the SET documentation does not even specify what would constitute a successful outcome, let alone state what should be done if it is unsuccessful.

Examples such as this demonstrate that it is unwise to claim that a protocol is verified unless the environmental assumptions are clearly specified. Even then, we can be sure that somebody will publish an attack against this protocol.

4 Another Success?

Abadi et al.'s certified email protocol. Abadi et al. [1] have recently designed a protocol for certified email delivery that appears to have numerous pros. The protocol TTP is stateless, and is *lightly* involved, as it never has access to the clear-text of the transmitted messages. The TTP works proportionally to just the number of handled messages, rather than to their sizes. No public-key infrastructure is necessary, and each principal should just have a shared secret with the TTP, such as a password. These features should favour the protocol implementation and deployment.

The gist of the protocol is that the email sender sends his email encrypted under a symmetric key to the intended receiver. (This is similar in spirit to the previously mentioned Zhou-Gollmann protocol: send the encrypted message first, and the corresponding key only later.) Here, the sender also attaches the symmetric key encrypted under the TTP's public encryption key. Hence, all the receiver can do is to forward the message to TTP so as to obtain the symmetric key. (In Zhou-Gollmann, it is up to the sender to lodge the key with the TTP.) As the TTP releases the key, it also releases a certificate documenting the transaction to the sender.

Figure 1 presents a protocol version simplified of headers and authentication options. The fullest authentication option, whereby the receiver should authenticate to both the sender and the TTP, is assumed. Our notation purposely makes no distinction between the various uses of symmetric encryption, asymmetric encryption, and digital signatures by using subscripts in all cases.

Abbreviations

$$\begin{array}{lcl} em & = & \{\!|m|\!\}_k \\ h_S & = & \mathsf{Hash}(q, r, em) \\ h_R & = & \mathsf{Hash}(q', r', em') \\ S2TTP & = & \{\!|S, k, R, h_S|\!\}_{\mathsf{TTPEncKey}} \end{array}$$

Steps

$$\begin{array}{llcll} 1. & S & \longrightarrow & R & : \mathsf{TTP}, em, q, S2TTP \\ 2. & R & \longrightarrow^{\mathsf{SSL}} & \mathsf{TTP} & : S2TTP', \mathsf{RPwd}, h_R \\ 3. & \mathsf{TTP} & \longrightarrow^{\mathsf{SSL}} & R & : k', h'_R \\ 4. & \mathsf{TTP} & \longrightarrow & S & : \{\!|S2TTP''|\!\}_{\mathsf{TTPSigKey}} \end{array}$$

Fig. 1. Abadi et al.'s certified email protocol

In step 1, the sender S sends the receiver R the encrypted email em, a query q, and a certificate for the TTP, $S2TTP$. The query is part of the authentication mechanism between R and S. They are required to agree out-of-band on some acknowledgement function to link a query q to its response r. The certificate is encrypted under the TTP's public encryption key, TTPEncKey, and contains the symmetric key k that protects em along with a hash linking em to the query/response pair.

In step 2, R issues a response r' to the query q' just received, which is not necessarily the same as S's query because the network is insecure. Then, R uses his query/response pair along with the received ciphertext em' to build his own hash h_R. Finally, R wraps this up with the received certificate and with his password RPwd and sends the outcome to TTP on an SSL channel.

In step 3, the TTP verifies that the received certificate has the expected form by decrypting it by its private decryption key. If this is affirmative, the TTP authenticates R by the password, and verifies that h'_S found inside the ticket matches h'_R. A positive check signifies that S and R agree on the authentication mechanism. If satisfied, the TTP replies to R on the SSL channel with the key found inside the ticket.

In step 4, the TTP sends the delivery receipt to S. It is the certificate signed by the TTP's private signature key. The TTP is trusted to take this step contextually with the previous one, so as to be fair with the peers. If the certificate received inside the delivery receipt matches S's stored certificate, then S authenticates R.

Notice that the TTP sees the symmetric key k, but not the clear-text message m as long as it does not overhear between S and R. (In that case, he could get the ciphertext em from message 1.) This is a desirable goal that reduces the trust in the TTP. It means that to trojan-horse it would not disclose clear-text emails. But, fundamentally, the protocol "should allow a sender, S, to send an email message to a receiver, R, so that R reads the message if and only if S receives the corresponding return receipt" [1, §2].

The designers conduct an informal analysis confirming that the protocol meets its goal. To do so, they make a number of (realistic) assumptions.

I. The TTP sends off the symmetric key and the delivery receipt at the same time. Hence, this applies to steps 3 and 4.
II. "The channels between TTP and the other parties are reliable" [1, §2] in the sense that the sent messages reach their destinations. Hence, this applies to messages 2, 3, and 4.
III. "R and TTP have a secure channel between them that provides secrecy and possibly authentication of TTP to R. (This is needed when R gives secrets to TTP in order to authenticate itself.) In practice such a channel might be an SSL connection ..." [1, §2]. Hence, this applies to messages 3 and 4.

Assumption I is obviously indispensable to achieve the protocol main goal, though it somewhat increases the necessary trust in the TTP's correct operation. In practice, The two protocol steps should be synchronised within a restricted time interval. Assumption II becomes acceptably realistic by sending a message repeatedly until an acknowledgment is received. Assumption III is acceptable too, on the basis of the popularity of SSL. The authentication requirement only serves to complement the use of the password, while the confidentiality requirement appears to serve to keep the symmetric key confidential.

Verifying the protocol inductively. From the verification standpoint, this protocol essentially poses threenew challenges, which we have taken using Isabelle and the inductivemethod. We can verify the protocol goals with only minor modifications tothe existing machinery. One challenge to the verification was the protocol reliance on the goalsof another security protocol to meet its own goals. To our knowledge, thishas never been investigated thus far. We should assume the goals of SSLavailable and find a formalization for them. The inductive method comeswith three network events: one for sending messages, one for receivingmessages, and one for noting down messages. There is a spy who sees allmessages ever sent or received by anyone, and those messages noted down bycompromised agents. The spy is the only agent who can deviate from the protocol.

Rather than formalising new *ad hoc* events for transmission overan SSL channel, we observed that the event whereby an agent A sends amessage X to an agent$B$ on an secure channel can be formalised as a Notes event:

$$\mathsf{B}\{\!|A, B, X|\!\}$$

Message X is made visible to agent B only, which models the confidentiality goal offered by SSL. Upon reception of a message on an SSL channel, we allow the receiver to check both events (by an appropriate inductive rule) so as to identify the sender uniquely. This models the authentication goal offered by SSL.

The second issue to solve prior to the verification was to adapt the threat model to the new setting. The existing spy is inspired to the conventional Dolev-Yao model. She can see every message travelling on the network, so no message can be assumed confidential, and she can interpose in any transmission, so no channel can be assumed authenticated. Clearly, all this should be prevented on the channels secured by SSL. How to achieve this aim was not obvious in principle, but turned out to be easy in practice.

We realised that our formalization with a Notes event keeps a sent message confidential every time sender and receiver areuncompromised. As for the authentication goal, it was sufficient toprevent the spy from sending arbitrary messages on an SSL channel. Inother words, SSL forces the spy to act legally by sending the rightprotocol messages and providing her true identity. In practice, we werepleased to observe that the existing rule formalising the spy's illegalcapabilities could be left unaltered.

The final obvious challenge was how to deal with the new goals. Properties of certified email delivery had never been formally investigated before. We found that they can be formulated as *traceability* properties, which we have already studied [9], namely they reduce to proving that an event precedes another. Most importantly, also the corresponding proof strategies were found to scale up. We have succeeded in proving all the main goals of this protocol [6].

5 Conclusions

The formal verification of cryptographic protocols has reached a level at which it can be considered mature. It has proposed and motivated modifications and amendments to a number of protocols. Large and deployed protocols can be analysed successfully, as well as recent protocols aiming at currently emerging goals. But new goals will be raised by the forthcoming distributed applications, possibly raising more and more challenges to the verification.

Acknowledgements

Research was funded by the EPSRC grant GR/R01156/01 *Verifying Electronic Commerce Protocols.*

References

1. M. Abadi, N. Glew, B. Horne, and B. Pinkas. Certified email with a light on-line trusted third party: Design and implementation. In *Proceedings of the 11th International Conference on Wold Wide Web (WWW-02).* ACM Press and Addison Wesley, 2002.

2. M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15:103–127, 2002.
3. A. Abdallah, P. Ryan, and S. Schneider, editors. *Formal Aspects of Security*, 2002. Technical Report CSD-TR-02-13.
4. M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. preprint, IBM Zürich Research Laboratory, 2003.
5. G. Bella. Inductive verification of smart card protocols. *J. of Comp. Sec.*, 11(1):87–132, 2003.
6. G. Bella, C. Longo, and L. C. Paulson. Verifying second-level security protocols. In D. Basin and B. Wolff, editors, *Theorem Proving in Higher Order Logics: TPHOLs 2003*, LNCS 2758, pages 352–366. Springer, 2003.
7. G. Bella, F. Massacci, and L. C. Paulson. The verification of an industrial payment protocol: The SET purchase phase. In V. Atluri, editor, *9th ACM Conference on Computer and Communications Security*, pages 12–20. ACM Press, 2002.
8. G. Bella and L. C. Paulson. Kerberos version IV: Inductive analysis of the secrecy goals. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Computer Security — ESORICS 98*, volume 1485 of *Lecture Notes in Comp. Sci.*, pages 361–375. Springer, 1998.
9. G. Bella and L. C. Paulson. Mechanical proofs about a non-repudiation protocol. In R. J. Boulton and P. B. Jackson, editors, *Theorem Proving in Higher Order Logics: TPHOLs 2001*, volume 2152 of *Lecture Notes in Comp. Sci.*, pages 91–104. Springer, 2001.
10. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426:233–271, 1989.
11. E. Cohen. TAPS: A first-order verifier for cryptographic protocols. In *Proc. of the 13th IEEE Comp. Sec. Found. Workshop*, pages 144–158. IEEE Comp. Society Press, 2000.
12. S. Gürgens and C. Rudolph. Security analysis of (un-) fair non-repudiation protocols. In Abdallah et al. [3]. Technical Report CSD-TR-02-13.
13. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems: second international workshop, TACAS '96*, volume 1055 of *Lecture Notes in Comp. Sci.*, pages 147–166. Springer, 1996.
14. G. Lowe and B. Roscoe. Using CSP to Detect Errors in the TMN Protocol. *IEEE Trans. on Software Engineering*, 3(10), 1997.
15. R. M. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Comm. of the ACM*, 21(12):993–999, 1978.
16. G. Norman and V. Shmatikov. Analysis of probabilistic contract signing. In Abdallah et al. [3]. Technical Report CSD-TR-02-13.
17. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *J. of Comp. Sec.*, 6:85–128, 1998.
18. L. C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Trans. on Inform. and Sys. Sec.*, 2(3):332–351, 1999.
19. J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Symposium on Security and Privacy*. IEEE Computer Society, 1996.

Secure Sessions from Weak Secrets

Bruce Christianson[1], Michael Roe[2], and David Wheeler[3]

[1] Computer Science Department, University of Hertfordshire, Hatfield
[2] Microsoft Research Limited, Cambridge
[3] Computer Laboratory, University of Cambridge,
England, Europe

Abstract. Sometimes two parties who already share a weak secret k such as a password wish to share also a strong secret s such as a session key without revealing information about k to an active attacker. We assume that both parties can generate strong random numbers and forget secrets, and present new protocols for secure strong secret sharing, based on RSA, Diffie-Hellman, and El-Gamal. As well as being simpler and quicker than their predecessors, our protocols also have stronger security properties. In particular, our protocols make no cryptographic use of s and so do not impose subtle restrictions upon the use which is subsequently made of s by other protocols. Neither do we rely upon the existence of hash functions with serendipitous properties. In the course of presenting these protocols, we also consider how to frustrate some new types of cryptographic and system attack.

1 Introduction

Sometimes there is a requirement to establish a secure session between two parties who initially share only a weak long-term secret. "Secure" includes the requirement that the parties can be sure that they are talking to each other, as well as properties of integrity and secrecy. By "weak secret" we mean a secret that is chosen from a moderately small set, so that an attacker could search through all possible values. Passwords are often weak secrets, as the total number of words in a dictionary is searchable.

A weak secret cannot be used directly as a cryptographic key to secure the session, as this is vulnerable to a known plaintext attack. If the attacker knows (or can guess with high probability of being right) the message plaintext m corresponding to a known encrypted text $E_k(m)$ then they can search through all possible values of the password until they find the value k which decrypts the cyphertext to m. This reveals the password, which can then be used to decipher the session.

Suppose that the parties who wish to communicate have good random number generators. This means that they can generate secrets which are strong (chosen from a set which is too large to search) but not shared. We would like to have a protocol which starts with a weak shared secret and a pair of strong non-shared secrets and which ends up with a secret which is both strong and shared. We refer to such a protocol as a Strong Secret Sharing Password (S3P) Protocol.

Previous attempts at solving this problem include Bellovin and Merritt's Encrypted Key Exchange [3], Jablon's SPEKE [9], and Lucks' Open Key Exchange [12]. A related, but slightly different approach is taken by Gong *et al* in [7].

In the next section, we give a careful specification of the properties which we desire S3P protocols to have, and of the assumptions under which we believe our protocols to have these properties. In Section 3, we present an S3P protocol based on RSA and show that it resists a number of attacks, some known and some novel. In section 4 we consider various DH-based S3P protocols which have stronger security properties than the conventional versions, and discuss some novel threats. In section 5 we consider the implications of using EG in place of DH when the requirement is to transfer an existing secret rather than to agree a new one. In section 6, we discuss what should be done when an active attack is detected, and introduce the notion of a robust protocol wrapper. In the final section we summarize our conclusions.

2 S3P Protocol Properties and Assumptions

In this section, we give a careful specification of the properties which we desire S3P protocols to have, and of the assumptions under which we believe our protocols to have these properties.

The S3P protocols described in this paper are peer-to-peer protocols, which operate directly between two principals rather than between a principal and a server. In accord with tradition, we assume that the two parties trying to operate the S3P protocol are unambiguously known to each other as A and B.

2.1 Properties

We turn now to describing the features which we desire the S3P protocol to have. At the start of the protocol A and B share a weak secret k. Following a run of the protocol which A believes to have ended correctly, it should be the case that B really did participate in that run of the protocol, and that the two of them really do now share a fresh strong secret s. The corresponding statement should also be true for a run which B believes to have ended correctly. The protocol should not reveal information about the weak secret k in any case.

The protocol should be secure against active attacks in which the attacker creates or modifies messages. Leakage or cryptographic compromise of a strong secret s shared using a protocol run should not reveal information about the password k. If several strong secrets s_i are shared by different runs using the same password k then obtaining one such s_i should not help an attacker to obtain s_j with $j \neq i$.

An attacker should not be able to obtain any information about whether a guessed value of the password is correct without making an active attack: effectively the attacker should be forced to masquerade as one of the participants to the other. An active attack should be detectable by at least one of the genuine participants, unless the guessed value is correct, and each such failed attack should eliminate no more than one possible value of the password (ie the unsuccessful guess) from the attacker's list of possible password values. Approaches such as [1] do not satisfy this requirement. Finally, if the password is compromised (by whatever means) this should not assist the attacker to obtain strong secrets agreed using the protocol with that password prior to the point of compromise, or to obtain subsequently agreed strong secrets by passive attack.

2.2 Assumptions

We assume that neither party can reliably maintain the integrity of a strong secret s from one protocol run to another: in other words if A tries to use a strong secret from one run in another run, then there is a good chance that s either leaks, or is forgotten, or changes (or is changed) without A noticing that it has. This assumption may correspond to the fact that the parties move frequently from one piece of hardware to another, or may be because the hardware is initialized in some way between protocol runs to erase secret information. We discuss this issue further in section 6.

We assume that both parties can reliably maintain the integrity of public, slowly varying data such as software and public keys: other protocols are available to assist with this [11].

The protocols which we consider include the operations "generate a random bit pattern n" and "forget the bit pattern m". We assume that both ends have good irreproducible random bit generators and can forget secrets. By the first assumption we mean that our threat model does not consider the possibility of an attacker determining n by examining other bit patterns produced (previously or subsequently) by the same or other generators. By the second assumption we mean that our threat model does not consider the possibility of an attacker subsequently determining m from from an examination of the hardware which has been instructed to forget it. Note that the hardware which must forget includes the random generator. This assumption is probably the most difficult requirement to realize in practice.

We wish to make no assumptions about what the strong shared secret will be used for. The S3P protocol run used to agree the strong shared secret s ends as soon as both parties can be sure that s has been appropriately shared. In the light of known chosen protocol attacks [10] we wish to impose no restrictions upon the nature of the cryptographic protocols or algorithms to which s is handed off for subsequent use, or upon the length of s itself. It may be intended to reveal s (for example s may be used as a one-time pad) or it may be that s is not intended to be used as a key at all, but as a salt or initial value. In particular, the S3P protocol should not assume that s is strong: it may be feasible for an attacker to search for s in the time available between steps of the protocol. This may be because s is required to be weak (less than 40 bits, suppose) or because the protocol is running very slowly (the messages may be carried by a diskette sent through the post, for example.)

2.3 System Context

In the protocol descriptions that follow, we omit from each message the header information identifying which protocol is being used, which parties it purports to operate between, which run of the protocol the message relates to, and the sequence number of the message within that protocol run. These data will be conveyed by the outer wrapping protocol, which will be discussed in section 6. Also, we have not yet specified explicitly what conditions cause a participant to treat a particular run as having failed (eg receiving an incorrect bit pattern or timing out.) We shall also consider these points further in section 6, but first we describe the inner S3P protocols themselves.

3 RSA Based Protocol

In this section, we present new S3P protocols based on RSA. Correct protocols of this form have previously resisted construction, to the point where some have doubted their possibility. We show that this RSA-S3P protocol resists the known attacks as well as some novel ones.

3.1 Basic RSA-S3P Protocol Description

A generates an RSA modulus $N = pq$ with p, q prime and so that $(p-1)/2, (q-1)/2$ each contains a large prime factor. We assume that the bit-lengths of p, q and N are prescribed exactly.

We assume that there is a publicly known function e which converts a password k into a large prime number $e(k)$ suitable for use as an RSA exponent. By large we mean that the bit length of $e(k)$ is strictly larger than that of N by some suitable algorithm. For example, supposing that the password k is encoded in such a way that the bit length of k is small relative to that of $\sqrt{N}$, then one suitable algorithm would be to search through ascending values of i until a prime of the form $1 + k(a + ib)$ is found, where a, b are published co-prime constants guaranteed to exceed N.

The RSA-S3P protocol runs as follows:

$$A \rightarrow B : N \tag{1}$$
$$B \rightarrow A : z^{e(k)} + 2^{e(k)} \bmod N \tag{2}$$
$$A \rightarrow B : n_a \tag{3}$$
$$B \rightarrow A : n_b \tag{4}$$

Here $z = c|s|n_a|n_b$ where s is the session key, c is a strong random number called a *confounder*, and n_a, n_b are random numbers called *nonces*. The vertical bar $|$ denotes concatenation of bit strings, with the high order bits on the left.

Note that a fresh key N is required for each run of the protocol, but the function $e(k)$ can remain constant. Only A need verify the strength of the public key N, although B must check that N has the correct bit-length. A must forget

$d(k)$, the decryption key corresponding to $e(k)$, as well as p and q. Both A and B must forget c. B must forget the whole of z if the protocol fails at step (3).

In the RSA protocol, the key s and all nonces are chosen by B. Although s need not be strong, it must contain no redundancy and must not be predictable. Prior knowledge of the value of s which B will choose allows an attacker masquerading as A to determine k. Also note that s must appear random and so can't be a public key. Although the confounder c must be strong, the nonces n_a and n_b need not be strong, although they should be significantly harder to predict than k.

Jolyon Clulow points out that A must reject an offered value of 0 in message (2), or an attacker can force $z = N - 2$ without knowing or learning k.

3.2 Design Discussion

The presence of c prevents an attacker using a compromised session key and a copy of message (2) to search for k.

It is vital that there be no redundancy in the plaintext $z = c|s|n_a|n_b$ which is encrypted in message (2). If there were, then an attacker masquerading as A could use this to search for k in time to generate message (3) correctly and complete the protocol run. Note that consequently z must be a random number in the range $1 \ldots N$. The statistical distribution of the high order bits of c is thus skewed, because N is not a power of 2. However, this effect dies away exponentially with bit order, so the low order bits of c plus all bits of s cannot contain enough skew to be useful to an enemy after any achievable number of protocol runs. It must be infeasible for an attacker to search over the low-order bits of c. Otherwise after s is revealed for a completed run, passive search would reveal k by a match on message (2).

The purpose of messages (3) and (4) is to convince A and B that they are not experiencing an active attack. Instead of using the nonces n_a and n_b in messages (3) and (4), we could use cryptographic hashes of them instead. But nothing is gained by doing this, and it requires us to exhibit a cryptographic algorithm with suitable subtle properties, a commitment which we prefer to avoid. (3) and (4) by an attacker masquerading as B, who finds s in time to send message (4). Effectively, if s is used in this way and is not sufficiently strong, then the attacker gets an undetected guess at k.

It is tempting to try and shorten the protocol run to three messages by combining the texts of messages (2) and (4) into a single message. This doesn't work, because there must be no redundancy in message (2).

3.3 Number-Theoretic Attacks

Factorization of N by an attacker gives the attacker k and, worse, allows the attacker to obtain old values of s. The public key N must therefore be many times longer than s, consequently a large number of bits is available for n_a, n_b and c. However the protocol must specify the exact bit-length of N, and B must check that N has the correct number of bits, in order to ensure that an attacker does not insert an extra factor into N in order to gain residue information about $e(k)$.

The term $2^{e(k)}$ in the second message is required to block the *Bleichenbacher attack*: if this term is omitted, then an attacker masquerading as B can send $z^e \bmod N$ in message (2), where $e = \prod_i e(k_i)$ and the k_i are candidate passwords. Comparison of n_a with $z^{e/e(k_i)}$ for each i now reveals the correct k_i, and this can even be done in time for the false B to generate message (4) correctly and finish the protocol run. Variations are possible in message (2), for example using $3^{e(k)}$ as the added term or using the exclusive-or $\oplus$ in place of modulo$-N$ addition.

We turn now to a consideration of the constraints on the function $e(k)$. Suppose that e is a fixed prime, and consider a modified form of the RSA-S3P protocol where message (2) contains $z^e + k \bmod N$ in place of $z^{e(k)}$. If an attacker masquerading as A can choose $N = pq$ where p is a prime of the form $r.e + 1$, then the Euler totient $\phi(N) = re(q-1)$ so for almost all values of z we have $(z^e)^{r(q-1)} = 1 \pmod N$. Exhaustive search following a single foiled active attack now reveals k.

This attack on the modified protocol can be blocked by making the requirement that $e > N$, which ensures that e is relatively prime to $\phi(N)$.

Suppose p_i are small odd primes and the attacker would like to find i with $p_i = e(k)$. Define $p = 1 + 2\prod_{i \text{ even}} p_i, q = 1 + 2\prod_{i \text{ odd}} p_i$ and arrange the indexing of p_i so that p and q are prime. Setting $N = pq$ we have (almost certainly) that $p_i | e(k)$ iff $(z^{e(k)})^{P/p_i} = 1 \pmod N$ where $P = 4\prod_{\text{all } i} p_i$. Other number-theoretic attacks are considered by Patel [13].

To block attacks of this form it suffices to ensure that $e(k) > N$ for all k, since the bit-length of N is prescribed by the protocol and checked by B. Alternatively, we could just insist that all $e(k)$ have a one somewhere in the high order bits. This still allows an attacker to eliminate two values of k per active attack, but no more.

We also need to ensure that the mapping from k to $e(k)$ is, as nearly as possible, one-to-one. The fact that a and b are relatively prime and $ab > N$ ensures this. The density of primes below N implies that a prime will be found on average for i of order $\ln N$ and almost always for i very much less than $\sqrt{N}$. There are other subtle constraints upon the algorithm for e. For example, if the function $e(k)$ were instead defined to be the first prime after $a + b.k$, then the density of prime numbers implies that $e(k) = a + b.k + i$ for small i. The fact that i is typically bounded by a small multiple of $\ln N$ gives the attacker information about $e(k) \bmod b$. An attacker pretending to be A can choose N to contain a factor of the form $br+1$, and hence deduce information about $z \bmod br+1$ which can be used to recover z from message (2) with a greater probability of success than guessing k.

4 Diffie-Hellman Based Protocol

In this section 4 we consider various DH-based S3P protocols which have stronger security properties than the conventional versions, and which make less use of superencypherment. We also discuss some novel threats.

4.1 Basic DH-S3P Protocol Description

Let q be a publicly known large prime of prescribed length, and let g be a publicly known residue modulo q. To prevent various known subtle attacks [3,9,13] we assume that $p = (q-1)/2$ is a prime and g is a generator modulo q, so that g^n has period $2p$. Note that in case $p \bmod 4 = 1$ we can take $g = 2$ [8, Theorem 95]. We assume that k has a smaller bit-length than q and is encoded in such a way that any two candidate values for k differ in at least three bit positions. The last requirement is for technical reasons which will be discussed later, but can be guaranteed by adding a small amount of redundancy to k.

A and B select strong random numbers x, y respectively. By strong we mean that exhaustive search is infeasible. The basic version of the DH-S3P protocol runs as follows:

$$A \rightarrow B : g^x + k \bmod q \tag{1}$$
$$B \rightarrow A : g^y \bmod q \mid n_b \tag{2}$$
$$A \rightarrow B : n_a \tag{3}$$

where we write $g^{2xy} \bmod q = z = c|s|n_a|n_b$ with semantics as in the RSA protocol.

Both A and B should check to ensure that $z \neq 0$ and $z \neq 1$. If $z = 0$ or $z = 1$ then the run fails, since otherwise an active attacker could masquerade as B by using these values in message (2).

The nonces n_a, n_b and the confounder c need not be strong, although they should be significantly harder to predict than k. We require that x, y strong and large relative to $\log_2 q$.

A must forget x and g^x, while B must forget y. Both A and B must forget c. A must forget the whole of z if the protocol run fails at message (2). The DH protocol ensures that s appears random, but does not allow it to be chosen or predicted by A or B.

4.2 Design Discussion

Whereas the RSA-based protocol required four messages, the Diffie-Hellman variant can be done in three, effectively by combining both texts uttered by B in the same message. Consequently, in marked contrast to the RSA case, the second message in the DH protocol contains verifiable redundancy in the form of n_b. The reason an attacker cannot use this to break the protocol is that the redundancy is only detectable by an entity who knows x or y. These are not searchable by hypothesis, and the value of x is not deducible from message (1) even with a guessed value for k.

As with RSA, eventual cracking of the chosen public key will give the attacker k and, worse, allow the attacker to obtain old values of s. For this reason the public parameter q must be many times longer than s, and so a large number of bits is again available for n_a, n_b and c.

The purpose of the Hamming-distance restriction alluded to earlier upon the encoding of k is to prevent the attacker testing multiple values of k in a single run. If $g = 2$, then the attacker knows the discrete logarithms of small powers of 2, and can use this fact to test simultaneously a set of candidates for k, each of which differs in only one bit from some value k_0. The attacker sends k_0 as the first message and inspects g^{2ix} for all i less than the bit-length of k, to see if one contains n_b. If all candidate values for k are at least three bits apart then this attack is defeated. For example including a Hamming code would add only 10 bits to a 1,000 bit k.

4.3 Choosing the Modulus

In this sub-section, we consider possible alternative approaches to the choice of q and g. This in turn leads to some variations on the DH-based S3P protocols.

To avoid narrowing attacks, we require that q be a prime of the form $2p + 1$ for some prime p, and that g be a primitive root modulo q. Such parameters are relatively expensive to generate, and in the DH protocol both A and B must check that q, g are suitable values, since using poor values can reveal k. In the RSA case only A need check. However, while the RSA protocol needs a new key N for each run, the DH protocol can use same parameters g and q many times. Consequently the parameters g and q could be relatively long-term and chosen by A and B jointly prior to the first run of the protocol, or else chosen, certified and published by some party, protocol or algorithm trusted for this purpose by both A and B.

Alternatively, A could choose the public parameters q, g and send them to B in the first message. B must carry out a deterministic test to verify that the parameters have the required properties. A deterministic test should be used, since many non-deterministic tests assume random rather than malicious choice of candidate primes. To enable B to carry out such a test efficiently A can send a witness along with the parameters. However, we still need q to be many times longer than s. If A chooses q, g each run then it may be more efficient to find a prime q of the more general form $q = rp + 1$ where p is a large prime and r is relatively small, since these primes can be sieved for more quickly, although it is then more difficult to find a generator g. The previous case corresponds to $r = 2$, whereas for this case $r = 2^n$ for a small n might be better. To avoid narrowing attacks when q is of this more general form, take $g^{rxy} \bmod q = z = c|s|n_a|n_b$ to force z into the large subgroup, and check $z \neq 1$ and $z \neq 0$. Another option is to replace $g^y \bmod q$ by $g^y + k \bmod q$ in message (2).

A further possibility is where a fresh q of the prescribed length is chosen in some way for each particular run of the protocol. For example, the value of q may depend in a deterministic way upon both k and an unpredictable random value r, so that $q = q(k, r)$, where r is produced during or just prior to the protocol run. This unpredictable value of r need not be kept secret, and may published by a beacon, or agreed by A and B using some other protocol. In this case the protocol requires no superencypherment by k at all. The first message contains just $g^x \bmod q$ with the provision that trial values for x must be picked and discarded until one is found for which the high order bit of $g^x \bmod q$ is zero, and similarly for y. This ensures that the protocol run gives no information about which q was used, and hence leaks no information about k. By forcing all $q(k, r)$ to have high-order bits $100\ldots0$ for some fixed number of zeros, we can make the probability of a high order one in $g^x \bmod q$ as small as desired.

4.4 Modified DH-S3P Protocol

We conclude this section by considering in more detail the case where the random number r is produced during the S3P protocol run itself. The simplest method is for A to send r in the first message along with $g^x \bmod q$. However the protocol which follows is designed to illustrate a more paranoid scenario. We assume that A and B wish to use part of the value of s to settle a bet [15]. Even if they have no doubt of one another's honesty, they must be able to prove to a sceptical third party that neither of them has the capability to influence the value of the shared secret s in a predictable way. The random values such as x, y actually generated during the course of the protocol run must be destroyed, and so cannot subsequently form part of an audit trail.

A picks strong random numbers m, x and y'. B picks strong random numbers m', x' and y. In the protocol description which follows, $q = q(0|m, k), q' = q(1|m', k)$ and g, g' are the corresponding generators. We assume that $q = r.p + 1, q' = r'.p' + 1$ for large primes p, p'. The functions unzip0 and unzip1 denote the even and odd-numbered bits respectively.

The DHm-S3P protocol runs as follows:

$$A \rightarrow B : m \mid g^x \bmod q \tag{1}$$

$$B \rightarrow A : m' \mid \text{unzip0}\ ((g')^{x'} \bmod q' \mid g^y \bmod q) \tag{2}$$

$$A \rightarrow B : (g')^{y'} \bmod q' \tag{3}$$

$$B \rightarrow A : \text{unzip1}\ ((g')^{x'} \bmod q' \mid g^y \bmod q) \mid n_b \tag{4}$$

$$A \rightarrow B : n_a \tag{5}$$

Here $z = c|s|n_a|n_b = (g^{rxy} \bmod q) \oplus (g'^{r'x'y'} \bmod q')$ where $\oplus$ denotes bitwise XOR. Effectively the DHm-S3P protocol runs two instances of the basic protocol back to back, but reveals information only about the exclusive-or of the two results. This means that an attacker must crack discrete log for both q and q' simultaneously, rather than searching log tables one at a time. The unzip functions force A to commit y' before learning g^{rxy}, but after B commits to y.

5 El Gamal Based Protocol

In this section we consider the implications of using EG in place of DH, when the requirement is to transfer an existing secret rather than to agree a new and unpredictable secret. The EG variation of the S3P protocol allows B to pick the session key and nonces, as was the case in the RSA protocol. The EG-S3P protocol runs as follows:

$$A \rightarrow B : g^x + k \bmod q \tag{1}$$
$$B \rightarrow A : g^y \bmod q \mid z \cdot g^{2xy} \bmod q \mid n_b \tag{2}$$
$$A \rightarrow B : n_a \tag{3}$$

where $z = h|s|n_a|n_b$ as for the DH protocol, except that instead of the confounder c, z contains a known fixed bit pattern h chosen so that it is not invariant under shifts or subtraction from q. The constraints on h will be further discussed below. The password k is encoded as in the DH protocol. As in the Diffie-Hellman case, A and B should check that $g^{2xy} \notin \{0, 1\}$. A should also check that z contains the expected value for h.

A must forget x and g^x, while B must forget y. A must forget the calculated value of g^{2xy} and z if the protocol run fails at message (2). Apart from this, h need not be kept secret. The EG protocol allows s as well as n_a, n_b to be chosen by B so as to contain redundancy or known text.

The El Gamal variant shares some features with the RSA case and some with the DH case. As with DH, it is a three-message protocol and the middle message must contain redundancy. In the RSA protocol the value of s is chosen by B but s must contain no redundancy discernible to the attacker: otherwise k is in danger. The DH protocol ensures that s appears random, but does not allow it to be chosen or predicted by the participants. The EG protocol allows B to choose s, and for s to contain redundancy in any form desired. Indeed for the EG protocol even prior knowledge of s by the attacker does not assist in an active attack against k. Also, in the EG protocol n_a and n_b may contain redundancy or known text. As with the other protocols, the nonces n_a and n_b need not be strong, although they should be significantly harder to predict than k. However in the EG protocol h is not a confounder at all. Instead, it contains redundancy to prevent a person in the middle modifying messages (2) and (3) in such a way that the protocol appears to complete successfully, but with A and B disagreeing on s. For example an attacker can multiply z by two in the second message and shift n_b and n_a one bit to the left and right respectively, with a 50% chance of escaping detection. The result is that s is shifted left one bit in transmission. Division and complementation are also possible. Such attacks can be prevented by placing a fixed bit pattern in h, for example a single 1-bit with n 0-bits on either side will suffice, provided 2^{-n} is small relative to the chance of guessing k.

The EG protocol, like the DH protocol, can use the same parameters g and q many times. As in the DH protocol, both A and B must check that q, g are suitable values, since using poor values will reveal k. As with RSA and DH, eventual cracking of the chosen public key will give the attacker k and, worse, allow the attacker to obtain old values of s. For this reason the public key must be many times longer than s, and so a large number of bits is available for n_a, n_b, h.

An alternative approach (which we do not pursue here) is to obtain n_a, n_b from g^{2xy} as in the DH protocol, rather than from z.

6 System-Level Considerations

In this section, we discuss what should be done when an active attack is detected, and introduce the notion of a robust protocol wrapper. We also discuss the system context for the deployment of S3P protocols, and the hardware support required.

6.1 Action Following a Detected Attack

An important feature of all the S3P protocols we consider is that it is not acceptable to ignore an active attack. If active attacks are ignored, the attacker can make one active attack for each possible k, and will eventually succeed. If suitable emergency action is taken in the event of an active attack being detected (eg switching to a more expensive but physically secure channel, or to another, previously agreed, password, after a certain number of failed runs), then the attacker never gets enough information to improve his chances of guessing correctly by more than some previously agreed security parameter.

In an extreme case we can confine the attacker to two guesses, one with each of A and B. In a less extreme case, with (say) a million equally likely values for k, we could choose to allow the attacker 32 guesses with each of A and B. The attacker has less than a one in ten thousand chance of obtaining the true value of k. Of course, this strategy requires some assumptions about the physical locations of A and B, and their ability to remember the number of active attacks detected over an appropriate time scale such as the expected life of the long term password k. We also need to specify, in any particular system context, how these numbers are stored and whether they are secret.

In particular, if the protocol is used by many pairs of participants, then an attacker can make a small number of attacks against each of a very large number of passwords, and will almost certainly succeed against one. The effects of this form of penetration, and the countermeasures for containing it, depend upon the interactions between the system-level protocols for which the strong shared secrets are used.

An attractive alternative to using a deterministic counter and a threshold is to invoke emergency action with a certain constant probability after each detected attack. For example, if we set this probability at 2% then the alarm will almost certainly be raised after 70 detected attacks, regardless of who detects them. Since we assume that all parties who use the S3P protocols are able to generate good random numbers, this stochastic technique imposes no new system constraints.

6.2 Robust Protocol Wrappers

The primary system context which we consider for the S3P protocol is one of paranoia rather than hostility. In other words, we assume that the world is full of very clever and hardworking attackers, but at the same time we are confident that things will go right most of the time. In effect, we assume that the S3P protocol is nested inside another protocol, which we call the wrapper, and that the outer wrapping protocol works nearly all the time unless there really is an active attack by an extraordinarily malicious and ingenious entity. The inner S3P protocol is intended both as a trip-wire to indicate whether the outer wrapper has been deliberately breached, and as a last-ditch defence.

A primary purpose of the outer protocol wrapper is to ensure that the inner S3P protocol is under no accidental misapprehension about whether an offered bit pattern represents an attempt to engage in the S3P protocol, and if so in which run, at what stage, and as whom. A similar two-layer scheme for distinguishing accident from malice was used by Lomas and Christianson [11] and a related notion of robustness is discussed by Anderson and Needham [2]. The S3P run must fail if any such presented bit pattern is incorrect, otherwise the enemy gets a free guess.

We also assume that “eventually” a run of the protocol which does not proceed will be regarded as having failed by at least one of the participants. However this timeout may be very long. One reason for this is that we do not wish to have too many false alarms, but there is another reason. We wish also to allow a system context in which a run of the S3P protocol is transported by a slow non-cryptographic outer protocol such as fax, snail-mail, or sneakernet. This gives rise to two further considerations: re-entrancy and interactive breaking.

If an S3P protocol run can take a long elapsed time, then the S3P protocol must be re-entrant. The total number of active runs (plus the number of previously detected failures) must be less than the threshold value for the number of active attacks which we are prepared to tolerate. This ensures that all runs which terminate successfully are safe. In particular, runs can be pipelined or used back-to-back between the same two parties on tamper-proof hardware such as smart cards which are kept locked up when not in use.

The possibility of a long elapsed time also provides one motivation for our consideration of the possibility that a value of s could be broken between steps of the S3P run, for example if it were used to encrypt a known plain text as part of the S3P protocol. In any protocol involving key agreement, it is possible to specify the agreement of a much longer shared key than required, and to forget all but the required number of bits. Many applications of other published protocols would benefit from doing this.

Just as we do not assume s to be strong, neither do we *require* k to be weak: although the S3P protocols were originally designed to work with passwords k drawn from a space of order 2^{20} possibilities, the protocols also have particularly nice properties when a 40-bit shared key k is being traded up to a series of 120-bit keys s_i.

6.3 Tamper-Proof Hardware Platform

One possible system context for S3P is where we wish to ensure that the right person is using a particular box. The box may be designed to be used by several different people (eg a workstation in a shared area) or by only one person (eg a mobile telephone or a hand-held authenticator). The box may be stateful (eg able to retain session keys) or stateless (all mutable information is deliberately erased between uses). However a box may be stolen or tampered with. We wish to ensure that the box can only be used by a person who knows the correct password.

To deploy the S3P protocol we assume that the box is tamper-evident, and unviable to forge. We assume that the user checks the tamper-evident seal before entering the password at the start of each run. We assume that the box forgets the password once the S3P protocol run ends or fails, and that while the run is in progress the box is tamper-proof, in the weak sense that that the box will irrevocably destroy (forget) secrets rather than allow them to be read. This property might require that the box is used in a different environment from the one in which it is stored between runs.

Under these assumptions, the S3P protocol suffices to ensure that the box cannot be used by a person who does not know the password. In the case where the box may be used by more than one person, each user may have a different password. Note that tamper-proofing is not required except while the protocol is running. Tamper-evidence suffices the rest of the time even in the stateful case. State which persists between runs can therefore be used to support the outer wrapping protocol, so long as the state of the box between runs can reveal no information about the password. However the inner protocol must not rely upon the outer protocol preserving state correctly. This point will be illustrated at the end of the section by discussion of a reflection attack.

6.4 Blocking Reflection Attacks

We conclude this section with a brief consideration of how to block a reflection attack. The inner S3P protocol is assumed to be stateless, and so a new run cannot reliably determine which other protocol runs are still active when it begins. Suppose that A attempts to run the protocol with B. The attacker takes the first message from A and replays it to A as if it came from B initiating a different run of the protocol. A's reply to this is in turn reflected as a reply to A's initial message, and so on for the subsequent messages. If A is not careful, she will end up sharing a fresh strong secret with herself, rather than with B, in violation of our requirement that the other intended participant must actually be involved in any apparently successful run.

Of course, this attack cannot actually succeed against the protocols as we have described them here, since A always sends the odd-numbered messages and B the even. But suppose we wish to allow either party to initiate the protocol, possibly re-entrantly, so that A may legitimately use k to speak the lines attributed to B in the script.

We can block the reflection attack by associating the nonces firmly with principals. In respect of each password, one party is (by mutual agreement) the $a-$end and the other is the $b-$end. Suppose that *Carol* is the $a-$end and that *Ted* is the $b-$end. Then whenever *Carol* has to place a nonce in a message she always uses n_a, regardless of whether she is playing the part of A or of B.

7 Discussion

The protocols given in this paper are provocatively weak. For example, we use bit selection in place of a hash function, modulo addition to perform superencypherment, and a base of 2 for certain exponentiations.

This weakness is quite deliberate. From a practical point of view, our protocols could doubtless be strengthened by the judicious inclusion of "one-way" hash functions, or the use of more complex forms of superencypherment and convolution. We have instead put forward very concrete versions of the protocols, with primitives which rely upon specific number-theoretic relations between, for example, modular exponentiation, and addition or "unzip". Following Ockham, we wish to understand these simple protocols before we propose anything more complicated.

We have not provided correctness proofs for these protocols here. This is an area in which we anticipate future progress. The major present difficulty lies in determining precisely how the threat model interacts the desired properties of the protocol with those of the underlying cryptoalgorithm. For example, one standard *reductio* approach might be to prove that, if the RSA-protocol reveals more information than it should about k, then it also gives an attacker the ability to decrypt unpredictable RSA cyphertexts. However attacks in the spirit of Bleichenbacher show that such an outcome need not constitute a break of the RSA cryptoalgorithm, and hence ought not simply to be presumed counterfactual. Conversely, a protocol continues to satisfy the assertion of a correctly proved predicate, even after the protocol has been broken by another means.

On the positive side, cryptographic innovations which we claim for this paper include the successful use of RSA as a vehicle for encrypted key exchange, section 4.4. As a minor point, we also draw attention to the lack of superencypherment by k in the second message of the basic version of our DH protocol. However, we regard as primary our original contribution to consideration of the system context given in section 6, including the introduction of robust protocol wrappers and their application to the case of "stateless" platforms.

The protocols in this paper owe an obvious debt to the original discussion by Bellovin and Merritt [3], which opened up a number of fertile research directions. We make no attempt to give a systematic account of all this related work here. (An excellent roadmap is provided by Jablon's website at `http://www.IntegritySciences.com`.)

Some of the material in this paper appeared in preliminary form in University of Cambridge Computer Laboratory Technical Report 458 (1998). We would like to thank Daniel Bleichenbacher, David Jablon, David Wagner and everyone else who provided attacks and related comments on these early versions of the protocols.

8 Conclusions

Although the primary purpose of the S3P protocol is to share strong secrets, the design of the protocol does not assume that s is strong. The S3P protocol can also be used simply to allow a remote authentication service to authenticate a user to a "stateless" host which is local to the user. In this case s may be an authenticator for the audit trail. In our design we make no restrictions upon what the shared secret s is used for once the S3P protocol run has ended: s may be revealed, used as a one-time pad, a cryptographic key, as a salt or an initial value. At a slightly more general level, we remark that it appears very difficult abstractly to model security protocols in a formal way that takes adequate account both of the cryptographic properties assumed, and of the security service provided. Protocols may legitimately be used in ways not explicitly considered by their designers, and the safety of the resulting applications can depend in an unknown way upon the safety of obscure number-theoretic hostages which were abstracted away in the construction of the threat model.

Our S3P protocols make no use of hash functions or symmetric cryptography. However our protocols rely completely for their properties upon the security of the public key systems used. Consequently it is necessary for the moduli to be uncrackable for at least the lifetime of all secrets (weak or strong) used or agreed with that modulus. This provides a sufficient number of bits to provide a strong secret and a strong confounder, together with two nonces. In contrast with the confounder, the nonces can be searchable, so long as the most likely nonce is less likely than some system threshold parameter. Again, we remark at a general level the need to balance the bit-budget carefully when tuning the performance of security protocols which use public key cryptography.

Our S3P protocols also rely upon the ability of those using them to generate irreproducible random bit patterns, and to delete information irrecoverably. These are both interesting technical challenges. In particular the task of finding a suitable source of randomization, upon which (in the context of a particular system) it would be impractical to eavesdrop is one which would repay further study. As a final remark of a general nature, we stress the importance of explicit consideration, when specifying the threat model, not only of the hardware platform supporting the security protocol, but also of the system context, and the security policy under which the hardware will be configured.

28 January 1998; revised July 1998, January & November 1999, June 2000, October 2004. Contact: `b.christianson@herts.ac.uk`

References

1. Anderson, R., Lomas, M., 1994, Fortifying Key negotiation Schemes with Poorly Chosen Passwords, Electronics Letters, 30(13) 1040–1041.
2. Anderson, R., Needham, R., 1998, Programming Satan's Computer, Springer LNCS 1000.
3. Bellovin, S.M., Merritt, M., 1992, Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks, Proc IEEE Computer Society Symposium on Research in Security and Privacy, Oakland 92, 72–84.
4. Diffie, W., Hellman, M., 1976, New Directions in Cryptography, IEEE Transactions on Information Theory, 22(6) 644–654.

5. ElGamal, T., 1985, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory, 31(4) 469–472.
6. Gong, L., 1995, Optimal Authentication Protocols Resistant to Password Guessing Attacks, Proc 8th IEEE Computer Security Foundations Workshop, 24–29.
7. Gong, L., Lomas, M., Needham, R., Salzer, J., 1993, Protecting Poorly Chosen Secrets from Guessing Attacks, IEEE Journal on Selected Areas in Communications, 11(5) 648–656.
8. Hardy, G.H., Wright, E.M., 1978, An Introduction to the Theory of Numbers, 5th edition, Oxford University Press.
9. Jablon, D.P., 1996, Strong Password-Only Authenticated Key Exchange, Computer Communications Review, 26(5) 5–26.
10. Kelsey, J., Schneier, B., Wagner, D., 1998, Protocol Interactions and the Chosen Protocol Attack, Security Protocols 5, Springer LNCS 1361, 91–104.
11. Lomas, M., Christianson, B., 1995, To Whom am I Speaking? Remote Booting in a Hostile World, IEEE Computer, 28(1) 50–54.
12. Lucks, S., 1998, Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys, Security Protocols 5, Springer LNCS 1361, 79–90.
13. Patel, S., 1997, Number Theoretic Attacks on Secure Password Schemes, Proc IEEE Computer Society Symposium on Research in Security and Privacy, Oakland 97, 236–247.
14. Rivest, R., Shamir, A., Adleman, L., 1978, A Method for Obtaining Digital Signatures and Public Key Cryptosystems, Communications of the ACM, 21(2) 120–126. Operating Systems Review, 29(3) 22–30.
15. Wheeler, D., 1997, Transactions using Bets, Security Protocols 4, Springer LNCS 1189, 89–92.

Secure Sessions from Weak Secrets
(Transcript of Discussion)

Bruce Christianson

University of Hertfordshire

This story starts with a specific protocol, which we thought up in the Eagle back in 1997, and which was inspired by an earlier Security Protocol Workshop[1]. Here's the protocol, which is intended to leverage a weak shared password k up into a fresh random strong shared secret s:

$$\begin{array}{ll} A \rightarrow B : (N, e) & \text{an RSA public key} \\ B \rightarrow A : (z^e + k) \bmod N & \text{where } z = c|s|n_a|n_b \text{ is random} \\ A \rightarrow B : n_a & \text{A knows } k, d \\ B \rightarrow A : n_b & \end{array}$$

We thought it would be nice if we could *prove* that our protocol had the properties which we wanted to have. Then we discovered that there was a gap between what we could prove (or what other people did prove) and the kind of property that we actually wanted.

The Scenario. Here is the scenario in a little more detail. Alice and Bob have already agreed a weak secret (which I shall refer to as a password) and they want to agree a fresh strong secret. Neither of them can preserve both (1) the secrecy and (2) the integrity of a (3) strong bitstring in between protocol runs, but they both can meet any two out of these three requirements.

I won't go at any length into the motivation for this restriction. Maybe Alice or Bob is moving from one piece of hardware to another in between protocol runs. Maybe the hardware is unsecure between runs. Maybe Alice and Bob really are human. But a good application to have in mind for this kind of protocol is the task of ensuring that the right person is using something like a mobile telephone. We look into the camera on telephones, we touch them with our fingertips, we type PIN numbers into them, we speak into them, it's a biometric paradise. Local authentication shouldn't be a problem.

However the best way to ensure that an attacker can't get an important secret out of the telephone when it's not being used is to ensure that there are no secrets in there in between runs. So Alice checks the hardware for tampering, types in her password and then runs the protocol with Bob (who might be a server) to agree a strong fresh key and perform mutual authentication. Then she can download all her bits.

This sort of scenario places further assumptions on the hardware: that the hardware can create good random numbers, do cryptography, exchange messages

[1] Blake-Wilson and Menezes, "Entity Authentication and Authenticated Key Transport Protocols", LNCS 1361, 137–158.

(subject to active attacks) and forget secrets. This last assumption is particularly problematic at the moment. I also assume that the implementations of the protocol are good, and in particular Alice and Bob perform all the specified checks, on the correct bit patterns.

Any assumption is really a statement that certain threats are going to be ignored. I'll come back to this point later. However subject to the assumptions listed there, the position paper contains several protocols that (we assert) do more or less what we want.

At the end of the run, Alice and Bob know that each of them has engaged in a protocol run with the other, and that they now share a fresh strong secret with each other and with nobody else.

Question: What are you assuming about the active attacker?

Reply: We assume that Eve can't access Alice or Bob's hardware while they're actually using it, and that Eve doesn't know the real password. Since the password is weak, the attacker could get lucky and guess it correctly. Then there's no hope. But we want to limit the guessing attack by forcing it online.

Hash Functions with Mystic Properties. We looked at what was done by people who analyse these kinds of protocols. The protocols which they analysed had things like strong hash functions in them, which seemed to be there just in order to make the proofs work. Essentially, if you assume that the hash functions had certain mystic properties, then the protocol is correct. (Or equivalently, if the protocol doesn't have the required properties then the hash functions were not magic after all.)

But these papers didn't quite provide implementations for any genuinely magic hash functions. And we rather wanted to analyse the protocol that we actually had, which generates a particular set of bits at each stage. We can even tell you exactly what these bitpatterns are.

I'm one of those philistines who likes to understand a simple protocol before trying to understand a more complicated protocol. And as with any countermeasure, I want to know what value-added security the hash functions are actually buying. What threat are they protecting against. What attacks would succeed if they were not there?

Here's an example. The protocol I started with uses n_a and n_b as verifiers to prove knowledge of the shared secret s. Now versions which are formally analysed[2] usually use (in effect)

$$h_a(z), h_b(z), h_s(z) \quad \text{in place of} \quad n_a, n_b, s.$$

[2] For an excellent example of an analysis of an RSA-based protocol similar to the one at the start of this talk, see MacKenzie, Patel and Swaminathan, "Password-Authenticated Key-Exchange based on RSA", Proc Asiacrypt 2000, LNCS 1976, 599–613. For an analysis of a protocol based instead on Diffie-Hellman, see MacKenzie, "More Efficient Password-Authenticated Key-Exchange", Proc Top Crypt 2001, LNCS 2020, 361–377.

Here the h_x are a family of hash functions with the necessary mystic properties, properties which we can deduce by looking at the proof.

There's no harm in the verifier space being searchable, so long as the verifiers n_a, n_b are much harder to guess than the password k. On the other hand, hash function source values mustn't be searchable.

This gives the implementer a dilemma. If the implementations of the hash functions really do have the properties required of the idealizations, then the more bits input to the hash the better.

But now suppose it turns out that my implementation of h_x (and remember, this implementation is not supplied with the kitset and so isn't covered by the warranty) is not really one-way. Then the protocol with the hash functions is less secure than the version without them. Indeed, the more things are added to the hash input, and the less searchable the verifiers are made, the more information the protocol leaks.

This sort of dilemma is a genuine practical problem. There are some valuable things to be learned from looking at practical details. For example, it's often a good idea to agree a key that's much longer than the one you actually need, and then throw most of it away, but you won't learn this if you just follow the logic, which usually doesn't address bit-budget issues carefully.

A reasonable compromise, given that z is much longer than we need, would be to set $n_a = h_a(z_a)$ etc, where now $z = z_c|z_s|z_a|z_b$. But of course this in turn changes the nature of our reliance upon assumptions about the hash functions.

Changing Assumptions. Another reason for trying to keep protocols very simple is that a protocol implementation which works will often end up being used in a slightly different context. [laughter] It's so tempting isn't it.

For example I might decide that I don't want to assume that the fresh strong secret which the protocol shares is actually strong: I might want to use the S3P protocol to agree one or two bits that I'm going to hand off to another protocol to settle a bet[3]. I might not want to assume that the secret which I'm agreeing is going to be used as a secret key. It might be used in a way that makes it public as soon as it's agreed, for example as a shared pseudorandom seed. I might not want to assume that the password is weak. There was a time when I might have wanted to assume that the password was exactly forty bits long, and that I was using this protocol to agree a 128 bit key. I might want to change my assumptions about what can be done in between runs with the hardware, and about interactive breaking attacks.

And as I said earlier, from the analysis point of view, each of my assumptions really corresponds to a constraint on the threat model. When I say, the system can create random numbers, what I'm saying is my threat model doesn't consider an attack where the attacker learns a later value of the random seed and uses that to work out what the pseudo-random sequence was yesterday.

When a proof assumes tamper-proof hardware, that means that the proof doesn't consider any threat that involves tampering with the hardware. And

[3] D. Wheeler, "Transactions using Bets", LNCS 1189, 89–92.

when a proof assumes magic hash functions that means that the threat model just doesn't consider any cases where the magic wears off, potentially even years later.

Changing the threat model changes the properties which we need the hash functions to have. Inserting hash functions in such a way as to make a protocol have one set of properties, very often mucks up the other properties that the simple protocol might have had if the magic hash functions hadn't been added. I don't mind having to re-prove properties under new assumptions, or even having to restate the properties slightly. But I really don't want to have to re-implement the protocol. Think of the testing.

And at some point we really do have to implement these things. So how do we model the protocol, and how do we model all the threats, in such a way as to capture all the security properties that we might want it to have?

Changing Abstractions. The essential problem is that in real life we need to consider security protocols at several different levels of abstraction, and the various refinements which we need to make to get to an implementation typically do not respect the abstraction boundaries[4].

For example, the protocol I showed you at the beginning can be subjected to a narrowing attack. Suppose the part of Alice is played by Eve, and Eve deliberately chooses a bad value of N, which forces Bob into a narrow semigroup. For example, if Eve could choose $N = pq$ where $p = \alpha \cdot e + 1$, then $(z^e)^{r(q-1)} \bmod N = 1$, and so Eve can find the password by a process of elimination, possibly quickly enough to complete the protocol run.

One cure for this is to ensure that e is relatively prime to $\phi(N)$, for example by requiring that e be a prime with $e > N$, or alternatively[5] with $e^2 > N$ and $N \bmod e$ neither zero nor a divisor of N.

But it's really very hard to see how to model the protocol in a way that captures all the horrible potential for interactions between the number theory used by the cryptography and the protocol interleaving caused by the application semantics[6].

Another example is the shifting attack on the ElGamal version of the S3P protocol (this attack is described in section 5 of the position paper). Adding hash functions with appropriate properties would avoid the need to consider interaction attacks like this one explicitly. But understanding the properties which the hash function needs to have is enough to see how to block the attack. Then we don't need actually to put the hash functions into the implementation at all.

[4] For more on this theme see for example Chapter 10 of the book "Modelling and Analysis of Security Protocols" by Ryan, Schneider et al, Addison-Wesley 2001, ISBN 0201674718.

[5] If prime e divides $\phi(N)$ then either e^2 divides N (impossible if $e^2 > N$) or else e divides $p-1$ for some prime factor p of N. But then $p \bmod e = 1, N \bmod e = N/p \bmod e = N/p < e < p$. The condition is sufficient not necessary: 7 is relatively prime to $\phi(15) = 8$ but 15 mod 7 = 1.

[6] For a splendid example in the symmetric key case see the presentation "What we can learn from API security" by Anderson and Bond, these proceedings.

The dangerous temptation is to attempt to define away such attacks, by inserting protocol elements such as random hash functions in such a way as to "remove" the possibility of bad interactions, and then to prove properties of clean abstractions instead of of the implementations which refine them[7].

But there's another, possibly deeper, problem arising from the necessary use of abstractions.

Protocol Layering. When we deploy protocols like S3P, we typically want to layer them in with other protocols, or compose them with some sort of optimistic regime[8] or with some kind of fault-tolerance[9], or to hand off the artefacts (such as a fresh session key) to some other protocol which does something else with it. This introduces further subtleties into the modelling process.

I want to talk in a bit of detail about one important case, where the S3P protocol is used as a tripwire. In this case the S3P protocol is enclosed within an outer protocol wrapper. The outer protocol header on the front contains all the robustness fields that good sense says should be there: which protocol is this, which run does this message belong to, which message is this in that run, what's the hash of all the shared protocol states you've been in during this run, and so on.

Now there may be good reason to believe that this outer wrapper is itself a completely secure protocol, so there's no way that that Alice could not be talking to Bob to begin with. The inner key agreement protocol is being used as a tripwire to find out if anybody has breached the outer defences. It's both a tripwire and last line of defence: even if Eve can get into the Castle she still can't get into the Keep, and now Alice or Bob knows that she is inside, and maliciously so. If Eve trips over the tripwire then it's not an accident, it means she climbed over the dyke, swam the moat, scaled the battlements, and then tripped. Eve can't claim she was just on her way home. [laughter]

But this means that we need to have very clear models of what constitutes a failure. What constitutes somebody falling over the tripwire? When does somebody intend to present a particular bit pattern to this protocol? What if a message got corrupted accidentally in transit?

Suppose Alice sends a message, it doesn't seem to get through, Alice sends it again, OK, now Bob receives the same message twice: is that a replay attack? Or just a protocol failure at a lower level? How do we model the interactions between the different levels of abstraction here? At what level(s) of abstraction do we log events?

[7] I should make it clear that the people who actually do the formal proofs are careful to a fault about making clear their limitations. The lovely papers by Phil Mackenzie which I mentioned in earlier in footnote 2, and the book by Ryan, Schneider et al in footnote 4, are models of good practice in this regard. The problems with maladapted refinement and abstraction boundaries tend to occur further along the food chain.

[8] B. Christianson, J. Snook, "Shrink-Wrapped Optimism: The DODA Approach to Distributed Document Processing", LNCS 2845, 74–95.

[9] G. Price, "Broadening the Scope of Fault Tolerence within Secure Services", LNCS 2133, 155–169.

What is a timeout, at the S3P level of abstraction? These messages may be travelling by fax, or by post, or by courier. There's a huge conceptual difference between a timeout of three seconds and one of three Megaseconds. It's not just a matter of putting a larger integer in a configuration file, it's a qualitatively different threat model. Similar remarks apply to interactive breaking and the level of re-entrancy, and I'll come back to this point in a minute.

Non-deterministic Failure Detection. What should we do when we detect a failure? One possibility is the Strangelove scenario: set off a doomsday device that will immediately bring the world to an end. That seems to be what usually happens in the model. But in real life, protocol failures are bound to happen eventually, probably more than once. Sometimes it's significant, and sometimes it's not. Sometimes we just have to push the car back up the cliff and see if it happens again.

For example, consider Eve's fundamental attack of attempting to guess the password k. It really isn't acceptable to allow no wrong guesses at all. Even users as experienced as Alice and Bob do occasionally mistype. But if you have a fixed limit (say three wrong guesses on a single user ID and you're out) then Eve can make just two guesses for each user. Provided there are enough users and their passwords are chosen semi-independently then Eve is sure to find somebody's password, and then she is in.

So there are advantages to taking non-deterministic actions on failure detection, because then the defenders don't need to collude. If everybody follows the same probabilistic regime, then the alarm is raised with the acceptable degree of certainty after a distributed guessing attack above a certain threshold level.

Once again, the combination of layering of abstractions and non-determinism at some of the layers, means that it is not easy to see how to do the modelling with the current approaches.

Remember that non-determinism is a property of the model, the abstraction, not something which is "really" either there or not in a fielded system. Consequently there's the potential for horrible interactions between the abstractions, just like the interactions between number theory and application semantics.

Re-entrancy. We've got to allow re-entrancy, because the S3P protocol might legitimately be run several times back to back. Alice establishes a one-way session with Bob, Bob immediately establishes a call-back channel with Alice, and so on. But maybe what appears to be Bob is actually Eve, using message reflection to get Alice to act as an oracle and tell Eve what the correct response is (see section 6.4 of the position paper).

In this case, at the end of the run Alice shares the secret only with herself. True, nobody else knows the secret, but remember one of our primary security requirements was that if Alice thinks she has successfully completed a protocol run with Bob then she must have done so. Now Alice is going to be willing to swear that Bob was alive at 9.30 yesterday evening, when in fact he was murdered that morning.

But we need the boxes running the protocol code to be stateless, or rather we don't want to assume that the state mechanism is reliable. We don't want to have to retain or correlate state between protocol runs, except for performance reasons.

There are various nice solutions, but which the usual models would not only obscure but actually make harder to use. You've got to model the state at different levels of abstraction, and you've got to model persistence of state at each level, and model the interactions between the levels.

The idea is that protocols should proceed optimistically, in order to get good performance. Verification can usually afford to go more slowly[10]. Usually everything will turn out to have worked OK, and when you find out that it didn't then you have a possibly non-deterministic action[11] which you take when that happens.

In this way you can even tell whether the violation of the inner state is deliberate or accidental[12], a point which the conventional analysis doesn't catch.

Every proposition is affirmative, therefore no proposition is negative. To conclude, here is another version of the RSA protocol which I showed you at the beginning of this talk.

$$\begin{array}{ll} A \rightarrow B : (N, e) & \\ B \rightarrow A : (z^e + k) \bmod N, w & \\ A \rightarrow B : n_a, v & \quad \text{where } v^e = w \pmod{N} \\ B \rightarrow A : n_b & \end{array}$$

Usually in order to prove that a protocol like this is correct I would prove something like $P \Rightarrow Q$, *if* Eve can learn more about the password than the fact that her guess was wrong, *then* RSA is broken in the sense that Eve can decrypt unpredictable plaintext, or something like that. But for this protocol it is trivial to prove the proposition Q in the form:

$$\begin{array}{l} \forall (N, e) \text{ chosen by Alice} \\ \quad \forall w \text{ chosen by an independent referee and given to Eve} \\ \qquad \text{Eve can determine } v : v^e = w \pmod{N} \end{array}$$

whence $P \Rightarrow Q$ for any P.

Far from establishing the correctness of this particular protocol, however, this 'proof' doesn't help at all to determine whether or not the protocol is broken. Is it broken? I wanted to to finish by having a dig at the random oracle model but I'll leave the details as a homework exercise.

[10] Again, see LNCS 2845, 74–95 (footnote 8) for details.

[11] Remember once again, that non-determinism is a property of the abstraction, not the thing. See B. Christianson, "Auditing against Impossible Abstractions", LNCS 1796, 60–64.

[12] Mark Lomas and Bruce Christianson, "To Whom am I Speaking?", IEEE Computer 28(1), 50–54, 1995.

Panel Session: Is Protocol Modelling Finished?
(Transcript of Discussion)

Mike Roe (Chair)

Matt Blaze: I think we have gotten very close to the single biggest practical problem which protocol modelling and analysis has, but without actually mentioning it explicitly. That problem is that we tend to design very secure protocols that in practice cannot be implemented correctly. Larry mentioned SSL, which is very close to this problem. When we design these protocols we are sending them to implementers, and implementers make assumptions, they put protocols into environments, and they make what seems to them to be a very unimportant modification to the protocol.

Bruce Christianson: A low level design decision!

Matt Blaze: Yes, they make what seems to them to be a low level design decision, but the modification has the effect of just breaking the protocol completely. Is there some hope for formal analysis of protocols to capture a distinction between an implementable protocol and a non-implementable protocol?

Mike Roe: It's not really so much that we need more formalism, as that the people doing this formal analysis need to be talking more to the people that deliver requirements, and producing a protocol that meets with those requirements. We're seeing a great many instances of square pegs being fitted into round holes because implementors read books that give them some protocol which is secure under a particular combination of assumptions. The actual assumptions in the environment which the implementors had in mind are completely different, but never mind, it's a security protocol, we'll just tweak it. This happens a lot and it's a problem.

Virgil Gligor: Then it comes back to Matt's question. Peter Ryan's presentation shows that there is a gap between enforceable policies (things like access control checks) and non-enforceable policies which are given by what we used to call *non-properties* (properties that don't follow in the Alpera-Schneider framework).

There are two large classes of security properties, specifically of logic information flow properties, which are similar to liveness. One was called security, and the one other was called compatibility. I once pointed out to Martin Abadi that there are these large classes of properties that couldn't be composed under these criteria, and he challenged me with the following statement: Why can't we take all these non-enforceable properties and *approximate* them using enforceable properties?

Let's take your favourite information flow policy as an example. You give me a statement of the policy, which is about approximating the safety properties given by certain checks. Clearly there is a gap, and this gap comes from the theory. In practice, we know how to put in access checks, and we do that very competently and then enforce it.

So my answer to the challenge is the following. There are lots of examples (of which I have at least one) in which I take approximations of two separate and distinct information flow policies, so one is the Bell-LaPadula approximation and one is the Biba approximation, and these both give information flow policies. I take those two approximations, make practical systems to do these things, and compose them by conjunction, and what I get is an approximation of a policy which is weaker than either of the previous independent flow policies.

This is in a sense composing two protocols and getting a worse one as a result. So I can answer Matt's question, not in the general protocol sense, but very precisely in this particular case. I know exactly where the gap comes from in this case, it comes from the access control policies. I'm sure the problem is larger than this one example, I'm sure we can find the same kind of thing with protocols. We specify protocols which in some sense may not be implementable in practice because of some properties that we need to know about, but only approximations of that knowledge exist, just like for the access control problem.

Wenbo Mao: OK, I have a feeling that this thing looks like a composition. It's a major problem, we compose simple things together and the things which we compose are very simple and well understood but once you compose them they become massive. This is actually a very fundamental thing, I regard it as being like machine translation of natural language, yes, very hard so far, although translating every word can be very precise. The problem is that you want the machine to do something which involves composing simple things together.

Virgil Gligor: The composition is only a consequence of the gap, or rather you only notice that there is a gap between the real property and its protocol instantiation (or its security translation) when you try to compose things. It's tempting to say that this is just a nuisance, but I am sure that there will be lots of properties in practice where the gap will come and haunt us even if we don't try and compose them.

Wenbo Mao: Well in fact the analysis is like composition because it makes things hard, but actually in safeguarding a particular action, composition must be carefully used. For example, you compose a hash function with RSA, with all these number-theoretically nice features. After composition it looks like it will be harder to attack. Of course when you run an analysis you must say, OK my composition is idealized. There is a gap between the analysis result and the real composition. The gap is, asserting that the composition is equally hard for everyone. Hard for an enemy to attack, hard for each one of us. Now I could add this assertion to some idealisation and so fulfil my analysis, but the trouble is that then my idealisation is always going to assume a good interaction. That's not good for analysis. So I assume this composition problem to be more fundamental. It's like the Turing test.

Bruce Christianson: When we have an idealisation, even if it's a good idealisation, there are often other implementations of the same idealisation and which are bad instantiations. And, going along the same lines as what Virgil said, this is because the idealisation is failing to capture some *anti*-property, some property which we need the algorithm to *not* have.

It's often easy enough to say what properties we need the algorithms to have, and to prove, or to assume, that they have those properties. But we also want them not to have certain other properties, and often we don't know what those anti-properties that we want them to not have are, until we've looked very carefully at the system contexts, or the configurations used, or very itty bitty details about how the hardware is manufactured, and so on. Knowledge of these anti-properties kind of trickles down from analysis of attacks rather than being synthesised directly from the security protocol requirements, and I don't know what to do about that.

Birgit Pfitzmann: As to cryptography, which is of course not at all the whole problem but which stimulates the definitions we are making, I think we can actually solve this problem. I mean solve it to the extent that if you have an idealisation, and have any implementation that can be proved, (and I assume we have proof as secure as this idealisation) then you can rely on it for all sorts of privacy properties, which are the typical thing in cryptography, where you say that something should not have some particular information flow.

Now on the one hand it's only cryptography, it doesn't do anything about the socio-economic system of course, and on the other hand these are pretty strong notions, so there are lots of things that you might want to idealise in certain ways and you just can't prove them in this way (I mean really mathematically prove those composition theories) because there would be a mistake.

Bruce Christianson: It comes down to the details of proving that these refinements are at least as secure. The problem is unpacking that. Suppose at some time you discover something you don't want the attacker to be able to verify, some very explicit conjecture. In a lot of cases the proof already tells us the attacker can't calculate the correct value to return in order to continue the protocol, but often it's enough for the attacker to be able to *use* the eavesdropped correct value to verify a guess.

Birgit Pfitzmann: I see no problem. A standard property is like a temporal logic property. If the ideal system you chose doesn't have this property, and later you notice you want this property, then you put it in. Then you can rely on it for the real system, provided you did the cryptoanalysis.

Simon Foley: One question that always comes to my mind is, what do we mean by composition. If you look back at non-interference, when we first started looking at composition we'd say, oh non-interference is not composable.

Later on when people started casting non-interference properties within process logics, which have a well understood notion of composition, people started discovering that, hey it actually is composable. So before we start making claims about composition, we need first of all to say what exactly we mean by composition. Are we using forms of composition that are inherently broken?

Virgil Gligor: In my case in terms of composing those two approximations, Bell-LaPadula and Biba, the composition is straightforward conjunction.

Larry Paulson: I will not claim that it's impossible, but we just don't know how to do it as we do when we write programs. If I write a routine and I call a square-root function I can assume it will return the square root and I don't have to ask what algorithm was used to compute it. Whereas if you use one protocol and it invokes another protocol, somehow we think most of the time it should be OK, and that the subsidiary protocol will do what it should and no more or less, and yet we have no rigorous way of saying that that is really the case.

Bruce Christianson: And for example does the implementor have to use a different modulus for the Diffie-Hellman in the second protocol from the modulus in the first one?

Ross Anderson: There appears to be a growing gap between the kind of results that people prove about protocols, and the ways in which protocols fail in real life. This may be in some sense a result of the success of the early approaches such as BAN and tuning people's awareness to the ways in which the obvious things fail. It may also be because logics have difficulty dealing with number-theoretic aspects of protocols, and perhaps much more importantly nowadays, what happens with state. It's very tempting, if one is a theorist, to go and prove yet another theorem on top of the existing structure, in order to get a publication. There is a risk that the theoretical protocols world, and the practical world where we're breaking these things, will get further and further apart.

Larry Paulson: I think that one of the biggest problems is, how do we get programmers to comply with the most elementary hygiene precautions.

We prove a protocol at a high level, and the implementation is subject to a buffer overrun attack because the implementors are not even checking their buffers. I believe that the Internet worm in 1988 should have told the whole industry that buffer overflow was very serious and should not be allowed, and that one should not allow programmers to use the unsafe libraries which are vulnerable to attack. But nothing was done, and so over a decade later we find that our highly verified protocol is subject to a buffer attack. There's not a darn thing we can do about it if these guys will not practice the most elementary programming rules they should have learned in their first year at University.

Birgit Pfitzmann: You could try to verify the actual code. I think there are some tools where you can do abstract protocol compilations, and then there are some tools which try to verify code and make it map on to the protocols. I don't know whether any of these are tools in which there is analysis of security.

Larry Paulson: You could, but I'd like to see people behave like professionals first. *Then* if they still need help, yes, let's help them. [Laughter]

Mike Roe: I think you're being a little bit unfair now, certain well-known computer companies are spending exceedingly large amounts trying to get buffer overflows out of their code.

Larry Paulson: I believe that already in the 1980's there existed a design for safe compiler libraries, in which you indicate the size of the buffer, and this could then be checked. It would have been relatively straightforward for someone like William Gates to say, never use an old library, and at least then the problem would not grow. Over time they could have rewritten the protocol code which used the old library. This sort of approach managed to cope with the year 2000.

John Ioannidis: It's not just the libraries. If you read a buffer from the safe library that's 100 bytes and try to copy it into a buffer that's only 50 bytes, then there is no amount of library safeness that will help you.

Mike Roe: I think we are wandering off the subject a little bit. All this amounts to a refinement problem. When you go from the protocol that you proved to the actual implementation, you will be introducing an awful lot of things (like what the environment really is) which were not in the original protocol specification.

Larry Paulson: I believe one *could* separate the proof of security from the proof of correctness, in other words, prove the high level of protocol correct, and then define abstractions from the abstract messages to bit patterns, and then prove that the code implemented this correctly without proving security a second time.

Bruce Christianson: Even that would be a huge advance. It is a refinement problem, just as Mike says, but the refinements typically do not respect the abstractions used to model the cryptography.

WAR: Wireless Anonymous Routing

Matt Blaze[1], John Ioannidis[1], Angelos D. Keromytis[2], Tal Malkin[2], and Avi Rubin[3]

[1] AT&T Labs – Research
{mab, ji}@research.att.com
[2] CS Department, Columbia University
{angelos,tal}@cs.columbia.edu
[3] Johns Hopkins University
rubin@jhu.edu

Abstract. There have been many efforts to anonymize traffic in traditional point-to-point networks. However, these techniques do not map directly onto wireless systems because the two types of networks have different properties. For example, whereas in wired networks adjacent communication links must know each others' addresses, wireless networks broadcast messages. Thus, an adversary may be able to obtain global information about network traffic. On the other hand, a wireless receiver's identity may not need to be known to anyone.

We introduce *Wireless Anonymous Routing (WAR).* We describe a threat model for WAR and compare it to its wired counterpart. We show that traditional anonymity protocols are either insecure or perform poorly when applied to the wireless model, and we describe new protocols that preserve security with better performance.

1 Introduction

Anonymity (and resistance to traffic analysis, generally) is an interesting and difficult problem in computer networking; many modern networks, including especially those based on the IP protocol stack, inherently identify the sources and destinations of traffic to routers, gateways, and ancillary servers. An especially difficult aspect of this problem involves hiding the identity of peers from the correspondents themselves; most non-trivial protocols assume the ability to exchange round trip traffic.

A number of schemes for anonymity in these networks have been developed in recent years; most are variants on "MIX networks" [1] in which traffic is routed among participants in the network in an effort to hide the true source and/or destination of the messages. The details of these schemes vary according to such things as whether and between whom cryptography is used, how membership is managed, at which layer of the protocol stack they operate, who is the identity hidden from, and so on. What virtually all of these schemes have in common is the assumption (and exploitation of the fact) that the underlying communication topology consists of a network of point-to-point links, in which the endpoints of the links may be known to one another, but where it is infeasible for the adversary to obtain global information about network traffic.

Unfortunately, this assumption excludes an important class of networks (one in which many of the most interesting applications of anonymity may be found): wireless communication systems. Wireless networks invert the "point-to-point" link property of most Internet-based networks; here, the physical location and identity of the sender may be difficult to conceal, but the receiver, being generally "passive," need not reveal its identity to anyone (not even the sender).

These differences also break (or render terribly inefficient) the "standard" anonymity protocols designed for point-to-point networks. At the same time, the broadcast nature of such networks tends to exacerbate privacy and traffic analysis issues across a wide range applications, making anonymity an especially important issue for many different kinds of users of these networks. The applications and users concerned with wireless anonymity are remarkably diverse, encompassing a spectrum from private individuals (*e.g.*, privacy of consumer behavior) to advanced military communications (*e.g.*, hiding command structure, protecting sensor networks, *etc.*).

In this paper, we advance the notion that wireless networks are an interesting and largely unexplored research area for anonymity and identity protection. In particular, we introduce a security model for privacy in such networks and a suite of protocols that provide anonymity under the model.

2 The WAR Model

2.1 Communications Infrastructure and Environment

We consider a wireless communication network, where stations can communicate only through a public wireless channel (*i.e.*, broadcast). Every node listens for transmissions in "promiscuous mode". Nodes can exhibit a high degree of mobility, and can join and leave the network at any time. Some nodes might have a gateway to another network (*e.g.*, the Internet).

For simplicity, we assume a wireless broadcast medium in which only one node at a time can transmit messages, which are potentially received by all other nodes within communications range; receipt of a particular transmission by any given node is not guaranteed by the medium itself, however. As a consequence of transmitting, the node may reveal identifying information (such as its physical position or unique transmitter characteristics, *e.g.*, the exact oscillation frequency of its crystals) about itself. Receiving a message, on the other hand, is entirely passive, and does not reveal any information about the receivers. Every node can act as both a transmitter and a receiver, but is identified to others only when it transmits. In other words, we assume abstract characteristics similar to those of most conventional general-purpose radio-based schemes, including, *e.g.*, commercially available wireless networks such as 802.11b systems operating in "ad-hoc" mode.

Each node transmits packets, called here "radiograms" to distinguish them from IP packets or higher level message abstractions.

Radiograms can be received by any node in range on the channel. All nodes receive in "promiscuous mode" and process all messages sent over the channel. Nodes enter and leave the network (and go in and out of range) as they please; different nodes may have different views of who the other nodes in the network might be. Some nodes might have a gateway to other networks (*e.g.*, the Internet); routing to and the identity of such nodes would be defined by a persistent endpoint identifier (see below).

Each node has (or can generate) public/private keypairs for use by the anonymity protocols as *online identifiers*. These keys define the "online" identity of each node and are separate from any identity used at the application layer. These keys may or may not be certified by a central or distributed authority, depending on whether the network is to be "open" to anyone (*e.g.*, a public wireless service) or "closed" for use by a private community (*e.g.*, a battlefield network used by one side's military force).

We also assume that each node has one or more persistent public/private keypairs for use as *endpoint identifiers* that provide whatever security is required at the application layer. These keys can be used by applications to authenticate and encrypt messages but are not used by (and are logically decoupled from) the anonymity protocols.

2.2 Adversary Model

We consider two types of adversaries: a *listening adversary* and a *Byzantine adversary*, each of which can be either *non-adaptive*, *adaptive*, or *pro-active*.

A listening adversary can monitor all traffic, and identify the sender of each radiogram. In addition, the adversary may be able to monitor the internal state of a set of current stations (including secret keys, randomness, computation of radiogram to be sent, *etc.*). This adversary is honest-but-curious, namely follows the prescribed protocol in each node, but tries to deduce information in order to compromise anonymity.

A Byzantine adversary is even stronger. Like the listening adversary, it can monitor all traffic and the internal state of some set of stations. In addition, it can also maliciously control up to t stations, for some parameter t. The adversary can make controlled stations behave arbitrarily, including injecting new radiograms, dropping radiograms, or changing radiograms that were supposed to be transmitted.

We can make a further (by now standard) distinction between a non-adaptive, adaptive, or pro-active adversary, depending on when the corrupted nodes are chosen. A non-adaptive adversary chooses the set of nodes to be corrupted (and the t nodes to be controlled in the Byzantine case) at the beginning of his attack. An adaptive adversary may choose which nodes to corrupt as his attack progresses, based on the information gathered so far (but without violating the access structure). A pro-active adversary can adaptively choose which nodes to corrupt, but can also choose (synchronously) to shift its corruption from a certain node to another, granting the previously corrupted node healed. Here, the only restriction is that the adversary does not violate the access structure in any particular time period. More details on pro-active security (in a different context) may be found in [2,3,4].

2.3 Security Goals

Our goal is to maintain anonymity of both source and destination of a given message. Clearly, if an adversary corrupts the source, he can see the originated message m and its destination[1]. It is also clear that if the adversary corrupts the destination node, he can figure this out upon delivery of the message. Anonymity should be maintained for all other cases (including anonymity of the source when the destination is corrupted).

Consider a listening adversary who corrupted a set C of nodes, and denote its view after an execution of a protocol in which a source p_i delivers a message to a destination p_j by the random variable $V_C^k(i, j)$, where k is the security parameter (this view contains all radiograms sent in the network, and the internal state of all nodes in C).

Source and Destination Anonymity to Observer: We require that for any message delivered, as long as the source and destination are not corrupted, the adversary's view reveals no information about their identity.

Complete Anonymity: A stronger requirement is that as long as the source is not corrupted (even if the destination is), the adversary's view reveals no information about the identity of the source. If the destination is not corrupted then no information about its identity should be revealed either. Note that if the destination is corrupted, the message can be decrypted, which might contain the identity of the source (we do not restrict the content of messages!).

An additional goal which may be desired, is that of **hiding the existence of communication**. That is, an adversary cannot even determine whether any communication is taking place (unless he has corrupted a source or destination of such communication). Instead of formalizing this goal directly, we formalize an even stronger goal which implies it, is achieved by our main protocol, and has other advantages.

Anonymity Per Radiogram: For each radiogram R that is sent during protocol execution, let p_i, p_j be corresponding sender and receiver (where $p_j = \perp$ for a cover radiogram). We require that, as long as $p_i, p_j \notin C$, the adversary's view gives him no information about p_j. In particular, the adversary cannot distinguish the cases where R is a cover radiogram or a radiogram that can be decrypted by a certain p_j.

We note that anonymity per radiogram should not be viewed as a required goal, since the real concerns are with the core traffic (high-level messages).

[1] The source must know the destination to which he sends a message. But note that this destination may be specified in terms of a public key that is a pseudonym of an unknown party, such as a public key that was previously sent within an anonymous message.

However, this is a useful tool since, if achieved, it can be proved to imply source and destination anonymity to observer, as well as hiding the existence of messages.

Security against Stronger Adversaries: We defined our anonymity goals for a listening, non-adaptive adversaries, and our analysis will concentrate on this setting as well. The formal definitions for adaptive or pro-active adversaries, and for Byzantine adversaries are quite complex, and are not included here (some of the issues are similar to those in defining secure computation against active adversaries). For example, if the adversary is adaptive, it is not enough to ask that source and destination remain anonymous as long as the adversary has not corrupted them, since the protocol might allow an adversary to adaptively corrupt parties so that the source and destination are always corrupted with high probability. Such an attack should be prohibited by the definition. Of course, for certain protocols it may be possible to show that an adaptive or pro-active adversary cannot gain more information than a non-adaptive adversary.

We note that a general transformation from a protocol secure against a listening adversary to one se cure against a Byzantine adversary, can proceed by requiring each radiogram 's payload to be authenticated with respect to the public key in its first ID part (and discarding radiograms that fail to authenticate). The anonymity features will be preserved, although reliable delivery may no longer be guaranteed (*i.e.*, the system may become susceptible to DoS attacks). Reliability can be addressed by other means, and is not the focus of the current work.

2.4 Discussion

Our anonymity goals are strong, and should be achieved regardless of the core traffic characteristics. To achieve our strongest goals (including hiding the existence of communication), the pattern of radiograms to be sent by a node should be independent of the volume, content, or any other aspect of the core traffic. That is, the protocols should specify the size and frequency of radiograms to be sent at each node, *e.g.*, by sending at a constant rate (which may depend on the overall traffic volume and group size, but not the core traffic). To an observer, the distribution of these radiograms will look the same throughout the protocol.

We note that weaker goals of anonymity were sometimes posed by previous works, such as MIX-based mechanisms (see Section 4). For example, anonymity may be achieved based on statistical properties of the core traffic (requiring high volume). Or it may be allowed to leak the existence or even content of messages, as long as there is no linkability between a message originated by a source and a message delivered to a destination. We do not elaborate on these weaker goals as our work provides stronger security guarantees.

Finally, we should mention that if an adversary can gain arbitrarily many identities (public keys), then it is clearly easy for him to control a large number (or fraction) of nodes. This problem is outside of our model, and should be prevented at the public-key infrastructure level. However, we note that if a protocol achieves anonymity against an arbitrary access structure $\mathcal{C}$ (without a size limitation), then this attack does not help the adversary. Indeed, this is the case with our main protocol, which achieves source and destination anonymity to observer for arbitrary $\mathcal{C}$.

3 Protocols and Analysis

3.1 Basic Protocols

Here, we describe some basic WAR protocols, which demonstrate general solutions and ideas that will be used by our main construction in the following section. In all the protocols, we assume that each joining node broadcasts its public key, and listens for a while to hear public keys of other (existing and new) members.

From Strong Anonymous Routing to WAR: Previous approaches to anonymous routing in networks require varying anonymity properties, under different adversarial models. Almost all these adversarial models consider an adversary that has access only to part of the network links, and cannot obtain global information about the network traffic. However, few works (*cf* [5] require anonymity against an adversary that can monitor *all* traffic, like our listening adversary. We refer to such protocols as strong anonymous routing protocols.

It easy to see that such protocols can be transformed to our wireless setting, where instead of sending the radiogram R on a link from processor i to j, we have node i broadcast the payload $(j, E(PK_j, R))$, where $E(PK_j, R)$ is the (asymmetric) encryption of R under the public key of node j. Such a transformation preserves the anonymity guarantees, although not necessarily reliability (as the network is dynamic and nodes leave as they please). The transformation also maintains the performance of the original schemes, with respect to a clique network topology. This transformation provides several candidate WAR protocols, such as those adapted from variants of [5].

However, these protocols have weaknesses in terms of performance and security; since such protocols were designed for a fixed point-to-point network, they cannot take advantage of the broadcast channel, and thus impose higher communication overheads. Intuitively, these protocols obtain anonymity *despite* the broadcast channel, but cannot use it for improved performance. Such protocols may also exhibit particularly bad performance in our clique setting, *e.g.*, if performance depends on the number of edges in the network. In terms of security, these protocols achieve anonymity against a listening-only non-adaptive adversary.

From Anonymous Routing to WAR: This protocol starts from any anonymous routing protocol for a point-to-point network (such as a MIX), and transforms it to a WAR protocol using generic mechanisms, as follows.

As a first step, use the broadcast channel to implement secure channels against each pair of nodes[2]. This simply boils down to key exchange among

[2] This requires an authenticated broadcast channel, which can be achieved using the public-key infrastructure of joining members.

each pair (*cf* [6,7], where the key is then used for symmetric encryption of radiograms between the pair of nodes. For example, we can use a Diffie-Hellman-based key exchange, ending up with a secret key of the form g^{xy} for each of the n^2 pairs of current members.

As a second step, we use the established (virtual) secure channels to implement the point-to-point protocol, where instead of sending the radiogram R on a link from processor i to j, we have node i broadcast the payload $E(SK_{i,j}, R)$, where $E(SK_{i,j}, R)$ is the (symmetric) encryption of R under the secret key of the pair (i, j). For each broadcast radiogram from node i, every node j tries to decrypt the payload using its key with node i. If the decryption fails, the radiogram is discarded (as it was intended for a different receiver). If the decryption succeeds, the node continues as specified by the underlying protocol.

For example, we briefly describe the scheme using onion routing [8]. A node that wants to send a message to some group member, chooses at random a set of $M - 1$ current group members, where M is a security parameter. It then creates an "onion" of M encrypted encapsulated payloads. The inner-most layer is encrypted with the public key of the intended destination. Each successive layer consists of an encryption of the previous layers' payload together with the identity of the corresponding group member.

Thus, upon broadcast of a radiogram, the receiver decrypts it to find a payload R consisting of an index j and an encapsulated payload R', requiring the receiver to encrypt R' using its secret key with node j, and broadcast the resulting payload. This continues until the innermost layer is reached (in which case the payload is decrypted to the actual message). Note that this protocol involves only a symmetric decryption operation for each station, per broadcast radiogram, which is quite fast.

Anonymity properties are inherited from the underlying protocol. In particular, using onion routing mechanisms with an onion of depth M, it can be shown that complete anonymity is guaranteed against an adversary who corrupted at most $M/2$ parties in the onion. This is because each node knows the identity of its onion-neighbors (*i.e.*, the sender of the previous radiogram, and the receiver of the next radiogram). This implies that for adaptive adversaries, anonymity is maintained as long as less than $M/2$ are corrupted.

WAR Using Public-Key Cryptography: We now describe a protocol that achieves the best security guarantees of all our protocols. However, its performance is prohibitive for real applications. This will be fixed in our main construction, described later.

Consider again an anonymous point-to-point protocol, such as onion routing. We propose a modification to the protocol above, in which senders of radiograms do not generally know who the receivers are (thus allowing for better anonymity properties). Specifically, the protocol proceeds as the one above, except that the onion does not include the identity of the receiver for each layer, thus departing from the traditional point-to-point onion. We provide some more details about the protocol below, as it will be used as a sub-protocol for our main construction. There are three kinds of payloads in this protocol:

- Cover: Cover payloads are just random bit-strings, and are sent any time a node wants to send a radio-gram but has no outgoing traffic in its queue.

- Message: Message payloads contain a message for another node, intended to be passed up the protocol stack by the intended recipient, and encrypted with its public key.

- Encapsulation: Encapsulation payloads contain another payload, and are intended to be retransmitted after decryption and re-padding.

Upon receipt of a radiogram, every node adds the transmitter's public key (obtained from the radiogram) to a table of current network members. Next, it attempts to decrypt the payload. Most of the time, the payload will not decrypt correctly because it is encrypted with some other node's public key or is a Cover payload. If the payload does decrypt correctly and it is of type Message, it passes it up the local protocol stack. If the decrypted payload is of type Encapsulation, it re-pads it out to the message size of the received radiogram and adds it to its outgoing traffic queue.

Finally, whether the message decrypted correctly or not, the node consults its local randomized transmission control function to determine whether it is time to send the next radiogram in its output queue of which every node maintains one. A node will transmit whenever either of two conditions has occurred:

– If a timeout interval has passed since the last time it has received any traffic.
– After it has received a radiogram (whether it decrypted correctly or not) and a local random function determines that it is time to transmit. This random function will return "true" with probability approximately $1/N$, where N is a current estimate of the total number of nodes currently in range. When this occurs, the node waits a fixed interval (based on the bandwidth to be consumed by the network) before transmitting.

If a node has no outgoing traffic in its queue but is scheduled to transmit a radiogram anyway (*e.g.*, when its timeout has expired because its random transmission function returned "true") it sends a COVER message. Otherwise, it removes a message from its outgoing queue and sends it.

To initiate an outgoing message to some group member, a node selects at random a set of $M - 1$ current group members. It then creates an (undirected) "onion" of M encrypted encapsulated payloads, with the inner most payload encrypted with the destinations public key, and adds the whole package to its outgoing message queue.

Note that this protocol is very inefficient on modern computers because it requires a public key decryption operation on every received radiogram by every node. The value of this protocol is twofold: first, it is simple enough to prove basic properties about it (as we informally sketch below) and, second, it is useful as a building block for more efficient protocols.

SECURITY ANALYSIS: It is not hard to prove that (assuming employed encryption is secure), this protocol achieves source and destination anonymity to observer, for *any* adversary access structure $\mathcal{C}$ (not restricted in size). Complete anonymity (even when the destination is corrupted) is achieved as long as an adversary has not corrupted all M nodes in the onion, which happens with probability roughly $\binom{N}{M}^{-1}$.

Anonymity per radiogram can also be proven, which implies that this protocol hides even the existence of communication. Moreover, even the sender of a radiogram in this protocol does not know who the receiver is, unless the sender is also the source who prepared the radiogram from scratch. This argument can be used to prove that an adaptive or pro-active adversary cannot gain much more information than a non-adaptive one in this protocol. Finally, as mentioned in Section 2, security against a Byzantine adversary can be achieved by adding authentication.

Note that (as discussed in Section 2), all these security guarantees are maintained for *any* scheduling of outgoing radiograms by the transmission control function, as long as it does not depend on the core traffic. Our choice of the randomized function above is designed to optimize round trip time.

3.2 Our Main Protocol

In this section, we describe our main protocol, which achieves the same anonymity guarantees as the protocol of Section 3.1, but with much better performance. The fundamental performance limitation of this protocol is that it requires a public-key operation (decryption) for each packet received, whether that packet was addressed to the node that received it or not. Ideally, such a public-key scheme would allow for efficient determination of whether the a message would decrypt correctly, without requiring a node to perform a full decryption of the message. Unfortunately, we are aware of no public key cryptosystems that have this property (and we believe motivates an interesting open cryptologic problem). Our main protocol uses the protocol of section 3.1 as a key distribution subprotocol for a more efficient hybrid protocol. In effect, our protocol simulates a public key cryptosystem in which non-recipients do not have to do expensive trial decryptions for data they will be unable to successfully decrypt.

Essentially, we run two protocols in parallel: the protocol of section 3.1 at low bandwidth (configured to occupy, *e.g.*, $\frac{1}{100}$ of the total channel computation or communication bandwidth), and a more computationally efficient version of the protocol that uses symmetric keys. Here, we refer to the part of the network running the low-bandwidth sub-protocol as the *key distribution subnet* and that part running the efficient symmetric sub-protocol as the *traffic subnet*.

When a new node joins the network, it must first join the key distribution subnet, using the same procedures and protocols as discussed in Section 3.1. Once part of the network, it identifies the other nodes in range and sends each of them via the key distribution protocol k unique (random) symmetric $(key, label)$ pairs (where the label is derived from a cryptographic hash of the key). It records each of these keys in a table, indexed by its label and the node to which it was sent. Observe that by using the public-key protocol, the receiver does not know which node originated the symmetric keys.

Upon receipt of a key distribution message containing $(key, label)$ pairs, the receiver records the keys and labels in a table, indexed by label. A node is free to delete both any keys it sent and any keys it received at any time from its tables; the number of keys stored is an arbitrary configuration parameter. In general, extra keys should be deleted FIFO, but will be used (see below) LIFO.

The traffic subnet uses these keys to encrypt and route bulk message traffic. To send a message, a node selects a routing as in the key distribution protocol, and successively encrypts the message. This time, how ever, it uses the symmetric keys it sent to the nodes for the encryption. Each encrypted layer is prefixed with the label of the key used to encrypt it. Once a key is used, it is deleted from the table.

Upon receipt of such a message, a node checks the key label of the outermost encryption layer against its table of received keys. If the key is not in the table, the message is discarded. If it is, the message is decrypted and processed as in the protocol of Section 3.1. New symmetric keys are periodically sent via the key distribution subnet.

Observe that the processing of bulk traffic messages is very efficient here; a node can immediately determine whether it will be able to decrypt a received message, and only symmetric cryptographic operations are required even by the intended receiver. Security analysis of this protocol follows directly from that of the previous section; details are omitted here for brevity's sake.

Note the frequency and method of key distribution has some security implications. The strongest anonymity properties (equivalent to the basic protocol) are obtained if only one symmetric key is included in each message on the key distribution subnet and each such key is used to send at most one message on the traffic subnet. However, this carries with it a performance penalty, since that means that every radiogram on the traffic subnet has to have at least one corresponding radiogram on the (less efficient) key distribution subnet.

When more than one key is included in a given key distribution subnet message or when the same key is used to encrypt more than one message on the traffic subnet, it becomes possible for the receiver to link messages from the same sender together. This may be acceptable in practice, however, especially when multiple messages are part of the same logical flow or if new keys are generated frequently enough that only a limited number of messages are linked together. Allowing the same key to be used throughout a flow can greatly improve the overall efficiency of the network by permitting the key distribution subnet to be run at lower bandwidth.

4 Related Work

Previous work has identified a larger security problem in mobile *ad hoc* networks (MANETs) than with conventional wired and wireless networks [9,10]. The main concerns are:

1. The use of wireless links makes certain types of attacks (eavesdropping, man-in-the-middle, denial of service, *etc.*) much easier in a MANET.
2. Security solutions that take advantage of static configurations, such as the restrictions in the network topology that are exploited by traditional firewalls, are not applicable in a mobile environment.

3. MANETs depend on the co-operation of all nodes for their correct, or at least efficient, operation. Misbehaving nodes are typically difficult to detect and contain. Some preliminary work in this area is described in [11,12]. [13] applies intrusion detection to the problem of misbehaving routers in a MANET, while [10] takes a preventive approach to the same problem.
4. The nature of nodes that typically participate on a MANET (low computation and bandwidth capabilities, limited power budget) expose them to new attacks (*e.g.*, power exhaustion through repeated packet retransmission [14] or increase their vulnerability to known attacks by making it difficult to adopt expensive mechanisms.

Traditional anonymizing approaches focus on hiding the client's identity, and only take the *traffic rate* into consideration insofar as it facilitates traffic analysis by an observer. However, the fundamental assumption behind most of these approaches were that an adversary did not (or could not) have global view of the network. While this may be realistic in a wide-area wired infrastructure, it is inappropriate in a local-area MANET, where the attacker can eavesdrop and triangulate on any communication. Traditional approaches against traffic analysis attacks [15,16,17,18,19,20] focus on individual links, based on the "wired infrastructure" assumption. The most basic anonymity solution is to interpose a proxy between two communicating parties. The usefulness of such an approach is limited to certain applications such as Web browsing under certain weak threat assumptions.

Chaum's MIXes [1] were an early proposal to construct an untraceable email system. The system was based on special-purpose nodes, called MIXes, which would perform the anonymization by re-ordering received messages and forwarding them through the MIX network. Under this approach, and subsequent work on various *remailers* that were put in service, an eavesdropper can only determine that a participant is communicating with the MIX. Traffic padding and message fragmentation would be needed to protect against adversaries that can monitor the entire MIX network, as may be the case with a wireless network. MIXes have been implemented to support many types of communication, for example electronic mail (*e.g.*, [21]), ISDN service [22], IP-layer infrastructure, and general synchronous communication (including Web browsing, see below).

ANON [23,24] is an IP-layer anonymizing infrastructure, allowing server addresses to be hidden from clients, and vice versa. It uses a set of network-resident nodes that act as anonymizing forwarders, similar to Chaum's MIXes. One basic assumption is that an attacker cannot monitor or subvert the whole ANON infrastructure itself, which is arguably realistic in certain scenarios (*e.g.*, in countering DoS attacks on servers) but inappropriate in a MANET. In [24] they introduce the concept of *on-demand link padding*, which adds padding traffic based on the bandwidth usage observed from real traffic. In theory, this allows the amount of the padding to be limited. However, this approach will still reveal the source or destination of many traffic flows (*e.g.*, a military HQ), since it is only done on a per-link basis.

Onion Routing [8,25,26] is an extension to MIXes that supports synchronous communication, such as Web browsing. It uses nested encrypted addresses, called an onion, constructed by the initiator of a connection. Each successive onion router peels off a layer and forwards the connection. To avoid public key operations on a per-packet basis, onion routers use per-connection symmetric secret keys. Link padding is mentioned as a mechanism for countering traffic analysis. A connectionless approach that is otherwise similar to Onion Routing is Non-Disclosure Method (NDM) [27].

Crowds [28] is a web-oriented peer-to-peer anonymizing infrastructure for synchronous communications. The main difference between Crowds and MIX-based solutions such as Onion Routing is that the routing path and the path length in Crowds are dynamic, versus static routing and preset path lengths in MIX networks. Analysis has shown that without active participation by the users of the anonymity system, there are attacks against anonymity that are more severe against Crowds than against MIXes [29,30]. At the same time, there are attacks that work better against MIX-based solutions than against Crowds [31]. There are properties of both of these types of attacks that are specific to Web browsing on personal computers, so they are not directly relevant to our solution.

Tarzan [32] is an IP-layer anonymizing network that uses a peer-to-peer infrastructure to hide the client's (source) identity. In addition to anonymity against casual eavesdroppers or a limited number of malicious participants, it also makes it impossible for an adversary that can observe the entire network to determine which participant initiates a particular message. However, it does not protect traffic sinks, or hide the amount of traffic generated by a particular source.

[33] examines the problem of selecting routes from among various MIXes in a wireless *ad hoc* network. [34] studies the use of per-flow *vs.* per-link cover traffic in a wireless *ad hoc* network, and concludes that the latter is less expensive in terms of required "dummy" packets, but requires encryption of each link. We extend their work for a broadcast medium with link-level encryption, which allows use of per-node constant-rate cover traffic, further reducing the amount of necessary dummy packets.

Agrawal and Kesdogan [35] propose using the number of observations of a user's actions that are needed to violate that user's anonymity (in the context of a particular protocol or application) as a metric for measuring the anonymity achievable through a particular privacy mechanism. They model attackers at different levels of omni-presence, ranging from those capable of observing all communications pertaining to a user, to attackers with only limited eavesdropping/monitoring capabilities.

[14] discusses denial of service attacks in 802.11-based wireless access networks. The authors show that the lack of basic security measures in the MAC layer allows attackers to completely disrupt service to the network using relatively few packets and low power consumption. They discuss several non-cryptographic countermeasures to these attacks, which address most of the problems.

5 Conclusions and Future Work

Anonymity has many potentially interesting applications in wireless networks, but conventional protocols do not work well in these environments. We have introduced a security model for wireless anonymity as well as a suite of protocols that provides basic anonymity functions in local broadcast networks.

A number of interesting and significant problems remain, however. Admission control and network management is perhaps the most significant area here: how do we control network membership, especially in ad-hoc public networks, and how can we best link such networks together? How do anonymity networks perform under, and how can they adapt to, highly dynamic and difficult radio conditions (especially where there are many, mostly disjoint, users with only a few links between them)? And, of course, issues of scale are likely to be especially difficult. We believe the model and analysis we presented in this paper will serve as a useful launching pad to answering these interesting questions.

References

1. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM (CACM) **24** (1981) 84–88
2. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks. In: Proc. of the 10th annu. ACM symp. on Principles of Distributed Computing. (2001) 51–59
3. Canetti, R., Herzberg, A.: Maintaining security in the presence of transient faults. In: Advances in Cryptology – CRYPTO '94. (1994) 425–438
4. Herzberg, A., Jakobsson, M., Jarecki, S., Krawczyk, H., Yung, M.: Proactive public key and signature systems. In: 1997 ACM Conference on Computers and Communication Security. (1997)
5. Beimel, A., Dolev, S.: Buses for anonymous message delivery. J. of Cryptology **1** (2003) 25–39
6. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Advances in Cryptology – EUROCRYPT 2001. (2001) 453–474
7. Canetti, R., Krawczyk, H.: Universally Composable Key Exchange and Secure Channels. In: Advances in Cryptology – EUROCRYPT 2002. (2002) 337–351
8. Reed, M., Syverson, P., Goldschlag, D.: Anonymous Connections and Onion Routing. IEEE Journal on Selected Areas in Communications (JSAC) **16** (1998) 482–494
9. Zhou, L., Haas, Z.J.: Securing Ad Hoc Networks. IEEE Networks **13** (1999)
10. Deng, H., Li, W., Agrawal, D.P.: Routing Security in Wireless Ad Hoc Networks. IEEE Communications **40** (2002) 70–75
11. Zhang, Y., Lee, W.: Intrusion Detection in Wireless Ad-Hoc Networks. In: Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom). (2000) 275–283
12. P. Albers *et al.*: Security in Ad Hoc Networks: A General Intrusion Detection Architecture Enhancing Trust Based Approaches. In: Proceedings of the 1st International Workshop on Wireless Information Systems, held in conjunction with the 4th International Conference on Enterprise Information Systems. (2002)
13. S. Marti *et al.*: Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In: Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom). (2000) 255–265

14. Bellardo, J., Savage, S.: 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In: Proceedings of the 12th USENIX Security Symposium. (2003) 15–28
15. Raymond, J.F.: Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In: Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability. (2001) 10–29
16. Venkatraman, B.R., Newman-Wolfe, R.E.: Transmission Schedules To Prevent Traffic Analysis. In: Proceedings of the 9th Annual Computer Security and Applications Conference (ACSAC). (1993)
17. Venkatraman, B.R., Newman-Wolfe, R.E.: High Level Prevention of Traffic Analysis. In: Proceedings of the 7th Annual Computer Security and Applications Conference (ACSAC). (1991)
18. Hajek, B., Radosavljevic, B.: Hiding Traffic Flow in Communication Networks. In: Proceedings of the IEEE Military Communication Conference (MilCom). (1992)
19. Guan, Y., Fu, X., Xuan, D., Shenoy, P., Bettati, R., Zhao, W.: Efficient Traffic Camouflaging in Mission-Critical QoS Guaranteed Networks. IEEE Transactions on Systems, Man, and Cybernetics **31** (2001)
20. Guan, Y., Li, C., Xuan, D., Bettati, R., Zhao, W.: Preventing Traffic Analysis for Real-Time Communication Networks. In: Proceedings of the IEEE Military Communication Conference (MilCom). (1999)
21. Gulcu, C., Tsudik, G.: Mixing E-mail with BABEL, Symposium on Network and Distributed System Security (1996) 2–16
22. Pfitzmann, A., Pfitzmann, B., Waidner, M.: Isdn-mixes: Untraceable communication with very small bandwidth overhead. In GI/ITG Conference: Communication in Distributed Systems (1991) 451–463
23. Kung, H.T., Bradner, S., Tan, K.S.: An IP-Layer Anonymizing Infrastructure. In: Proceedings of the IEEE Military Communication Conference (MilCom). (2002)
24. Kung, H.T., Cheng, CM., Tan, K.S., Bradner, S.: Design and Analysis of an IP-Layer Anonymizing Infrastructure. In: Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX). (2003) 62–75
25. Goldschlag, D., Reed, M., Syverson, P.: Onion routing for anonymous and private internet connections. Communications of the ACM (CACM) **42** (1999) 39–41
26. Syverson, P.F., Goldschlag, D.M., Reed, MG.: Anonymous connections and onion routing. In Proceedings of the 1997 IEEE Symposium on Security and Privacy (1997)
27. Fasbender, A., Kesdogan, D., Kubitz, O.: Variable and Scalable Security: Protection of Location Information in Mobile IP. In: Proceedings of the 46th IEEE Vehicular Technology Society Conference. (1996)
28. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Transactions on Information System Security **1** (1998)
29. Wright, M., Adler, M., Levine, B., Shields, C.: An analysis of the degradation of anonymity protocols. Proceedings of the Network and Distributed System Security Symposium (NDSS) (2002)
30. Wright, M., Adler, M., Levine, B., Shields, C.: Defending anonymous communications against passive logging attacks, In Proceedings of the 2003 IEEE Symposium on Security and Privacy (2003)
31. Felten, E., Schneider, M.: Timing attacks on web privacy. 7th ACM Conference on Computer and Communications Security (2000)
32. Freedman, M.J., Sit, E., Cates, J., Morris, R.: Introducing Tarzan, A Peer-to-Peer Anonymizing Network Layer. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems. (2002)

33. Jiang, S., Vaidya, N.H., Zhao, W.: Dynamic Mix Method in Wireless Ad Hoc Networks. In: Proceedings of the IEEE Military Communication Conference (MilCom). (2001)
34. Jiang, S., Vaidya, N.H., Zhao, W.: Routing in Packet Radio Networks to Prevent Traffic Analysis. In: Proceedings of the IEEE Information Assurance and Security Workshop. (2000)
35. Agrawal, D., Kesdogan, D.: Measuring Anonymity: The Disclosure Attack. IEEE Security & Privacy **1** (2003) 27–34

WAR: Wireless Anonymous Routing
(Transcript of Discussion)

Matt Blaze

AT&T Labs – Research, USA

This talk is called Wireless Anonymous Routing, and of course that gives a nice acronym for going to DoD funding agencies: we need to contrive an acronym for civilian agencies that means the same thing. What the authors have in common is that we are all current or future former employees of AT&T Labs. [Laughter] As we are not all current former employees at the time of this workshop, perhaps it is best to strike that from the transcript, but then the production schedule may surprise us[1].

Anonymity in networks is something that we have found applications for and technologies for, for many years. There are some techniques for doing it that in fact are quite practical, and we see in things like mix networks, onion routing, the re-mailer networks, systems like Crowds and so forth. Interestingly we have almost an equal number of diverse applications that benefit from anonymity; things like electronic voting, digital cash, electronic publishing (both in the sense of reading as well as writing), and less discussed but probably more prevalent are tactical and military applications, such as battlefield and sensor networks.

Now one of the things that all of these anonymity schemes and applications have in common is that they assume, and to some varied extent exploit, the fact that the underlying network is based on point-to-point links, and in particular, an important property that leads us to anonymity is that nobody has access to complete information about what's going on in the network. So we require the attacker to either monitor or subvert more nodes in the network than we believe is feasible for the attacker to do, and these assumptions really underly, at some point, just about the entire landscape of what we consider to be practical anonymity schemes. And it's fortunate of course that things like the Internet seem to actually have this property, you know, we have data networks that are much larger than anyone can monitor, and so if we can take those properties and turn them in on themselves, we can get these very nice anonymous networks.

In research what we want to do is figure out what the assumptions are, and I think one can quite usefully do this here. First let's just think for a second about how current anonymity schemes work. First of all the nodes form a network of some kind, and you know, depending on the details of the scheme, that network may be ad hoc, and it may not be, and it may be at the network layer, or may be at some higher layer of abstraction specific to the application. Once we have this network (however we may have formed it), we want to route messages through some chain of nodes in the network (some of them may be trusted, some of

[1] Indeed it did!

them may be untrusted), until they reach their destination. Now there are lots of different variables here: whether or not you use encryption, whether or not you're concerned about eavesdroppers or defectors in the network, whether or not they're introducing delays to make the traffic analysis infeasible even for somebody who does have a global view of the traffic, whether you're hiding the identity of the sender from the recipient, whether or not it's source routed, or whether the network itself with individual nodes in the network can contribute to the routing around this network. These are all different parameters of the same basic concept, which is that we're exploiting and amplifying the point-to-point nature of the underlying network, or simulating a point-to-point network in order to make it difficult to figure out what's going on, and who is communicating with who, and what they're actually saying.

So for research purposes let's break these assumptions, let's get rid of the point-to-point anonymity property and see what happens. In particular, let's look at a very interesting type of underlying network topology based on wireless links in which a wireless link (I'm kind of over-abstracting it), instead of being based on point-to-point networking, allows everybody to see everything that's being transmitted. We'll be more generous to the attacker in our model, and assume that transmitting immediately reveals your identity, your location, and some linkage between your past behaviour and your current behaviour to everyone else, so that you're instantly direction-found, you're triangulated upon as soon as you transmit. But there are underlying wireless schemes that tend to break this with their modulation schemes and so on, things like spread spectrum, that try to make it difficult for outsiders to do this, but they only make it difficult for *outsiders* to do this; if you know the spreading code, if you know the secret used to communicate, then you can figure out who transmitted.

We also have in wireless networks the property that the bandwidth is a pool among local nodes, so if you transmit you're taking away the ability for other nodes in the network to use the bandwidth at that particular time. These are the essential features of point-to-point networking that are absent in a wireless network for the purposes of building an anonymity scheme.

Birgit Pfitzmann: What you're saying here is that somebody can see inside everything transmitted, not that he can subvert all the nodes. The idea of mixes is that someone cannot see the entire traffic unless you also subvert all the nodes in an exchange, so I don't really see the basic principles here yet.

Reply: Well what a mix does is it essentially implements a point-to-point network which goes through a large number of nodes that you have to subvert. Lets assume that knowing the traffic going in and out of a large fraction of nodes in a network, ultimately might let me derive the source and the sink of the traffic, it might let me subvert the node that's immediately after the source and immediately before the sink. You might be able to learn different things, again, depending on the parameters of the mix network.

Birgit Pfitzmann: There's lots of other things, but if the mix network is done well, then you can't see all the traffic: that was the main purpose of the mix, if it's not just a re-mailer but really a mix.

Reply: Well in a mix network you try and be very robust against the subversion of a large number of nodes and against somebody who can see lots of the traffic in the network, so joining a mix network as a node is a somewhat heavyweight operation, your neighbours have to be known to you in some sense, you have to know who you sent the message to, and so on. In a wireless network, that underlying assumption may be broken.

Virgil Gligor: I think that in a wireless network you have no single point where you can see everything, because you can only hear broadcasts in your neighbourhood.

Reply: Well within a neighbourhood you have pretty global information.

Ross Anderson: No, it's even stronger than that, because in sensor networks that are being built now everybody uses the same frequency. There's one radio frequency for all million nodes in the network, so no single device, no matter where located, could listen in to more than fifty or so nodes.

Reply: I don't want to make that assumption, I want to assume the attacker has global information; it's probably a reasonable assumption to make in a wireless network, in particular I might have a defector in each neighbourhood.

John Ioannidis: A defector can give a very good sample of what you're seeing, so it is a reasonable assumption.

Reply: So my argument here is that the fact we have ignored the wireless environment in developing anonymity systems is a bit of a shame, because that's really where a lot of very interesting applications come. You know, we talk about battlefield and sensor networks, but also mobile ad hoc networks generally are ones in which there is probably demand for anonymity. We've heard concerns about the anonymity issues of things like these RF sensor tags on clothing, we've heard about advertising systems that are based on a mobile phone providing information that is local to the immediate area, but the consumer may not want to provide the marketing information to every company that they happen to have passed by (even if they're interested in that information while they're local). So aside from just the military applications, there do seem to be mobile network schemes that quite naturally require anonymity, at least from somebody's perspective, and the existing schemes that we have make it somewhat difficult for us to do that.

This is pretty new ground from the point of view of actually building practical systems. In particular, things that are efficient for the point-to-point environment are likely to be inefficient in the shared bandwidth wireless environment. If you have a link that's public, everyone may be able to find the sender, but there are other things that we might be able to exploit in a wireless environment: the identity of the sender may be clear, but the identity of the receiver need not be, unless you're assuming technology that can detect behaviour of individual receivers while they're actually receiving. When a signal is put out onto the air we don't know who it is directed at, and the sender may not, or certainly need not, know who the intended recipients are. So the bottom line is that this inversion of assumptions makes anonymity a great research topic because we can build systems and adapt systems and see how they behave.

So we have a suite of protocols called WAR for wireless anonymous routing, and the basic architecture that we're assuming is a wireless broadcast channel, in which everyone can hear everything, the sender is assumed to be direction findable and instantly triangulatable. Everyone in the network has the ability to do public key cryptography, and broadcasts their public key periodically so that other nodes in the network can know how to speak with them in secrecy, or at least with message secrecy. And there is some protocol for joining or leaving the network. Now what a protocol is depends on the circumstances, it may be an ad hoc protocol that allows anybody who claims to want to enter the network to enter it, or a scheme which distinguishes between those that are trusted and those that are not.

Frank Stajano: How does broadcasting one's public key reconcile with anonymity?

Reply: Well the public key identifies you as a node in the network not as a message sink.

These are public keys for the purpose of routing, not for identifying with whom you're ultimately communicating. You need some way of establishing an end to end secret channel on top of this, but that problem is easily solved in standard ways.

Frank Stajano: So what does broadcasting the key buy you?

Reply: It tells you who the other nodes in the network are. Let's use a forward reference for a moment: to send a message in this scheme, you create (for lack of a better term) an onion, a layered, encrypted message, encrypted using the public keys of some members of the network. The depth of the onion would be a security parameter, and how we choose the security parameter is an equally interesting question. Anyway, we broadcast this multiply encrypted message out to everyone. Now obviously we're not going to use public key cryptography for the whole message, there'll probably be a session key, and so forth. But for purposes of abstraction we're creating an encrypted wrapper with the public keys of the various nodes we want to route through. You broadcast this message out on the channel, and on receipt of the message everybody attempts to decrypt every received message. Now it could decrypt successfully, and we're assuming some redundancy that lets you figure out whether or not it decrypts successfully. In this case, the outer layer of the onion was encrypted using the public key that you have the private key for, and now you look on the inside and the decide whether or not this message is in fact intended for you, or whether it contains additional cipher-text. If it's for you, just forward it up your protocol stack, and if it's for someone else, add it to your output message queue.

Now, any time a message is received, every node in the network flips a coin that has the number of sides equivalent to its estimate of the number of nodes in the network (with some keep-alive scheme in case the network suddenly becomes very quiet). If it comes up heads, in other words, for one out of every N messages received, you now decide that it's your turn to transmit, and send the next message from your output queue. If your output queue is empty just send a message of random bytes.

Ross Anderson: What do you gain from using onions in a network like this, with a broadcast channel? Frank and I[2] already talked about this: in the context of cocaine auctions, if everybody is present, it's not at all obvious that you need any indirection because whoever a message is for has seen it and decrypted it. And if his key is compromised then he's screwed anyway so the onion doesn't buy you anything.

Reply: I'll answer that with a forward reference, but you may be right.

George Danezis: There is another issue: if everybody knows everybody else, so you know everybody else's keys, you don't need any asymmetric cryptography in the network at all because you have session keys with everybody else, so basically . . .

Reply: No, that's a forward reference too.

Let's think about the ad hoc environment for a second because this leads to an interesting cryptographic problem. If we describe this as simple public key cryptography, say RSA, then everyone has to do a trial decrypt on every message sent on the broadcast channel, but only a fraction of these messages are going decrypt successfully. So there's an enormous amount of extra computation going on, and if you imagine that the messages are rather small, and that there are a lot of different keys being used at any given time, this is going to greatly limit the bandwidth available to the channel. So what we'd like is a public key primitive that we don't actually seem to have with existing cryptographic schemes, such that it is easy to determine without going through the computational effort of decryption whether or not a message was encrypted using the public key for which you know the private key. At the same time it should be hard to determine, knowing only someone else's public key, whether or not the message was encrypted with it (unless you were in fact the sender). Existing public key schemes do not in any obvious way have this property, and so the invention of a public key primitive that has this would probably be a contribution; it's not a contribution we have made, I think it's likely to be a hard one, although we might be able to simulate it.

It's easy to look at this and see that this is secure, but it's computationally extensive, as we've seen, and it's also bandwidth intensive, because of this multiple layers of encryption which, as Ross points out, may not in fact be necessary. However, the advantage of this network is that it's easy to reduce this to a mix network: you can analyse it precisely the way that we analyse a mix if we're careful. There's still a lot of details to manage; how do you manage membership

[2] R. Anderson, F. Stajano, "The Cocaine Auction Protocol", Information Hiding, LNCS 1768, 434–447, 1999.

in the network, how do you deal with lost messages, how do you bootstrap and get started, and so on.

A more efficient version of the protocol might in fact be quite practical. One can imagine a network in which we use the protocol as initially described, but only as a small fraction of the available bandwidth of the channel. The purpose of that protocol is strictly for key management, and not for message encryption, so we might only allow one message every few seconds to be transmitted on this network overall, and the fact that this is locally inefficient in terms of bandwidth and computation doesn't really matter to us because it's only consuming a small amount of the channel's bandwidth. We use that protocol to exchange symmetric keys with everyone else in the network. Essentially when it's our turn to transmit using this low bandwidth network, we figure out who else is on the network and we send to everyone else in the network a symmetric key that we keep track of. When they receive one of these keys they keep track of what it is for some period of time. Now when I want to create an onion and transfer a message in this kind of network, instead of using public key cryptography, I just select a collection of these symmetric keys and use them. On receipt I can try decryption with every symmetric key that I have received, and I need a very large network indeed before the costs of these symmetric decryptions approaches the cost of doing even single public key decryption, so we can take it up from straightforward optimisations there.

So here we have a heavy-weight protocol for joining the network and a light-weight protocol for communication; we allow each node in the network to have two identities, essentially one for its purpose as a source or sink of messages, and another as a routing node in the network.

Some other variants we can do. One is that, rather than exchanging symmetric keys, what I might exchange with the other nodes in the network is simply a short term nonce, with which I will prefix any message intended for them encrypted using public key techniques. Now when I receive a message I look up its prefix in a table of all nonces I have received, and if it matches then I know it might be worth doing the decryption. Another possibility is to use el Gamal to establish session keys that create essentially a unique and predefined key with every other node in the network. It doesn't provide originator anonymity with the recipient, but that may not matter. It might let the intermediate node relay and let the message bounce around until hopefully it reaches its intended destination.

So we think this is the beginning of a rich area of research here. We have a suite of basic protocols and some proof-of-concept implementation. I think the interesting questions are in the details: how do you actually build practical ad hoc anonymous routed wireless networks, the network management issues are interesting, the interference of malicious participant issues are interesting, denial of service, how do you deal with real radio conditions in which you may have overlapping or non-overlapping clusters of wireless nodes, how do they discover one another, how do you figure out which one you're in and which one your neighbours are in, and what kind of interesting applications come out of this that might address some of the privacy concerns that commercial wireless networks bring up, and some of the tactical and strategic issues that military wireless networks bring up.

Birgit Pfitzmann: The scheme with the labels that you exchange beforehand; Andreas Pfitzmann has it in one of these papers in the '80s, but there it was essentially just for broadcasting without any mixes.

Reply: Yes, we're using a variant on his scheme. It didn't precisely work in this scenario, so it's not exactly the same, but it's similar.

Richard Clayton: I'm not entirely sure you'd be able to pick up much of the mix stuff and move it into this, because a lot of the attacks on mixes are essentially sequential; if you run an intersection attack on one mix you can see the packet came in and came out again, and you can track it, but since you've got no idea who listens to it after that, it is not at all clear that an attacker will follow the traffic through the network. Therefore a lot of the defences being built into modern mixes to tackle all that really aren't relevant.

Reply: The advantage, though, of being able to reduce something to a mix is in the other direction: you can assure yourself it is at least as secure.

George Danezis: You talk about tactical command and control. The main problem is the sender is not hidden; even if the relationship with other senders is not known they still have very clear patterns showing what they are, so a radar node will be completely different from a meteorological node or a command and control centre node. The properties you are looking for are not just relationship between senders and receivers, but also node visibility.

Reply: Well it's hard to hide who the nodes are, but their identities should be hidden by their transmission discipline which says everybody's transmitting a protocol pattern that's unrelated to the messages.

George Danezis: That's hard.

N. Asokan: When deciding whether to use an onion or not, if one consideration is power consumption, you don't want to transmit more than you have to.

Reply: The obvious problem here is that with a straightforward protocol every node is automatically consuming an equal fraction of bandwidth regardless of what it actually needs to transmit, and that may or may not be practical. Military applications very often want to really, really optimise power consumption: a statistic I've learnt recently is that aside from body weight, the majority of weight carried by a soldier is batteries.

Ross Anderson: I've been thinking a bit about the difference between this non-anonymous broadcast and the anonymous broadcast that Frank and I played around with a couple of years ago. It strikes me that one of the main differences is that in a truly anonymous broadcast network, if you have a relatively busy node then it can replicate itself, and this means that you can use bandwidth more efficiently. This is because you're getting the inverse of multiplexing as it were, whereas with your assumption that the nodes can all be tracked, you have to exercise transmission discipline of a different type, which is more expensive in power consumption. So there may be a cost, there may be an argument where battery weight is everything; that you have a secret node and a top secret node, and only the top secret node assumes that the enemy has got immediate real time DX on it.

Reply: Right, or we might use this hybrid model; use the very secure protocol for your key management, and use a less secure protocol for your actual message traffic, depending on the contents of the message you send.

Ross Anderson: I can imagine that in a close combat situation the traffic analysis that you would be worried about would be the other side trying to look for those of your shooters who are receiving large quantities of image data, because these are the guys that you want to take out first, and they'd be the guys who could be acknowledging the largest number of possible packets. So there might be quite a lot of room for messing around with protocols to obscure that kind of thing for that specific application.

Reply: Yes, or hiding of command structures.

Bruce Christianson: Your scheme could provide us with resilience properties which allow a message to be delivered even if nodes move in and out of range by sending the same message down two different routes to its destination.

Reply: Yes, we don't find out until very late when a message has been dropped in this scheme.

Bruce Christianson: Yes, that's right, and that has clear implications.

Limitations of IPsec Policy Mechanisms

Jari Arkko and Pekka Nikander

Ericsson Research NomadicLab,
02420 Jorvas, Finland
{Jari.Arkko, Pekka.Nikander}@nomadiclab.com

Abstract. IPsec, while widely implemented, is rarely used for end-to-end protection of application protocols. Instead, it is mainly used today as an "all or nothing" protection for VPNs. In this paper we discuss the structure and shortcomings of the IPsec security policy mechanisms as partial reasons for this situation. We describe our experiences in using IPsec in a number of situations, including IPv6 control protocols, mobility protocols, network management, and multimedia protocols. We conclude that more often than not, the existing policy mechanisms are inadequate. While IPsec is quite effective in authenticating the peer and establishing assurance about its identity, the lack of attention to authorization questions is a root cause of the existing inadequacies. We also claim that the problems are more fundamental than the lack of suitable APIs and management tools. Finally, we present some potential architectural modifications which could improve the situation, and discuss the practical challenges in achieving these modifications.

1 Introduction

Today, all the major operating systems and a growing number of devices include a standards compliant IPsec [16] implementation. This is occurring later than anticipated – perhaps due to the slow standardization process, or the widespread availability of the SSH [26] and TLS [10] protocols [13]. But the current wide support is significant, since it would allow IPsec to be used as a universal security solution. However, IPsec is today typically used only as an "all or nothing" protection for VPNs, connecting, for instance, two sites privately together or remote users to a central site.

Today, IPsec is rarely used for end-to-end protection of application protocols. It is also rarely used for protecting the various control protocols of the Internet. This is unfortunate, as the original goal of IPsec was to enable the protection of all types of communications between Internet nodes. It is even required that all IPv6 nodes implement IPsec. At the same time, there are a number of situations where there are no credible alternatives to the use of IPsec. For example, TLS and SSH are limited to TCP- and SCTP-based protocols. Furthermore, IPsec offers certain security benefits over the alternatives, such as better resistance to Denial-of-Service attacks than TLS.

In the IETF, all new protocol specifications have to describe how the protocols are to be secured. Often there is a mandatory requirement that all implementations must support a particular security solution [23]. However, many of the older specifications suggest using IPsec without providing sufficient information on how exactly to use IPsec. Many of the newer protocols rely on TLS. Some of newer protocols require IPsec, but the details are often relatively complicated in practice [1,7]. More importantly, there is lack of users that actually turn *on* the specified and implemented security mechanisms, even when they are there and readily available. In some cases we have even found evidence that the originally intended secure behavior fails in unexpected ways and has significant, undocumented limitations.

The purpose of this paper is to share our experiences on the reasons that have led to the rare of use IPsec outside of the VPN applications. One of our conclusions is that quite often the existing IPsec policy mechanisms turn out to be inadequate to the tasks at hand. While IPsec is quite effective in authenticating the peers and establishing assurance about the *identity* of the peers, we claim that the lack of attention to *authorization* questions is a root cause for many of the existing inadequacies. We also think that the problems are more fundamental than the lack of suitable APIs, a common complaint against IPsec [8,13].

The rest of this paper is structured as follows. In Section 2 we illustrate our experiences in using IPsec in a number of situations, including IPv6 control protocols, mobility protocols, network management, and multimedia protocols. In Section 3 we analyze these results and categorize the observed limitations. Finally, in Section 4 we present some potential architectural modifications which could improve the situation, and discuss the practical challenges in achieving these modifications.

2 Practical Experiences

2.1 IPv6 Neighbor Discovery

The Neighbor Discovery (ND) protocol is an integral part of IPv6. The protocol is defined in RFCs 2461 and 2462 [21,25], implemented in all IPv6 devices. Neighbor Discovery is used by IPv6 nodes to discover other nodes on the same local link, to determine the link-layer addresses of these nodes, to find routers, and to maintain reachability information about the active neighbors. The specifications call for the use of IPsec to protect the ND messages, though with the details and limitations have been largely left unspecified.

In this particular application IPsec can only be used with manual keying. The currently standardized key management protocols, i.e. IKE, cannot be used, since ND uses multicast, and multicast is not supported by IKE.

More fundamentally, a chicken-and-egg problem [2] appears when attempting to use IKE [12] before ND is operational. The problem is easy to explain through an example. Let us assume that Alice wants to communicate with Bob over the local link. Since all she initially has is Bob's IP address, she must first find Bob's link-layer address. To do so, she must run the Neighbor Discovery protocol. Now, if all traffic between Alice and Bob is expected to be secured with IPsec, this would imply that even the messages used for finding Bob's link-layer address would have to be secured with IPsec. In order to secure these messages, a pair of IPsec security association between Alice and Bob must either exist or needs to be established. Since we are assuming that manual keying is not used, such security associations must be created. This requires IP packets to be sent, such as the IKE UDP packets. However, in order to send such packets, the link-layer address of Bob would have to be known to Alice. Hence, a chicken-and-egg problem exists.

Even if manual keying can be used with Neighbor Discovery, the number of security associations needed is quite large [4]. The number of security associations on *each* node is proportional to the number of IP addresses assigned to *all* nodes attached to the link. This is impractical. Even if keys shared between all members of a group were acceptable, using manual keying becomes impossible on links where the IP addresses – which are decided by the nodes themselves in IPv6 – are not known beforehand.

More importantly, the use symmetric security associations does not prevent authenticated hosts from masquerading as routers for other hosts, given that the same key is known to all nodes when multicast is used. Similarly, a host can forge messages that appear to come from another host's address.

Finally, the lack of detailed instructions in RFC 2461 on how to set up the necessary security associations and policy entries creates a burden for administrators, and may limit interoperability.

As a result, IPsec is rarely, if ever, used for protecting Neighbor Discovery messages.

Similar problems about lack of IP addresses in the initial stages make the use of IPsec hard with DHCP [11]. Multicast problems appear also in other applications that use multicast or broadcast, such as routing protocols.

2.2 Secure IPv6 Neighbor Discovery

The problems in protecting Neighbor Discovery became apparent a few years ago. More recently, new proposals have appeared for solving some of these problems [5,6,18]. Even if this is still ongoing work, we discuss some of the experiences gained in these attempts to extend IPsec to make Neighbor Discovery protection easier.

The approach chosen in [6] involves the definition of a new IPsec transform, based on asymmetric cryptography. The new transform is designed to be compatible with multicast, avoid the chicken-and-egg problems, authorize routers and hosts separately, and be capable of telling which addresses belong to which hosts. IPsec AH [15], adopted to use public key cryptography, is used to protect the relevant Neighbor Discovery messages. However, a few problems still remain. A minor problem is that the IPsec policy mechanisms do not allow selection based on ICMP message types. Another issue is that some of the Neighbor Discovery solicitations are sent with an unspecified source address, and the "real" address appears only at the application layer. This prevents IPsec from ensuring that the host has used the right address in the message. These problems can be solved, by extending the IPsec policy mechanisms and by modifying the source address rules of the Neighbor Discovery protocol.

But there are additional problems that cannot be solved easily. The most interesting problem relates to the gradual deployment of secure Neighbor Discovery on links that have both secure and non-secure nodes. While it is possible to arrange this with IPsec, it would come at the cost of using different addresses for the two groups of nodes. For instance, each node would have to have two IPv6 link-local address, one secure and one insecure, and the address prefixes announced by the routers would have to be duplicated in order to make it possible to use both secure and insecure nodes at the same link.

In contrast, a security mechanism integrated to the Neighbor Discovery protocol can be made optional, allowing easy co-existence with older equipment [3]. The secure Neighbor Discovery implementation has immediate knowledge about the security status of any received message and can make specific checks with regards to the acceptance of the message. For instance, a securely communicated router advertisement would override any previously received insecure advertisement, but not vice versa. This level of control is hard to achieve with IPsec.

In addition, given the stateless nature of IPsec (in an environment that does not have a separate key management phase), replay protection can be easily arranged only through timestamps. In an solution integrated with Neighbor Discovery, request - response pairs can be easily matched using a nonce. Since the use of nonces is desirable, an IPsec-based security solution would need components both from the Neighbor Discovery and IPsec layers, thereby complicating the analysis of the mechanisms.

Given these issues, the IETF recently decided to abandon IPsec, and design the future Neighbor Discovery security solution in such a way that it is integrated with the actual protocol.

2.3 IP Mobility

An initial approach to secure Mobile IPv6 route optimization relied on IPsec and its key management. It was suggested to use IPsec to protect the so called Binding Updates, which are messages sent by the mobile nodes to their peer nodes in the Internet. While thas approach would work well on a small scale – for instance, between a user's laptop and her home server – it fails on the global scale.

An obvious reason is that we do not have (and most probably will never have) a global authentication infrastructure which could help the two hosts to set up security associations between them. More interestingly, even if such an infrastructure would exist, identity based authentication of Binding Updates would be almost worthless. Simple identity based authentication would verify that a claimed identity, e.g., an e-mail address or a Distinguished Name, corresponds to the used public keys. Binding Updates, however, operate on IP addresses. Thus, unless the used authentication infrastructure were to track all the addresses assigned to users (even with dynamic assignments) and to provide this information in a secure way, the authentication information would not help in authorizing operations on IP addresses.

As a result, it was necessary to develop new security mechanisms for Mobile IPv6 Route Optimization. The new mechanisms rely on the routing infrastructure to provide some level of the authorization of IP addresses [14].

Interestingly, IPsec is still used in Mobile IPv6 to protect the signaling between the mobile nodes and their home agents. In a way, protecting these message is an easier problem, since there must be an a priori relationship between a mobile node and its home agent. However, the current specifications create a hard binding between static IP addresses, security associations, and IKE endpoints [7]. This creates some undesirable complexity and limitations to the protocol. For instance, current protocols do not easily allow dynamic allocation of home addresses, as security policies are tied to the addresses.

2.4 Network Management Protocols

Interactions and conflicts between the application layer and the IPsec layer are also problematic for using IPsec as a form of protection for network management protocols. IPsec can easily provide a base-level security for all management traffic in a large network. However, typically there is a requirement to differentiate management users from each other. For instance, one user may be allowed to only view data (e.g. SNMP GET) while another would also be allowed to make modifications. A third user might only be allowed to change particular variables.

It is hard to model this in the IPsec security policy database, as the policies are not expressive enough to be able to understand which operations are being performed at the application layer. At the same time, the lack of standardized or deployable APIs between the IPsec and application layer prevents the application from learning the identity of the user from IPsec. As a result, either the offered security level resembles "all nodes are equal" VPNs, or other security solutions must be adopted.

If the IPsec implementation in a management station supports only machine level authentication and not user authentication, it can be difficult to distinguish two users with different privileges when that station communicates with a managed node. Similarly, problems arise if the implementation supports user authentication but does not ensure that only the traffic from this particular user is protected by security associations created for that user – achieving this in a kernel mode implementation may be hard. For these reasons, some protocols provide machine authentication at the IPsec level, but add their own user authentication mechanisms inside the application layer. Examples of such protocols include iSCSI [1] and L2TP [19].

2.5 Streaming Multimedia

Currently the selection of outgoing IPsec policy entries is based on the examination of IP addresses, the upper layer protocol identifier, and port numbers. This approach works well with protocols that use stable IP addresses and meaningful port numbers. Not all applications are like that. For instance, the streaming multimedia applications utilizing UDP and RTP [24] are quite problematic from this perspective. Both the source and destination port numbers used for RTP can be dynamically assigned, making it hard to define IPsec policies to select the appropriate UDP streams to protect.

3 Analysis of the Limitations

The limitations we have experienced fall under the following categories:

– Lack of expressive power in policy specifications.
– Lack of application control over policies.
– Lack of support for authorization, and the inability to link authorization decisions with security processing.

In the following, we discuss each of these issues separately.

3.1 Expressive Power

In some cases the expressive power of the security policy entries needs to be increased to cover more than the pair of addresses, the upper layer protocol identifier, and the ports. For instance, recent new requirements include SCTP addresses and port numbers, or ICMP message types.

More generally, IP addresses no longer carry semantic information to the extent they used to do. Given the widespread use of dynamic address assignment, nomadic users, and the growing use of mobility, privacy addressing [20], and multihoming, an IP address does no more identify the host. Yet, the IPsec policies and security association parameters are tightly bound with IP addresses. This limits the freedom nodes have in choosing the addresses they use for communications.

Furthermore, there is no support in the current (or planned) IPsec key management protocols for controlled address agility. For instance, it would be useful to change the address associated with a particular IKE phase 1 security association, or be able to use multiple addresses instead of just one.

Finally, decisions cannot always be made based on port numbers, given that not all protocols use static port numbers. Furthermore, port numbers are not universally available in security gateways when fragmentation occurs.

As a result of the above, there are scenarios where *none* of the available selectors can be used, for instance when both dynamic addresses and dynamic port numbers are used.

3.2 Application Control

Application control of policies is required in two cases: First, it may be necessary to use of IPsec in cases where there are no distinguishable selectors. Consider, for instance, the dynamic address and port number case discussed above.

Secondly, policies need to be configured by applications. As we have found out, significant protocols can become widely deployed without security ever being turned on. A partial reason for this is that users find it unreasonable that they have to configure IPsec policy entries, and that not all networks have professional network managers whose task is to ensure correct policies.

On the other hand, the end-host applications could set up the necessary entries since in many cases the expected policies are standardized. Some standard specifications even require that the application is aware of the underlying security mechanisms, or at least whether security is turned on or off.

3.3 Authorization

Not all peers have the same rights. For instance, the administrator of a host, connecting from a remote device, has different rights than an unknown peer connecting through opportunistic IPsec. Even peers whose identities are authenticated within a specific trust domain may still have different rights. For example, in a corporate network, network managers have different rights than employees.

In order to use certificates for authorization, it is typically necessary to either use attribute certificates, create a separate certificate infrastructure for each different application, or introduce standardized extensions to the certificate formats to be able to express the authorization that the certificate represents. The traditional alternative, using local access control lists is not a viable option in a large network.

Finally, there are no standard requirements for IKE to take in account the authorization information from certificates (phase 1) in accepting a specific request for new security associations (phase 2). And, taking this account can be problematic for the responder as it may need to use a specific identity in phase 1, but does not know for what purpose the security associations will be requested in phase 2.

More importantly, the security processing decisions often require information from the application layer, or vice versa. In the current IPsec architecture all security processing is expected to be performed at the IPsec layer, and the application typically does not even know if IPsec was applied. This makes it hard to accommodate finer grained decisions, for example, such as those required in co-existence of secure and non-secure Neighbor Discovery.

4 Potential Improvements

There appears to be two possible paths for the future development of security solutions in the context of protocols discussed in this paper. The first path involves focusing IPsec as a protocol solely for VPNs, and using other solutions for securing application protocols. The second path involves an attempt to make IPsec friendlier for application protocols.

In the following we discuss the required improvements for both of these approaches.

4.1 Application-Specific Security

This appears to be the current default assumption for most protocol developers. An advantage of this approach is that no additional requirements are placed for current IPsec implementations, and that tailored security solutions can be provided for the application protocols.

A disadvantage is that a tailored security solution is always needed, at least for those protocols that cannot employ TLS or S/MIME. This approach seems also at odds with the current IPv6 requirement of mandating IPsec for every node. (For instance, a requirement to support TLS for applications and secure Neighbor Discovery for control signaling might provide a better security / cost ratio.)

If this approach is chosen, it would be desirable to have a common security solution for those protocols that cannot employ TLS or S/MIME. It may not be possible to create a separate protocol or a separate "layer" due to the very different nature of the applications. However, it may be possible to provide tools in the form of common security object formats and a library to support them. The library should allow application programmers to integrate the following features to their protocols:

– Liveness, denial-of-service, and address validity testing procedures.
– Retrieval and processing of certificates and certificate chains.
– Passing of signed and encrypted data objects.
– Passing information about the peer to the application, and responding to authorization questions from the application.
– Cryptographically generated address (CGA) processing [22].

While some such libraries exist today, they are typically tailored for traditional applications, and lack mechanisms (such as address tests) required in control protocols.

4.2 Enhanced IPsec

The following improvements appear necessary for the wider use of IPsec in securing applications:

1. Provide a mechanism for applications to control policies. We recommend that specific applications automatically provide at least a default configuration for IPsec. This can be provided through an API. That would help ensure that security is not accidentally turned off, improve interoperability, and it might help in dealing with the port number agility problem. The drawback of the approach is an additional burden on the applications. However, there is a trade-off between a one-time effort in software construction and manual configuration effort by all users, making the application based solution beneficial in the longer run.

   Unfortunately, such default configurations may not always suffice. For instance, the default configuration may not be compatible with specific requirements for protecting all traffic from the node in question.

2. Allow applications to make authorization decisions. One approach to allow applications to control authorization is to create an application programming interface (API) between IPsec, IKE, and applications. The creation of such an interface is particularly easy when the “applications” are integral parts of the IP stack, such as in IPv6 control signaling. A standardized API would also make it possible for user space applications to rely on IPsec and IKE to provide meaningful security information to them.

   However, it is not enough to copy the functionality of the IPsec security policy database to the application layer; the application layer policy information has to be presented in “semantic” form. For instance, policies could be based on application concepts rather than port numbers.

   In order to make use of the API, applications need tools to deal with authorization questions. For instance, an application may have a need to perform certificate chain validation to a particular trusted root. Such needs can be satisfied either through the API or via a specific library, such as Keynote2 [9]. The effective use of certificates for authorization purposes requires certificate mechanisms extensions so that it is possible to represent authorization information in them in an easy manner.
3. Reduce the reliance on IP addresses in the policies. This reduction applies in IPsec security associations, in policy entries, and in application layer policies. One approach to go to that direction is presently available in the form of the Host Identity Protocol (HIP) [17] proposal. Reducing reliance on IP addresses is necessary to allow for address agility.

   While the enhancement of the expressive power of IPsec policies appears necessary, it seems hard to decide exactly how much more expressive the policies should be, and from which point applications and interfaces to applications should be used instead. Currently, our opinion is that the policies should be extended to cover new transport protocols, ICMP message types, and address agility.

5 Conclusions

We have described our experiences in attempting to use IPsec to secure several types of protocols. Based on these experiences, we recommend either dedicating IPsec for pure VPN usage, or extending IPsec with APIs that can provide the required level of information and control for the applications.

As usual, the biggest challenge of the IPsec enhancements seems to be deployment. Implementing the suggested changes does not seem to have any larger technical or theoretical obstacles, though they can only be provided in native host implementations and not with Bumb-in-the-Wire implementations. While distributing the new implementations will be time consuming, the main problems are in the applications.

Handling authorization properly seems to be especially challenging. It requires attention both from security and application semantics. Integrating IPsec with generic authorization mechanisms, such as Keynote2 [9], and providing an

API for applications so that they can query IPsec for both identification and authorization information, is only the beginning. The applications must also be changed so that they actually use this information in some meaningful way.

Our experience suggests that there is an apparent dichotomy between the security experts and application developers, at least at the IETF. Creating proper communication between these two groups of people seems to be really hard.

Most application developers have been trained to think about security in terms of users and user rights, i.e., access control lists. However, many future applications would probably benefit from different authorization models, such as relying on direct certificate based authorization instead of relying on locally configured username based access control lists. Application developers often also tend to think of security as an authentication and encryption mechanism, and fail to use their application expertise to develop requirements for the security solution. Thus, application developers need new mental models, and that will take time.

Many of the security experts are unwilling to consider any compromises to security, and may not perform a proper risk analysis. This, in turn, has led some applications developers to shun security.

Willingness to think in new ways is required both from application developers and from security experts. The application developers (and even some security experts) need to learn to think in new terms, paying much more attention to authorization than before. The security experts must accept that often "good enough" security is better than as-strong-as-possible security. For example, the key management procedures required by a full strength security solution might not be possible due to the planned deployment scenarios of the application. Both people need to pay much more attention to economics, to the interests of the key players, and to deployment scenarios.

Acknowledgments

The authors would like to thank Bernard Aboba, James Kempf, Steven Bellovin, and Stephen Kent for interesting discussions in this problem space.

References

1. B. Aboba, J. Tseng, J. Walker, V. Rangan, and F. Travostino. Securing Block Storage Protocols over IP. Internet Draft draft-ietf-ips-security-19.txt (Work In Progress), IETF, January 2003.
2. J. Arkko. Effects of ICMPv6 on IKE and IPsec Policies. Internet Draft draft-arkko-icmpv6-ike-effects-01.txt (Work In Progress), IETF, June 2002.
3. J. Arkko SEcure Neighbor Discovery (SEND). Internet Draft draft-arkko-send-ndopt-00.txt (Work In Progress), IETF, June 2003.
4. J. Arkko, P. Nikander, T. Kivinen, and M. Rossi. Manual SA Configuration for IPv6 Link Local Messages. Internet Draft draft-arkko-manual-icmpv6-sas-01 (Work In Progress), IETF, June 2002.

5. J. Arkko, T. Aura, J. Kempf, V.-M. Mantyla, P. Nikander, and M. Roe. Securing IPv6 Neighbor Discovery. In *Wireless Security Workshop*, September 2002.
6. J. Arkko, J. Kempf, B. Sommerfeld, and B. Zill. SEcure Neighbor Discovery (SEND). Internet Draft draft-ietf-send-ipsec-01.txt (Work In Progress), IETF, June 2003.
7. J. Arkko, V. Devarapalli, and F. Dupont. Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents. Internet Draft draft-ietf-mobileip-mipv6-ha-ipsec-06.txt (Work In Progress), IETF, June 2003.
8. S. Bellovin. Guidelines for mandating the use of IPsec. Internet Draft draft-bellovin-useipsec-00.txt (Work In Progress), IETF, October 2002.
9. M. Blaze et al. The KeyNote Trust-Management System Version 2. RFC 2704, IETF, September 1999.
10. T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, IETF, January 1999.
11. R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). Internet Draft draft-ietf-dhc-dhcpv6-28.txt (Work In Progress), IETF, November 2002.
12. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, IETF, November 1998.
13. J. Ionnadis. Why don't we still have IPsec, dammit. NDSS 2003, February 2003.
14. D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. Internet Draft draft-ietf-mobileip-ipv6-24.txt (Work In Progress), IETF, June 2003.
15. S. Kent and R. Atkinson. IP Authentication Header. RFC 2402, IETF, November 1998.
16. S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, IETF, November 1998.
17. R. Moskowitz, P. Nikander, and P. Jokela. Host Identity Payload and Protocol. Internet Draft draft-moskowitz-hip-07.txt (Work In Progress), IETF, June 2003.
18. P. Nikander. Secure Neighbor Discovery using separate CGA extension header. Internet Draft (Work In Progress), IETF, June 2003.
19. B. Patel, B. Aboba, W. Dixon, G. Zorn, S. Booth. Securing L2TP using IPsec. RFC 3193, IETF, November 2001.
20. T. Narten and R. Draves. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 3041, IETF, January 2001.
21. T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461, IETF, December 1998.
22. Greg O'Shea and Michael Roe. Child-proof authentication for MIPv6 (CAM). *Computer Communications Review*, April 2001.
23. E. Rescorla. Guidelines for Writing RFC Text on Security Considerations. Internet Draft draft-iab-sec-cons-03.txt (Work In Progress), IETF, January 2003.
24. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-time Applications. RFC 1889, IETF, January 1996.
25. S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462, IETF, December 1998.
26. T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. SSH protocol architecture. Internet Draft draft-ietf-secsh-architecture-12.txt (Work In Progress), IETF, January 2002.

Limitations of IPsec Policy Mechanisms
(Transcript of Discussion)

Jari Arkko

Ericsson Research NomadicLab, Finland

Matt Blaze: I'm not sure if this is deliberate or not, but when you talk about policy stuff you seem to be mixing application policy issues with IPSec policy issues. For example, whether or not X contains the routing for address A is not solved even if you do security of the DPD messages. You need to secure the semantics of DPD, you need a better policy for UPP itself. It's not clear that that's a deficiency of IPSec policy per se. That's a deficiency of trying to use communication security to solve an application semantics security problem, which is more than communication security.

Reply: I think I agree, but we have certain security problems where we don't really want to go into exactly which layer does what. The whole problem has to be solved whether it's divided into IPSec paths, communications or something else.

Pekka Nikander: The original intention was to give IPSec the ability not only to protect application protocols but also to protect IPSec protocols like mobility protocols and neighbourhood discovery protocols from attacks. Now you can't design the protocol without knowing where you are going to use it, it just doesn't work that way.

Matt Blaze: But we did know, we were just wrong.

Pekka Nikander: Basically we are agreed that it's better to just forget about policies and instead try to take a fresh look at the issue of how we deal with integrated communication and applications. It may be that policies are bad.

John Ioannidis: I think where we went wrong was that we thought that we could use only IPSec for security. Later practice and experience shows that there are some cases where we are worried about communication security. The same goes for VPN, let's just admit it and stop trying to shoehorn IPSec where it doesn't fit. At the same time we shouldn't just say IPSec is only for VPN.

Virgil Gligor: Remember what IPSec is, the highest layer of end-to-end link in which we need security.

John Ioannidis: On the policy front, IPSec policy is for a very specific need, and that is to decide what is encrypted. Just like it was a mistake to try to use IPSec for everything, it would be a mistake to try to use that definition of IPSec security to try and to solve all our security or our policies.

Matt Blaze: In particular, one of the authentication problems that you have is that to identify an end point of an IPSec imitation is to identify a network layer node, and that's almost never what we want to identify at the application level. In fact, with personal computers it's very tempting to assume that a human identity and a machine identity are the same thing, but you know, that's an obvious mistake.

Pekka Nikander: I think one of the real problems here is that whatever the security protocol that you're building, whether it's IPSec, or whether it's not, you really need to pay attention to the needs of the application. Geraint spoke yesterday about the interface between the application and the security protocol.[1] The question that we have in our minds is: what about policy? People would gladly use something else like Keynote. Would that be any better?

Matt Blaze: Well the thing is, IPSec policy is at best a network layer security policy, but with IPSec you need something other than just network layer security policy. This is a really bad problem.

John Ioannidis: An end-to-end authentication policy would bridge the gap between application and authorisation with IPSec.

Matt Blaze: We could do that. We could do a network layer version of that, and building in things like Keynote would make it better, but it's still only giving a network layer authorisation. It's still only an existing policy of the network layer and we violate the layer here at our own peril.

John Ioannidis: And if your application lends its authorisation assumptions to the model, then all you need to know to make a decision on whether to allow an IMAP timed run is whether this is over a secure channel or not. That's fine. But that doesn't give your IMAP better security.

Birgit Pfitzmann: I truly don't know anything about why people use IPSec. A somewhat different question is to what extent does simple key distribution play a role in this? Say if you take a fixed policy, and say it's then a natural way you can just have keys distributed, would that then become a problem? How is it actually done?

Reply: I think the main thing here is the semantics of the network layer protection.

Birgit Pfitzmann: If you meet all the problems you have about the semantics and the policy, I have the feeling that the very first thing I would have to do is distribute the keys. Typically things like PKI are examples where practice stops because it doesn't get the keys distributed and you never get to the questions like application level and security. The question was odd in that it's really only the application and semantics and policy question.

[1] G. Price, "Client v. Server side Protocols, Interfaces and Storage", these proceedings.

Bruce Christianson: Would you be able to reduce the problem to a situation where authorisation corresponds precisely to authentication?

Pekka Nikander: It seems that it can be done in special cases, but IPSec is broken in several ways, and creating a security protocol in a vacuum, and especially trying to separate the key management from the communication security, and to separate the policy from key management, just doesn't seem right. If you try to use a system like that you would end up with...

Michael Roe: I think we realise that even if we had the label bound to a machine address and built it, it still wouldn't do the business because we were unable to express at the IPSec level the security policies with which it's expressed.

Pekka Nikander: That's another thing that people know about here. To convert this into very specific protocols like SET – these are not IPSec protocols where you have a very special protocol for a special cause – seems to be a very hard problem. How should we design security protocols so that the authority can't generate a protocol with another architecture and separate the parts, because you'll get an authorisation problem? On the other hand if you do a protocol which doesn't really need a hand-off then there is the key distribution problem. I think there is a lesson to be learned in protocol design but what exactly the lesson is, I don't know!

Deniable Authenticated Key Establishment for Internet Protocols

Colin Boyd[1,⋆], Wenbo Mao[2,⋆⋆], and Kenneth G. Paterson[3,⋆⋆⋆]

[1] Information Security Research Centre, Queensland University of Technology, Brisbane, Australia
[2] Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol BS12 6QZ, UK
[3] Information Security Group, Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK

Abstract. We propose two public-key schemes to achieve "deniable authentication" for the Internet Key Exchange (IKE). Our protocols can be implemented using different concrete mechanisms and we discuss different options; in particular we suggest solutions based on elliptic curve pairings. The protocol designs use the modular construction method of Canetti and Krawczyk which provides the basis for a proof of security. Our schemes can, in some situations, be more efficient than existing IKE protocols as well as having stronger deniability properties.

1 Introduction

Systematic design of protocols for key establishment protocols was pioneered by Needham and Schroeder in 1978 [28]. Understanding of protocol design and security analysis has advanced steadily in the years following their ground-breaking work. This paper follows in that tradition.

In this paper we consider (i) a special privacy feature for Internet Protocols, (ii) its realisation and (iii) its security analysis. In this introduction section let us provide abstract-level descriptions of these three things.

1.1 Deniability for Internet Protocols

As electronic communications become ever more a part of everyday life, issues of individual privacy have started to increase in relevance. One property that was not often thought important in the past for key establishment, but is seen as increasingly relevant today, is *deniability*. This is a privacy property that ensures protocol participants can later deny taking part in a particular protocol run. Such a property has been declared as desirable for new protocols proposed to secure the IP (Internet Protocol) level on Internet communications [22]. A

⋆ Research conducted during a visit to Hewlett-Packard Laboratories, Bristol.
⋆⋆ Research partially funded by the EU Fifth Framework Project IST-2001-324467 "CASENET".
⋆⋆⋆ Supported by the Nuffield Foundation, NUF-NAL 02.

deniability service offered at the IP layer preserves this privacy feature from the upper layers, in particular, the application layer. This is important because sophisticated anonymity and privacy properties offered at the application layer can often be undermined by inappropriately designed lower layer protocols. Therefore, deniability at the IP layer should not be viewed as a luxury feature: it is an important option.

1.2 An Underlying Technique

Identity-based public-key cryptography (ID-PKC), in which public keys can be obtained and trusted without the use of certificates and without the need for a CA-based public-key infrastructure (PKI), has been the subject of much recent interest. This interest was sparked by three seminal papers being positive applications of a cryptanalysis result of Menezes, Okamoto and Vanstone [27], called the MOV reduction, which used a pairing technique to reduce the difficulty of the elliptic curve discrete logarithm problem in a "weak curve" group to that in a finite field. The first two were independent original works: one was by Sakai, Ohgishi and Kasahara [32] whose application of the MOV reduction is an ID-based, non-interactive key agreement protocol; the other was by Joux [23] whose application of the MOV reduction is a one-round three-party Diffie-Hellman key sharing (Joux names it "tripartite Diffie-Hellman"). Boneh and Franklin [6] further applied a "distortion map" technique of Verheul [34] which improved positive applications of the pairing technique: their result is the first proven secure and fully practical ID-based encryption scheme, answering a long-standing open problem of Shamir [33].

ID-PKC makes use of an alternative trust model to a traditional PKI, wherein a trusted authority (TA) is trusted to create and deliver private keys to entities, and not to abuse its knowledge of these keys. This trust model is well suited to building security solutions when a central TA can be used (for example, in many corporate scenarios). The schemes of [32, 6] will form the basis for our construction of alternatives to IKE which enjoy the strongest possible deniability properties.

1.3 Provable Security for Key Establishment Protocols

Today there are many protocols for key establishment that have a proof of security, particularly using the fairly mature model of Bellare and Rogaway [4, 5]. Yet new protocol designs continue to appear in the literature as new requirements are revealed for different applications. A feature of the Bellare-Rogaway model is that proofs tend to be monolithic and fragile; a small change in a protocol design may require a completely new proof to be developed. An alternative, newer, model by Canetti and Krawczyk [11] has the advantage that proofs are modular, making it easier to re-use partial proofs and instantiate abstract protocols with different concrete algorithms.

1.4 Our Work

The purpose of this paper is to suggest new, identity-based key establishment protocols that are suitable for use as an IKE replacement. We concentrate on two properties that have been lacking in some previous proposals, namely strong deniability and provable security. Our protocols enjoy stronger deniability properties than previous protocols based on signatures and a traditional PKI trust model. At the same time we are able to propose flexible, provably secure solutions through the use of the Canetti-Krawczyk design approach, and derive solutions which can be at least as efficient as other proposed protocols.

We regard the following as the main contributions of the paper.

– A generic design for identity-based key establishment with proven security in the Canetti-Krawczyk model.
– Two new abstract protocols with strong deniability properties.
– An investigation of different specific algorithms to provide concrete versions of our abstract protocols with properties comparable to existing proposals for Internet key exchange.

The remainder of this paper is structured as follows. In Section 2 we introduce IKE protocols and discuss one current proposal. Section 3 explores different ways to obtain shared secrets from public information, which is a main theme of our constructions. Section 4 explains how pairings on elliptic curves can be used to provide identity-based cryptography. This provides perhaps the most promising concrete implementation of our protocols. Then in Section 5 the provable security model of Canetti and Krawczyk is reviewed, including the building blocks that we need for our protocols. Section 6 puts the building blocks together to arrive at the new protocols with the properties we require.

2 Internet Key Exchange

The key establishment protocol known as Internet Key Exchange (IKE) was published by the IETF as a proposed standard for Internet Protocol Security (IPSec)[1] in 1998 [21]. However, many criticisms have been made on various aspects of the protocol, particularly its considerable complexity [17, 30, 35, 36]. IKE is really several protocols, since there are multiple options that can be used in any particular run. Recently there have been alternative proposals and it now seems inevitable that a new version of IKE will soon emerge with significant differences. Detailed descriptions of the current IKE protocols are given by both Borella [9] and Cheng [13] which are useful supplements to the description in the draft standard [21]. Canetti and Krawczyk [12] have provided a security proof for a generic protocol construction known as SIGMA [24]. This construction applies to the IKE protocol and so that is now supported by a formal security proof.

We use SIGMA as an illustration of a typical current IKE protocol. This also allows us to introduce our notation and give an idea of the main properties of

[1] We sometimes see IPsec in the literature, but we believe IPSec makes better sense.

IKE protocols. All the protocols in the current IKE versions use key agreement based on Diffie-Hellman key exchange [14], although the group in which this exchange takes place can be negotiated as part of the protocol.

Protocol 1 is based on generic Diffie-Hellman key exchange in a group of prime order q. Although we have used multiplicative notation we do not restrict the group, which could be a subgroup of $\mathbb{Z}_p^*$ or an elliptic curve group or any other appropriate choice. The protocol also makes use of any generic digital signature scheme which must be secure against adaptive chosen signature attacks – we denote the signature of entity X on message m as $Sig_X(m)$. A messages authentication code (MAC) is also required, and we denote the MAC of a message m using shared key K as $MAC_K(m)$. The shared secret is $Z_{AB} = g^{r_A r_B}$, and the session key is derived from Z_{AB}.

We note that this protocol, and many derivatives of it (including Σ_0, Σ_1 and their applications in IKE and IKEv2) suffer from a minor authentication flaw. We sketch this flaw in the Appendix.

A		B
$_A \in \ \mathbb{Z}_q$		
$t_A = g^{r_A}$	t_A	
		$_B \in \ \mathbb{Z}_q$
		$t_B = g^{r_B}$
		$Z_{AB} = t_A^{r_B}$
		$K_m = H_1(Z_{AB})$
Verify signature	$t_B \ B \ \underset{\longleftarrow}{\ } ig_B(t_B \ t_A) \ MAC_{K_m}(B)$	
$Z_{AB} = t_B^{r_A}$		
$K_m = H_1(Z_{AB})$		
Verify MAC		
	$A \quad ig_A(t_A \ t_B) \ MAC_{K_m}(A)$	Verify MAC
		Verify signature

Protocol 1: Basic SIGMA protocol of Krawczyk

2.1 Deniability

One of the many desirable properties possessed by Protocol 1 is a form of anonymity which is called "plausible deniability". Harkins *et al.* [22] state that plausible deniability:

> . . . enables Alice and Bob to communicate, leaving no ability for anyone to prove that they did have a conversation, even with the collusion of one of the parties.

Plausible deniability allows any user who may have taken part in a protocol run to deny that this was the case, since there is nothing to connect the user directly to a particular communication partner or protocol instance.

Plausible deniability may not provide strong enough anonymity for some situations. In particular, if the communications partners are mutually distrustful then they cannot be sure that the other will not provide a secret value which can link one or both partners to the protocol run. For example, although plausible deniability was a stated goal of Protocol 1, it has limitations in this regard. As long as both A and B cooperate, no third party can show that either of them was involved in a particular run. All that is available is a signature of both parties on two random Diffie-Hellman inputs. However, suppose that A wishes to implicate B in a particular session. By storing the random input, r_A, and the shared secret, Z_{AB}, A can show that B signed the parameters corresponding to session key. A similar situation holds if B wishes to implicate A.

A stronger form of deniability can be achieved using shared-key authentication. With a shared-key solution, either user in the protocol run could have produced all the messages in the run. An even stronger form of deniability can be obtained when the shared key is obtained using techniques from identity-based cryptography. Now, any user can simulate runs of the protocol involving *any* other potential user. Moreover, a user is not required even to reveal if he is registered with the trusted authority in order to take part in runs of the protocol; this is in contrast to schemes which employ certificates and a traditional PKI (including the signature-based schemes), where the existence of a certificate for a user indicates that user has registered, and identifies him as a *potential* protocol participant amongst all the users in a population. It is one aim of this paper to explore in detail specific protocols with these strong deniability properties.

3 Shared Secrets from Public Information

Below we consider key establishment protocols which can be used to derive a new key suitable to protect a subsequent communications session. The basis for the design is that the two users should be able to derive a shared secret which will be used as the key for a MAC. Because both parties can derive this secret, either of them can deny having taken part in the protocol. This property can be very useful in optimising the communications efficiency of protocols and can also be the basis of strong deniability. Suppose that F_{AB} is the fixed shared key between principals A and B used to authenticate the messages in a key establishment protocol. Even if one of A and B collaborates with an adversary, it is impossible to prove that the other was actually involved in the communication. This is because either party could have efficiently computed all the messages in the protocol run. Note that because F_{AB} is a static secret it is not suitable as a session key to protect communications by itself.

In order to ensure that the protocol is deniable, the shared secret should be derived from information that is, or could be, public. This means that no cryptographic channels need to be (in fact, can be) used by the parties before the shared secret is derived. The public information could include several types of data. In particular it could consist of any set of the following items:

1. Identities of principals.
2. Public key parameters, such as moduli and group generators.
3. Public key values.
4. Public key certificates.

In this section we consider some available options for obtaining such shared secrets. We first look at more established methods and then introduce the newer identity-based methods. This latter option seems to have advantages in our scenario.

3.1 Shared Secrets from Public Keys

A well-known method to obtain a shared secret is the Diffie-Hellman key agreement protocol [14]. Diffie-Hellman keys are generally divided into two types: static keys and ephemeral keys. For static Diffie-Hellman each party has a fixed public key supported by a certificate; if x_A is Alice's private key then her public key is $y_A = g^{x_A}$, where g generates a suitable group. If Bob has a similar public/private key pair then he can generate the static Diffie-Hellman key $F_{AB} = y_A^{x_A} = g^{x_A x_B}$. Alice can also generate the same value in a symmetrical manner. The static Diffie-Hellman key is a shared secret which requires all four of the types of information enumerated above. In terms of deniability, a disadvantage of the use of certificates is that these are public values which can be used to verify that the owner of the certificate has at the least registered to take part in the scheme.

Girault's key agreement scheme [20] allows a shared secret to be obtained without exchange of messages, but it does require that the public keys of each party are known to the other. Girault's scheme uses *self-certified* public keys which cannot be used to verify that any user has registered in the scheme. The scheme uses an RSA modulus n and key pair e, d chosen by the trusted authority TA, together with an element g of high order in $\mathbb{Z}_n^*$. When A registers to use the scheme she chooses her private key x_A and provides g^{x_A} to TA, who calculates the self-certified public key $y_A = (g^{x_A} - ID_A)^d \bmod n$. (We have changed the sign of x_A from Girault's original description in order to maintain a uniform notation.) Girault's scheme uses the self-certified keys to produce an authenticated static shared key. If the public keys are already available this can be achieved with no message exchanges: A calculates $(y_B^e + ID_B)^{x_A}$ in order to produce $F_{AB} = g^{x_A x_B}$ and B calculates the same value in an analogous way.

Many identity-based schemes (including those using pairings explained below) require A's private key to be generated by the trusted authority. This need not be the case for Girault's scheme since S cannot find x_A from g^{x_A} without taking discrete logarithms in $\mathbb{Z}_n^*$. However, Saeednia [31] has pointed out that a malicious TA could choose n so that finding discrete logarithms is easy, and for this reason the size of n should preferably be at least 2048 bits. In addition S

should provide a proof that n is the product of two safe primes. This significantly reduces the bandwidth efficiency of Girault's scheme.

3.2 Shared Secrets from Identities

The notion of identity-based cryptography was first introduced by Shamir [33]. Shamir proposed a concrete identity-based signature scheme but no identity-based encryption scheme. Recently there has been a resurgence of interest in identity-based cryptography. The main reason was the discovery, beginning with Sakai, Ohgishi and Kasahara [32], and Boneh and Franklin [6], that identity-based cryptographic primitives can be realised using pairings on elliptic curves. A secondary reason was the realisation that ID-PKC uses a trust model that is more appropriate then a traditional CA-based PKI model for some types of application. A survey of recent activity in the area has been made by Paterson [29].

One striking consequence of an identity-based infrastructure is that any pair of parties can derive a shared secret without the need for any interaction or any certificates [32]. This seems to provide the optimal situation for deniability, since users have neither certificates nor public keys, and are able to derive the shared secret F_{AB} from knowledge of the peer's identity alone, along with any public parameters. The mathematical basis for this is pairings on elliptic curves, which allows new algebraic structures to be exploited. We outline how this works in the next section.

4 Pairings on Elliptic Curves

In this section, we review key agreement schemes and identity-based encryption schemes based on pairings. We use notation of Boneh and Franklin [7]. We let $\mathbb{G}_1$ be an additive group of a large prime order and $\mathbb{G}_2$ be a multiplicative group of the same order. We assume the existence of an efficiently computable bilinear map e from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$. Typically, $\mathbb{G}_1$ will be a large prime-order subgroup of the group of points on an elliptic curve over a finite field, $\mathbb{G}_2$ will be the same order (multiplicative) subgroup of a related finite field (a small field extension) and the map e will be derived from either the Weil or Tate pairing on the elliptic curve. We assume that the pairing mapping has been modified (by applying Verheul's "distortion map" [7, 34]) and so for every non-infinity point $P \in \mathbb{G}_1$ it satisfies $e(P,P) \neq 1_{\mathbb{G}_2}$. By e being bilinear, we mean that for $Q, W, Z \in \mathbb{G}_1$, both

$$e(Q, W+Z) = e(Q,W) \cdot e(Q,Z) \quad \text{and} \quad e(Q+W, Z) = e(Q,Z) \cdot e(W,Z).$$

When $a \in \mathbb{Z}_q$ and $Q \in \mathbb{G}_1$, we write aQ for scalar multiplication of Q by a. As a consequence of bilinearity, we have that, for any $Q, W \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$:

$$e(aQ, bW) = e(Q,W)^{ab} = e(abQ, W).$$

We refer to the growing literature [1,2,6–8,16,18,19] for a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security. We assume in what follows that suitable groups $\mathbb{G}_1$ and $\mathbb{G}_2$, a map e and an element $P \in \mathbb{G}_1$ have been chosen, and that elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ can be represented by bit strings of the appropriate lengths.

We also introduce here the computational problems that will form the basis of security for our identity-based schemes.

Bilinear Diffie-Hellman Problem (BDHP): Let $\mathbb{G}_1$, $\mathbb{G}_2$, P and e be as above with $\#\mathbb{G}_1 = \#G_2 = q$ being prime. The BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ with $a, b, c \in \mathbb{Z}_q^*$, compute $e(P,P)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if

$$\Pr\left[\mathcal{A}(\langle P, aP, bP, cP \rangle) = e(P,P)^{abc}\right] \geq \epsilon.$$

Here the probability is measured over random choices of a, b, c in $\mathbb{Z}_q^*$ and the internal random operation of $\mathcal{A}$. We assume that BDHP is a hard computational problem: letting q have the magnitude 2^k where k is a security parameter, there is no polynomial time (in k) algorithm which has a non-negligible advantage ϵ (again, in terms of k) in solving the BDHP for all sufficiently large k.

It is important for our later discussion to note that by appropriately building our groups from supersingular curves over fields of characteristic 3, we can arrange that elements of $\mathbb{G}_1$ have a representation that is 260 bits in size while, at the same time, the best known algorithm for solving BDHP has complexity at least on the order of 2^{80} (it is difficult to exactly quantify the security because it depends on the complexity of Coppersmith's discrete logarithm algorithm in the field $\mathbb{F}_{3^{163}}$). By generalising our definition of a pairing to allow two different groups on the left-hand side of the map e, we can have much greater flexibility in the choice of groups available and the size of their bit representations. We refer to [8, Section 4] and [2,16] for further details.

Sakai *et al* [32] proposed the following non-interactive key agreement scheme using pairings (our description of their scheme uses the distortion mapped Weil pairing and so is a simplified one). Suppose A and B register ahead of time with a Trusted Authority (TA). The TA has chosen as a master secret an integer s, and gives A and B private keys which depend on s and the individual's identities. This is done as follows. The TA picks and makes public a cryptographic hash function H whose inputs are strings of arbitrary length and whose outputs are elements of $\mathbb{G}_1$. Then the TA computes as A's private key the value sQ_A where $Q_A = H(ID_A) \in \mathbb{G}_1$ is a publicly computable function of A's identity ID_A. Likewise, the TA gives B the value sQ_b where $Q_B = H(ID_B) \in \mathbb{G}_1$. Now A and B can both compute the shared key

$$F_{AB} = e(sQ_A, Q_B) = e(Q_A, Q_B)^s = e(Q_A, sQ_B),$$

each doing so by combining his private key with the other party's public identity information.

It is proven by Dupont and Enge [15] that a slight variant of this procedure generates a key which is secure against adversaries who can extract the private keys for arbitrary identities (except for A and B), provided that the BDHP is computationally infeasible. Notice however that the TA can generate A and B's common key for himself. So A and B must trust the TA not to produce communications protected by this key, nor to disclose the key to other parties.

The second new protocol that we propose in this paper makes use of public key encryption in addition to a MAC using the shared key F_{AB}. We want to employ identity-based encryption for this purpose in order to maintain enhanced anonymity properties. (Recall that a certificate-based scheme would immediately reduce the set of entities who could possibly have participated in a protocol to those who have been issued with a certificate.) The identity-based encryption scheme of Boneh and Franklin [6] using pairings is ideal for this purpose. It has the advantage that it can use the same supporting infrastructure as the key agreement scheme of [32]. Its security also rests on the difficulty of solving the BDHP, in an extension of the standard IND-CCA model suited to the identity-based setting. We refer to [6, 7] for the details of this scheme and its security properties.

5 Canetti and Krawczyk Approach to Provable Security

Mathematical techniques for protocol analysis have evolved over several years. One of the first and most influential was the logic of authentication of Burrows, Abadi and Needham [10] which was followed by many attempts to improve it. The success of the BAN logic has encouraged many other researchers to look for methods to gain assurance in the security of cryptographic protocols. These methods have come mainly from either the formal methods community or the cryptographic community.

In the cryptographic community, proofs for security have been based mainly on the notion of *reduction* from the problem of interest to a better understood computational problem. Such an approach to protocol analysis was initiated by Bellare and Rogaway in 1993 [4]. Although still popular today, one limitation of this approach is its monolithic nature. In 1998, Bellare, Canetti and Krawczyk [3] introduced a modular approach to protocol proofs, which was later refined by the last two of these authors [11]. In this section we give an informal overview of their approach, and describe the building blocks we use in our protocol design.

The model defines protocol principals who may run multiple sessions. A powerful adversary attempts to break the protocol by interacting with the principals. In addition to controlling all the communications the adversary is able to *corrupt* any principal and choose its long-term key (this models insider attacks). The adversary may also *reveal* any accepted session keys. The adversary must be efficient in the sense of being a probabilistic polynomial time algorithm.

Definition 1. *An AKE protocol is called* SK-secure *if the following hold.*

1. *If two uncorrupted parties complete matching sessions, then they both accept the same key.*

2. *Suppose a session key is agreed between two uncorrupted parties and has not been revealed by the adversary. Then the adversary cannot distinguish the key from a random string with probability significantly more than 1/2.*

Two adversarial models are defined. The first can be considered as an ideal world in which messages are authenticated magically. The second can be considered the real world in which we run our real protocols; this is the model in which we want to prove our protocols secure. However, in order to modularise the process protocols are first proven secure in the ideal world and then translated into the real world.

The authenticated-links adversarial model (AM). In this model the adversary is able to invoke protocol runs, masquerade as protocol principals, and find used session keys. Although the adversary is quite powerful it is unable to fabricate or replay messages which appear to come from uncorrupted parties.

The unauthenticated-links adversarial model (UM). In this model the adversary can do everything that it can do in the AM, but can also replay and fabricate messages using anything it can calculate.

5.1 Authenticators

An MT-authenticator is the key mechanism in the modularisation of the process. It is a transformation that applies to each message flow of a protocol and transforms an AM SK-secure protocol into one in the UM.

Definition 2. *An* authenticator *is a protocol translator that takes an SK-secure protocol in the AM to an SK-secure protocol in the UM. An* MT-authenticator *(message transmission authenticator) is an authenticator that is applied to each separate message sent in the AM.*

Each flow of the protocol in the AM will become a multi-flow sub-protocol in the UM. The resultant secure protocol can be simply a concatenation of all sub-protocols. However such a simple approach generates an inefficient protocol and is of limited interest to us despite its proven security. Fortunately we can collapse flows of the sub-protocols with only one assumption: the proof does not rely on the sequence of the internal flows of a particular MT-authenticator. The process is straightforward and simply combines flows which will be sent in the same direction.

An MT-authenticator is *valid* if running it in the UM has the same effect as a simple message transmission in the AM. What this means is that for any efficient adversary in the UM against the MT-authenticator, there is an efficient adversary in the AM so that the outputs of the two adversaries are computationally indistinguishable. Canetti and Krawczyk [11] have proved the following theorem which is the basis of the methodology. In keeping with the descriptions in this section we again give a less formal version of the result.

Theorem 1 ([11]). *Let π be a secure protocol in the AM. Then the protocol formed by applying a valid MT-authenticator is a secure protocol in the UM.*

We can now summarise the Canetti-Krawczyk approach in the following four steps.

1. Design a basic protocol and prove it is SK-secure in the AM.
2. Design an MT-authenticator and prove that it is a valid authenticator.
3. Apply the MT-authenticator to the basic protocol to produce a protocol that is automatically secure in the UM.
4. As necessary, re-order and re-use message components to optimise the resulting protocol.

5.2 MAC Based Authenticator

Bellare, Canetti and Krawczyk [3] provided two public key based authenticators, one using signatures and the other using encryption. Since we aim to use an existing shared key (derived non-interactively) for authentication purposes, we will use a simpler MAC based authenticator mentioned by Canetti and Krawczyk [11]. Protocols based on existing shared keys have often seemed less interesting, except with the use of an on-line server. The use of certificated Diffie-Hellman values and identity-based techniques allows us to effectively apply this shared-key authenticator in a public key setting.

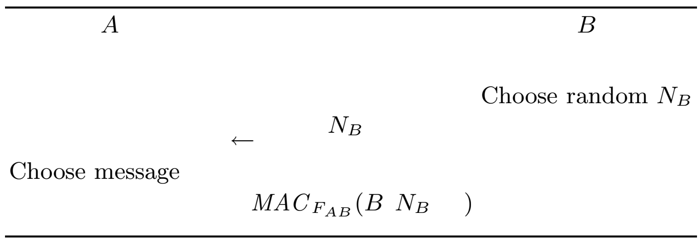


Fig. 1. MAC based authenticator

The MAC based authenticator, shown in Fig. 1, allows A to authenticate a message m which A wishes to send to B. The existing shared key F_{AB} is used as the key for the MAC. It can be proven that, provided F_{AB} is a securely shared key, this is a valid MT-authenticator, which means that when it is applied to the messages of a protocol in the AM, the result is a proven secure protocol in the UM. In the next section we apply this authenticator to two AM protocols to provide protocols with proven security and attractive deniability and efficiency properties.

6 Concrete Key Establishment Protocols

In this section, we sketch concrete key establishment protocols and their security proofs. These are based on the certificated Diffie-Hellman and identity-based approaches to generating shared secrets that we sketched in Section 3. The work of Canetti and Krawczyk actually provides two candidates which we can use to build secure key establishment protocols: the basic Diffie-Hellman protocol and a simple protocol based on public key encryption. We consider each in turn.

6.1 Key Establishment Using Diffie-Hellman

Protocol 2 is the result of applying a MAC-based authenticator to Diffie-Hellman exchange. In this protocol F_{AB} denotes a shared key that is derived from public information. Because of the use of a (static) shared key, either party can perfectly simulate protocol runs. Thus it enjoys a stronger deniability property than the original SIGMA design, Protocol 1, as we discussed in Section 2.1. The protocol structure is also simpler than Protocol 1. In repeated use of this protocol with the same party, the shared key computation is of course not required. This means that only MAC computations are needed, whereas Protocol 1 requires fresh signature computations for each protocol run. We also note that this protocol is not susceptible to the authentication flaw that SIGMA suffers from. We have three basic possibilities for the shared key F_{AB}:

1. F_{AB} could be obtained from static, certificated Diffie-Hellman values. In order to obtain a bandwidth efficient protocol, both these values and the ephemeral values exchanged in the protocol itself could be defined in an elliptic curve setting. The resulting protocol is bandwidth efficient (though the bits required to exchange certificates should not be discounted).
2. F_{AB} could be obtained using Girault's protocol as described in Section 3.1. Here the advantage is the lack of certificates; however the protocol is not particularly bandwidth-efficient in the light of the observations of [31].
3. F_{AB} could be obtained from the identity-based non-interactive key agreement protocol of Sakai *et al* using a single pairing computation. For this case, the ephemeral Diffie-Hellman session key agreement in Protocol 2 can also use an elliptic curve group. By choosing an appropriate elliptic curve group for the ephemeral values, we can obtain a particularly bandwidth-efficient protocol: with (say) a 128-bit MAC function and an elliptic curve over a 160-bit field with point compression, the total bandwidth for the protocol can be as low as $2 \times (161 + 128) = 578$ bits. This version of the protocol also enjoys our strongest form of deniability, where not even certificates are available to identify possible protocol participants.

By the results of Canetti and Krawczyk, Protocol 2 is secure in their model as long as F_{AB} is a shared secret that is chosen randomly and distributed securely to A and B. Each of the above methods for deriving F_{AB} requires its own assumptions in order to prove security. For example, for the last case the protocol

Shared Information: Fixed key F_{AB} derived from public information.

$$\begin{array}{ccc}
A & & B \\
{}_A \in \ \mathbb{Z}_q & & \\
t_A = g^{r_A} & t_A & {}_B \in \ \mathbb{Z}_q \\
 & & t_B = g^{r_B} \\
 & \underset{\longleftarrow}{t_B} \ MAC_{F_{AB}}(B \ t_A \ t_B) & \\
\text{Verify MAC} & & \\
 & MAC_{F_{AB}}(A \ t_B \ t_A) & \text{Verify MAC} \\
Z_{AB} = t_B^{r_A} & & Z_{AB} = t_A^{r_B}
\end{array}$$

Protocol 2: Key agreement based on Diffie-Hellman and shared key

of Sakai *et al* was proven secure by Enge and DuPont given that the BDHP is hard. In order to complete the security proof of our protocols we really need to prove that each of the MT-authenticators, with the different choices of the F_{AB} construction, are secure. However, in this paper we make the heuristic assumption that concatenating each of the above F_{AB} constructions with the MT-authentcator of Fig. 1 leads to a new secure MT-authenticator.

6.2 Key Establishment from Public Key Encryption

A second, encryption based, key establishment protocol is also proven secure in the AM by Canetti and Krawczyk. This protocol lacks forward secrecy but can be more efficient than using Diffie-Hellman. The protocol uses only one message in the AM, and basically consists of A choosing a new key and sending it to B by encrypting it under B's public key. Protocol 3 shows the result of applying the MAC based authenticator to the encryption-based AM protocol.

In order for Protocol 3 to be identity-based, the encryption used must also be identity-based. The Boneh-Franklin scheme [6, 7] outlined in Section 4 is ideal for this purpose. (Note that the security proof requires that the encryption algorithm is secure under chosen ciphertext attacks, so the variant in [6] with this property must be used.)

Once again, the shared key F_{AB} used in the MAC can be obtained by one of several methods; the most convenient (given that identity-based encryption is being used) is to make use of the identity-based non-interactive key agreement protocol of Sakai *et al.* Notice that the same public parameters and infrastructure can be used to support both the encryption scheme and the non-interactive key agreement scheme. It is advisable to use different identities for deriving the relevant public and private keys. For example, one might use $H(A||\,\texttt{encrypt})$ and $H(A||\,\texttt{share})$ for A's two public keys.

Shared Information: Fixed key F_{AB} derived from identity information. Security parameter t.

A B

$N_B \in \ [1 \ t]$

$\longleftarrow$ N_B

$K \in \ [1 \ t]$

$E_B(K) \ A \ N_A \ MAC_{F_{AB}}(B \ N_B \ E_B(K))$

Decrypt to obtain K and verify MAC

$Z_{AB} \underset{\longleftarrow}{=} MAC_K(A \ B \ N_A \ N_B)$

Verify MAC

Protocol 3: Key transport from identity-based encryption

Given that the protocol of Figure 1 is a valid MT authenticator for this choice of shared key, the security of Protocol 3 follows immediately. (We again make the heuristic assumption that deriving F_{AB} with any of our proposed methods does not break the security proof.) This protocol is also certificateless and perfectly simulatable by both parties, ensuring it has our strongest form of deniability. This protocol does not suffer from the authentication flaw that SIGMA has.

The protocol is also bandwidth and computationally efficient: the computation of F_{AB} and public-key encryption using the scheme of [6] require one pairing each, while the scheme of [6] results in compact encryptions (equal in size to the plaintext plus one element of $\mathbb{G}_1$ and one hash function output). With an appropriately chosen $\mathbb{G}_1$, group elements can be represented with around 260 bits. With a 128 bit session key K, 128 bit nonces N_A, N_B, and 128 bit MACs, the longest message in the protocol has $772 + |A|$ bits and the total communication is just 1028 bits.

7 Conclusion

We have described two identity-based protocols whose security is based on the generic security proofs of Canetti and Krawczyk. Although both these protocols arise naturally by applying the identity-based primitives to the building blocks of the Canetti-Krawczyk method, neither of them seems to have been published before. They have attractive properties that are not possessed by other protocols that have been considered in the literature. In particular, they have strong deniability properties and can be computationally and bandwidth efficient.

In future work we aim to complete the formal security proof of our protocols by proving that the MT-authenticator formed by first deriving F_{AB}, and then applying the MAC-based authenticator, remains valid. We also believe that it is worthwhile to explore more deeply the optimisation of these novel protocols with specific concrete algorithms.

Acknowledgments

We would like to thank Neil Dunbar of Hewlett-Packard Company for helpful discussions on IPSec and IKE.

References

1. P. S. L. M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002*, LNCS. Springer-Verlag, 2002.
2. P.S.L.M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in communication networks – SCN'2002*, volume 2576 of *LNCS*, pages 263–273. Springer-Verlag, 2002.
3. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428. ACM Press, 1998. Full version at `http://www-cse.ucsd.edu/users/mihir/papers/key-distribution.html`.
4. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO'93*, pages 232–249. Springer-Verlag, 1993. Full version at `www-cse.ucsd.edu/users/mihir`.
5. M. Bellare and P. Rogaway. Provably secure session key distribution – the three party case. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, 1995.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, volume 2139 of *LNCS*, pages 213–229. Springer Verlag, 2001.
7. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. `http://www.crypto.stanford.edu/~dabo/abstracts/ibe.html`, full version of [6].
8. D. Boneh, H. Shacham, and B. Lynn. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer-Verlag, 2001.
9. M.S. Borella. Methods and protocols for secure key negotiation using IKE. *IEEE Network*, pages 18–29, July/August 2000.
10. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society*, A426:233–271, 1989.
11. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – Eurocrypt 2001*, volume 2045 of *LNCS*, pages 453–474. Springer-Verlag, 2001. `http://eprint.iacr.org/2001/040.pg.gz`.

12. R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *Advances in Cryptology – Crypto 2002*, 2002.
13. P.-C. Cheng. An architecture for the Internet Key Exchange protocol. *IBM Systems Journal*, 40(3):721–745, 2001.
14. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transaction on Information Theory*, 22:644–654, 1976.
15. R. Dupont and A. Enge. Practical non-interactive key distribution based on pairings. Cryptology ePrint Archive, Report 2002/136, 2002. `http://eprint.iacr.org/`.
16. R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small MOV degree over finite prime fields. Cryptology ePrint Archive, Report 2002/094, 2002. `http://eprint.iacr.org/`.
17. N. Ferguson and B. Schneier. A cryptographic evaluation of IPsec. `http://www.counterpane.com/ipsec.html`, 2000.
18. S. D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Proceedings of AsiaCrypt 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *LNCS*, pages 495–513. Springer-Verlag, 2001.
19. S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory 5th International Symposium, ANTS-V*, volume 2369 of *LNCS*, pages 324–337. Springer-Verlag, 2002.
20. M. Girault. Self-certified public keys. In *Advances in Cryptology - Eurocrypt 1991*, LNCS, pages 490–497. Springer-Verlag, 1991.
21. D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*, November 1998. Internet RFC 2409.
22. D. Harkins, C. Kaufman, T. Kivinen, S. Kent, and R. Perlman. *Design Rationale for IKEv2*, February 2002. Internet Draft.
23. A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory, IV-th Symposium (ANTS IV), Lecture Notes in Computer Science 1838*, pages 385–394. Springer-Verlag, 2000.
24. H. Krawczyk. SIGMA: The SIGn and MAc approach to authenticated Diffie-Hellman and its use in the IKE protocols. `http://www.ee.technion.ac.il/~hugo/sigma.html`.
25. G. Lowe. Some new attacks upon security protocols. In *9th IEEE Computer Security Foundations Workshop*, pages 162–169. IEEE Computer Society Press, 1996.
26. W. Mao and K.G. Paterson. On the plausible deniability feature of Internet protocols. `http://www.isg.rhul.ac.uk/~kp/IKE.ps`, 2002.
27. A.J. Menezes, T. Okamoto, and S.A. Vanstone. Reducing elliptic curve logarithms to a finite field. *IEEE Trans. Info. Theory*, 39:1636–1646, 1983.
28. R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.
29. K.G. Paterson. Cryptography from pairings: A snapshot of current research. *Information Security Technical Report*, 7(3):41–54, 2002. `http://www.isg.rhul.ac.uk/~kp/pairings.ps`.
30. R. Perlman and C. Kaufman. Key exchange in IPSec: Analysis of IKE. *IEEE Internet Computing*, pages 50–56, November-December 2000.

31. S. Saeednia. A note on Girault's self-certified model. *Information Processing Letters*, 86:323–327, 2003.
32. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Sympoium on Cryptography and Information Security*, Okinawa, Japan, January 2000.
33. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of Crypto 84*, pages 47–53. Springer-Verlag, 1985.
34. E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In B. Pfitzmann, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'01, Lecture Notes in Computer Science 2045*, pages 195–210. Springer-Verlag, 2001.
35. J. Zhou. Fixing a security flaw in IKE protocols. *Electronics Letters*, 35(13):1072–1073, 24th June 1999.
36. J. Zhou. Further analysis of the Internet key exchange protocol. *Computer Communications*, 23:1606–1612, 2000.

Appendix

Unfortunately, the basic SIGMA protocol and its derivatives suffers from an authentication flaw similar to that developed by Lowe in [25]. In the attack, Mallory masquerades as other principals. Mallory masquerades as B to A and persuades A to initiate a protocol with him. Mallory communicates with B as himself in an incomplete protocol run.

1. $A \rightarrow$ Mallory("B"): g^{r_A}
   - $1'$ Mallory $\rightarrow B$: g^{r_A}
   - $2'$ $B \rightarrow$ Mallory: g^{r_B}, B, $Sig_B(g^{r_B}, g^{r_A})$, $MAC_{K_m}(B)$
2. Mallory("B") $\rightarrow A$: g^{r_B}, B, $Sig_B(g^{r_B}, g^{r_A})$, $MAC_{K_m}(B)$
3. $A \rightarrow$ Mallory("B"): A, $SIG_A(g^{r_A}, g^{r_B})$, $MAC_{K_m}(A)$
   - $3'$ dropped

At the end of the attack, I believes he has successfully completed a protocol run with R; in fact his run was with Mallory. On the other hand, R believes he has engaged in an incomplete protocol run with Mallory. The flaw we have demonstrated is *not* just a simple case of Mallory stopping the last message in a protocol run between A and B – that attack is possible against any protocol and is not particularly interesting.

For further discussion about this flaw and its consequences for building Denial of Service attacks on SIGMA and related protocols, we refer the reader to [26].

Deniable Authenticated Key Establishment for Internet Protocols

(Transcript of Discussion)

Wenbo Mao

Hewlett-Packard Laboratories, UK

Mike Roe: I don't really like using absolute deniability as opposed to plausible deniability. We called it plausible deniability in the beginning[1] partly as a joke; it was a reference to plausible deniability in the sense that a government official wants to be able to deny they were responsible for things they said off the record. More seriously it's plausible because we're thinking about non-repudiation. Anybody can say it wasn't them, but it may be that their signature proves they did it, so the fact that they say it wasn't them doesn't get believed. The reason it's plausible is when you say it wasn't him, and you go to a third party with all the evidence of the protocol, then it's plausible in the sense that the third party will think there's...

Bruce Christianson: There's some other way the signature could have got there.

Mike Roe: Yes, so it's plausible in the sense that the third party would think maybe he was telling the truth. Of course it can't be absolute because the whole idea about the protocol is you really did it, so there can't possibly be something that absolutely proves you didn't do it, because that then is a contradiction. So there has to be something using this area of doubt. I don't really like calling it absolute deniability.

Reply: I don't know, I would rather use the bad words. When Bob wants to allege Alice's involvement, it actually means that Bob must self-expose and say, "this is my public key." Even in that sense nobody can make a denial attack.

Mike Roe: I agree what you've got there is better than what's in IKE and many of the plausible deniability protocols we were looking at originally. I'm only looking to show that we thought it was a good idea. It wasn't a simulation. Bob can simulate a protocol run with Alice, even if Alice had absolutely nothing to do with it, and that's the way Alice can deny, by saying anybody can simulate

[1] Actually at the 3rd of these workshops. See also Michael Roe, "Cryptography and Evidence", PhD Thesis, University of Cambridge, 1997. URL: http://research.microsoft.com/users/mroe/THESIS.PDF.
Also at: http://www.chiark.greenend.org.uk/~mroe/thesis.pdf.

a run with me. That is what you've got there. What they have with SIGMA[2] has slightly weaker properties.

Reply: SIGMA is not a good example, it's always plausible if you look at their signature.

Matt Blaze: It would seem to be useful to borrow the terminology and the methodology from a model where deniability is essentially defined as receipt-freedom, so there is no proof of having behaved in a particular way.

Bruce Christianson: You can get a receipt for having voted, but not for what your vote was.

Matt Blaze: It's not a receipt that anyone other than you can interpret without deniability. It merely establishes that your vote was counted, with some rigorous definition of that. It would probably be useful to formalise that into what we have here.

Kamil Kulesza: If you try and formalise all these requirements then at least, at this stage, they won't be able to construct a system that is going to meet them all, you arrive at a contradiction.

Mike Roe: Other work has shown several of the goals of IKE turned out to be mutually exclusive. Particularly concealing the identity of the protocol participants against active attacks turned out to not be very compatible with many other goals.

Bruce Christianson: It was an own goal!

[2] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE-Protocols", Crypto 2003, LNCS 2729, 400–425.

Protocols for Supporting a Public Key Infrastructure in Ad Hoc Networks[⋆]

Mike Burmester and Alec Yasinsac

Department of Computer Science, Florida State University,
Tallahassee, Florida 32306-4530, U.S.A.
{burmester, yasinsac}@cs.fsu.edu

Abstract. Ad hoc networking technologies allow mobile devices to communicate by utilizing one another as communication relays. Resulting networks are highly dynamic, with nodes entering and leaving the network, for short duration membership. This absence of structure complicates planning in these networks, resulting in networks with limited functionality when their topology (their nodes and links) has highly varying and unpredictable characteristics.

We propose protocols for a Public Key Infrastructre (PKI) suitable for ad hoc networks that are based on threshold cryptography technology, adapted to general access structures.

Keywords: Ad hoc networks, security, access structures, threshold cryptography.

1 Levels of Trust in Ad Hoc Networks

Ad hoc networking techniques allow mobile devices to communicate among themselves by utilizing one another as communications relays. These networks have no fixed infrastructure and are highly dynamic. The absence of structure complicates and limits planning for these networks, potentially resulting in networks with limited functionality when their nodes and links have highly varying and unpredictable characteristics. As a simple illustration, message delivery is unreliable, even unlikely, if the communication latency is greater than the node lifetime within the network [19].

One essential limitation that befalls structure-less and structure-starved networks is the inability to establish a cryptographically secure Public Key Infrastructure (PKI). As a result, there is presently little theoretical foundation for securing these networks. In this paper, we propose a group PKI for ad hoc networks that is based on threshold cryptography techniques for appropriate access structures.

As we alluded to earlier, ad hoc networks are dynamically formed with no fixed structure. Traditionally, tacit structural assumptions allowed reasoning

⋆ This material is based on work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number DAAD 19-02-0235.

about such networks. The most common targets for structure are assumptions about the properties of the network nodes. For example, any dynamic (reactive) routing algorithm must assume that the lifetime of the route to destination node is longer than the duration of the routing algorithm itself, otherwise the target node cannot be reached before the route is established. However, this assumption is rarely stated. Research in ad hoc networks varies greatly because no broad classification of such networks exists. However, there are a great many de facto assumptions that guide much of the ongoing work. These often-overlooked assumptions presume unproven minimum node and path lifetime and ill-founded minimum and maximum network density and size bounds.

2 Public Key Infrastructures

Modern cryptography is based on three milestone technological advances:

1. Establishment of strong symmetric key encryption schemes [18,14]
2. Creation of Public Key Cryptography [9,16]
3. Introduction of secure Key Exchange Protocols [15].

Unfortunately, none of these technologies resolves the classic key distribution problem for dynamic networks with no fixed infrastructure. Solutions such as Kerberos key distribution [1] and the proposed solutions for Public Key Infrastructures [21] rely on significant fixed, or slowly evolving [4], infrastructures and require substantial associated set-up overhead. Such key exchange solutions are not appropriate for ad hoc networks.

More recently, flexible PKIs have been proposed for the ad hoc environment [22,13]. These methods are more suitable to dynamic networks than their predecessors in that they can support dynamic reconfiguration, yet they assume a substantial amount of synchronization between the nodes. Such overhead is highly undesirable in ad hoc networks.

2.1 Threshold Cryptography

Threshold cryptography has been proposed [5,7] as a mechanism to distribute the power to perform Public Key Cryptography operations, for example, to digitally sign or encrypt data. Parties are issued shares of a secret key and use these to execute the cryptographic application in a distributed way. For each application, in their computations, the parties learn no more about the cryptographic operation than is strictly necessary to execute the application. For a successful execution, a threshold of the parties is needed.

2.2 Adapting Threshold-Based Systems

An important question for any ad hoc service is how the service is able to perform under periods of rapid change. In our case, the question is how the system will continue operations if the original operating properties that established the threshold parameters become unworkable due to normally dynamic activity in the network. For example, what happens if a number of key certifiers leave the group, resulting in fewer than the threshold number of key certifiers?

A more challenging (and possibly more interesting) problem occurs when the size of the network increases so much that the key certifiers become bottlenecks, i.e. they are not able to handle the volume of key requests generated in the network. In such a case, we would like to be able to modify a t-of-n threshold cryptosystem to a t'-of-n' system.

The obvious solution is to reform the network with new parameters and new key certifiers. Alternatively, we prefer to extend the existing structure if we can do so in a way that minimizes the overhead. Adaptive threshold schemes have been proposed to modify threshold secret sharing schemes [10,8] but these are highly complex and require synchronization among the members that is difficult or impossible to accomplish in ad hoc networks. That is, for these schemes to work, the key certifiers must know that the scheme has changed and what the details of the change are.

2.3 Threshold Cryptographic Operations

The foundation of threshold cryptography [6] is that a threshold of participants is able to accomplish a designated cryptographic operation. More specifically, if n is the number of participants and t is the threshold, a (t, n)-threshold cryptographic scheme allows any t or more qualified participants to execute the operation. A threshold access structure, denoted by Γ, defines the qualified sets of participants that are able to execute the cryptographic threshold operation. For example, if G is a set with the three qualified participants a, b, and c, in a (2,3) threshold scheme, then $\Gamma(G) = \{(a, b), (a, c), (b, c), (a, b, c)\}$.

Most cryptographic applications involve (t, n)-threshold access structures. However, general access structures [2] reflect a variety of properties of ad hoc network nodes. We use the term "general" to reflect any monotone access structure, with or without the threshold limitation. For example,

$$\Gamma(G') = \{\{a\}, \{b, c, d\}, \{e, f\},\ \text{and all supersets of these}\}$$

is a general access structure for the set of participants $G' = \{a, b, c, d, e, f\}$. We shall also use the notation Γ_0 to denote minimal access structures. For the earlier example,

$$\Gamma_0(G') = \{\{a\}, \{b, c, d\}, \{e, f\}\}$$

is a minimal access structure of Γ.

Notice that we can also identify the maximal unauthorized sets of Γ, as

$$B_0 = \{\{b, d, e\}, \{b, d, f\}, \{b, c, e\}, \{b, c, f\}, \{d, c, e\}, \{d, c, f\}\}.$$

Here, maximal means that all unauthorized sets are characterized without redundancy. That is, for $0 < i \neq j \leq n = |B_0|$, no B_i is a subset of B_j for any two elements in B_0, since all subsets of unauthorized sets are necessarily unauthorized, thus are redundant. The notion of maximal unauthorized sets is essential to our construction in section 3.2.

In the next section we show how a general access structure is ideally suited to model security infrastructure aspects of ad hoc networks.

3 Access Structures for Key Certification in Ad Hoc Networks

In order to systematically accomplish any security goals, we must agree on some representative characteristics of the network nodes and links. While distributed methods prefer to rely on similar or homogeneous nodes, often, in heterogeneous applications, some nodes have more authority or trust placed in them than others in order to accomplish one or more group-oriented tasks. A node that acts as a key certifier reflects this property, requiring additional functionality and, in some sense, more authority than other nodes in the network.

In our model, the key certifier is not a single node; but rather is a group of nodes that jointly accomplish key certification for the group. This induces, in a natural way, an access structure on the nodes of the network.

It may also be the case that some key certifiers have more authority than others, essentially, that they are *more trusted*. The crux of this section is that trust levels can be reflected by implementation of threshold cryptography techniques employed against general access structures. Specifically, less trusted key certifiers may be paired with more trusted key certifiers in their respective qualified certification groups, while more trusted key certifiers may form small (or exclusive) certification groups with the same authority.

3.1 The Overarching Architecture

Utilizing certificates for group membership results in a natural partitioning of all possible communicating nodes into two categories, each with two sub-categories:

1. Outsiders
   (a) Uninterested nodes that never intend to join the group, but may be in the group communication reach [20].
   (b) Non-group members that may have desire and capability to join the group.
2. Insiders
   (a) Group members.
   (b) Key Certifiers. These trusted group members, in appropriate certification sets of the access structure, act as a group membership certifier.

We acknowledge the first of these as potential intruders, but since they will *always* be outsiders, we claim that the normal PKI characteristics should protect against them, thus we are otherwise uninterested in them. We claim that the primary role of a dynamic PKI is to robustly and securely utilize key certifiers to promote members from outsiders to group members, and to revoke them. As we noted earlier, there are many PKI schemes proposed for that purpose for non-dynamic models. The problem that we address is that of protecting the group from dishonest key certifiers, similarly to that presented in [13], but we go beyond with a novel mechanism that supports the dynamic nature of ad hoc networks.

Two main contributions of our scheme are that:

a. It is based on general access structures
b. That it can be adapted in simple way that corresponds directly to the needs in ad hoc networks.

Thus, our focus is on key certifiers. Our scheme is expandable [8,10,11], and has the added advantage that it allows key certifiers to be granted different trust levels.

As an example, consider a network with $n = 5$ key certifier nodes $\{n_1, n_2, n_3, n_4, n_5\}$, where any $t = 3$ nodes collectively act as the key certifier, or are considered key certifier participants. A standard threshold scheme [17,2,5] grants all triples of key certifier participants equal trust, by allowing them to collaborate to sign as the key certifier. Hence, each key certifier has one third of the signing power. General access structures allow nodes to be granted varying degrees of trust. For example, n_1 may be sufficiently trusted that it alone is allowed to promote nodes from outsiders to group membership by granting them a public key certificate, while nodes n_2 and n_3 are required to collaborate in order to issue a new certificate, and nodes n_4 and n_5 may be required to join together with n_3 to promote nodes to group membership. The minimal access structure for this arrangement is defined by:

$$\Gamma_0 = \{\{n_1\}, \{n_2, n_3\}, \{n_3, n_4, n_5\}\}$$

With this structure, outsiders have three options for achieving promotion to group membership by convincing each member of one of the three qualified certifying sets.

3.2 A Threshold Certification Scheme for General Access Structures

Mapping General Access Structures to Threshold Structures. As a starting point for our scheme, we use any access structure Γ for the qualified certifying sets. Then, we map Γ to a threshold structure Γ^* using the approach in Ito [12], or an approach similar to Horng [11]. The first reduction is rather inefficient, with the realized scheme being an (n, n)-threshold scheme for relative large n. The latter is more efficient giving a (k, n)-threshold scheme with smaller values of k and n.

As threshold structure, we propose to use an RSA threshold signature scheme [6].

An Illustration. Consider the network consisting of four key certifier participants $C = \{1, 2, 3, 4\}$. Given the access structure $\Gamma_0(C) = \{(1, 3), (2, 4)\}$, then $B_0 = \{(1, 2), (1, 4), (2, 3), (3, 4)\}$. Since $|B_0| = 4$, by Ito, et al [12], Γ can be mapped to a (4,4) threshold access structure Γ^*.

We must select RSA parameters: p, q primes with $|p| = |q|$, $n = pq$, and $d, e \in Z_{\phi(n)}$, with $\phi(n) = (p - 1)(q - 1)$ and $d \cdot e = 1 \, (\bmod \, \phi(n))$. Then,

- $\text{sig}_C(m) = h(m)^d \, (\bmod \, n)$ and,
- $\text{ver}_C(\text{sig}_C(m)) = 1$ iff $h(m) = \text{sig}(m)^e \, (\bmod \, n)$,

for a suitable hash function h.

Thus, we must select Γ^*-shares d_{ij} such that $x^{d_{1,2}} \cdot x^{d_{1,4}} \cdot x^{d_{2,3}} \cdot x^{d_{3,4}} = x$ (mod$n$), and such that the holders cannot learn anything about the factorization of n. These shares can be generated using the algorithm given in [3] by Boneh and Franklin. We map the Γ^*-shares to Γ-shares by selecting the members of the sets in the access structure as in Ito, et al. [12], to jointly interact with the Boneh/Franklin RSA generation algorithm.

Specifically, members 1 and 2 corresponding to $(1, 2) \in B_0$ collaborate to act as virtual user U_1; members 1 and 4 corresponding to $(1, 4) \in B_0$ collaborate to act as virtual user U_2; members 2 and 3 corresponding to $(2, 3) \in B_0$ collaborate to act as virtual user U_3; members 3 and 4 corresponding to $(3, 4) \in B_0$ collaborate to act as virtual user U_4. The shares $d_{i,j}$ are produced directly from the input contributed by the corresponding U_i virtual user.

We now distribute to each key certifier U_i, $i = 1, 2, 3, 4$ a Γ-share, s_i whose shadows are comprised of shares from the Γ^* scheme as:

U_1 :	$s_1 = (d_{2,3}, d_{3,4})$	(exclude 1)
U_2 :	$s_2 = (d_{1,4}, d_{3,4})$	(exclude 2)
U_3 :	$s_3 = (d_{1,2}, d_{1,4})$	(exclude 3)
U_4 :	$s_4 = (d_{1,2}, d_{2,3})$	(exclude 4)

Notice that with this distribution, U_1 and U_3 jointly possess each of the four shares necessary to sign in the Γ^* (4,4)-threshold scheme, as do U_2 and U_4, as is required in the given access structure $\Gamma_0(C)$.

We note that distributed RSA computation is complex and expensive. Even though this computation need only be conducted one time, for the sake of efficiency, our key certification scheme may be implemented by having a trusted member generate and distribute the shares of the RSA key scheme.

3.3 Extending the Certification Scheme

Because of the nature of ad hoc networks, as nodes enter or leave the network, qualified certifier sets C belonging to Γ will evolve, eventually reducing to sets C' that are not qualified C^* (not in Γ). If this is the case with all such sets C in Γ, then there will remain no qualified certifying sets in the network.

To maintain the operational capability of the certifying nodes, Γ must be able to evolve dynamically to address the ad hoc nature of the network. This evolution must be proactive, in the sense that it must protect the access structure from proactive threshold attacks [10]. For this purpose, we propose to combine the methodologies of Horng [11] and Frankel , et al [10].

Should a key server desire to distribute share, say d_1, to users 5 and 6, he would construct shares $d_{1,5}, d_{1,6}$ such that: $x^{d_{1,5}} \cdot x^{d_{1,6}} = x^{d_1} \pmod{n}$. To do this, he would select at random with uniform distribution[1] the share $d_{1,5} \in Z_n$. He would then compute $d_{1,6} = d_1 - d_{1,5}$ and issue share $d_{1,5}$ to user 5 and $d_{1,6}$ to user 6. Now, members 5 and 6 can jointly act as key server 1, essentially where any computation using d_1 is required.

3.4 Node Demotion

Group Membership Revocation. In our scheme, demotion occurs via certificate revocation. There are many mechanisms proposed for this purpose, with certificate revocation lists being the most common. We submit that the short-lived nature of ad hoc networks will result in very few such revocations. While revocation lists will be flooded, the overhead will be low because there will be relatively few revocations resulting in short lists, infrequently transmitted.

Key Certifier Demotion. Our model reflects the advanced trust placed in key certifiers via the selected access structure. For this reason, the need for key certifier demotion will be minimized. In the case where key certifier demotion is necessary, we recognize that our scheme is a distributed construction. Thus, we propose that key certifiers may issue certifier revocation lists that blacklist malicious certifiers, effectively demoting them by removing all certifying access sets that contain that member from the access structure Γ.

4 Conclusion

The classic problem of secure and efficient key exchange continues to challenge security researchers. Advances in this area tend to be incremental, seemingly with one step backward for each two (or fewer) steps forward. The problem is further exacerbated in ad hoc networks, where the structure leveraged in many of the incremental advances in key exchange is necessarily unavailable. In this paper, we propose utilizing threshold cryptographic techniques for a key exchange system for ad hoc networks. Due to the nature of ad hoc networks, only distributed services are suitable, and other key exchange infrastructures have been suggested. Our system has two primary distinguishing characteristics:

1. It is extendible, allowing modification on the fly, as the network changes.
2. It enables systematic implementation of authority levels.

[1] The distribution $\mathcal{D}^*_{\phi(n)}$ induced on $Z_{\phi(n)}$ by this selection is ε-statistically indistinguishable from the uniform distribution $\mathcal{D}_{\phi(n)}$, where $\varepsilon = \mathcal{O}(2^{-|n|/2})$. Indeed: $\frac{1}{2}\sum_{n-\phi(n)} |\frac{2}{n} - \frac{1}{\phi(n)}| + \frac{1}{2}\sum_{\phi(n)-(n-\phi(n))} |\frac{1}{n} - \frac{1}{\phi(n)}| < \frac{(2\phi(n)-n)(n-\phi(n))}{n\phi(n)} < \frac{(n-4\cdot 2^{|n|/2})\cdot 3\cdot 2^{|n|/2}}{2^{|n|}(n-2\cdot 2^{|n|/2})} = \mathcal{O}(2^{-|n|/2})$, when $|p| = |q|$.

Acknowledgements

We would like to thank N. Asokan and Le-Van Tri for helpful comments.

References

1. S.M. Bellovin and M. Merritt. Limitations of the kerberos authentication system. *Computer Communications Review*, October 1990.
2. G.R. Blakley. Safeguarding cryptographic keys. *in Proceedings of the National Computer Conference, FIPS Conf.erence Proceedings*, 48:313–317, 1979.
3. D. Boneh and M. Franklin. Efficient generation of shared rsa keys. *In B.S. Kaliski Jr., editor, Advances in Cryptology, CRYPTO '97*, Volume 1294 of Lecture Notes in Computer Science:425–39, Springer-Verlag, 1997.
4. M. Burmester and Y.G. Desmedt. Secure communication in an unknown network using certificates. *Advances in Cryptology - Asiacrypt '99, Lecture Notes in Computer Science 1716, Springer, Berlin*, pp. 274–287, 1999.
5. Y. Desmedt and Y. Frankel. Threshold cryptosystems. *In Crypto 89, Springer-Verlag Lecture Notes in Computer Science*, 435:307–315, 1990.
6. Y.G. Desmedt. Threshold cryptography. *European Trans. on Telecommunications*, 5(4):449–457, July-August 1994.
7. Y. Desmedt. Some recent research aspects of threshold cryptography. *In Proc. of the 1st Intl. Information Security Workshop*, pp. 158–173, 1997.
8. Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. *Technical Report ISSE TR-97-01*, George Mason University, July 1997.
9. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
10. Y. Frankel, P. Gemmel, P. MacKenzie, and M. Yung. Optimal resilience proactive public-key cryptosystems. *In Proceedings of the* 38^{th} *Symposium on Foundations of Computer Science*, 1997.
11. G. Horng. A new method for constructing multiple assignment schemes for generalized secret sharing. *Journal of Information Science and Engineering*, 17:959–965, 2001.
12. M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing any access structure. *Proceedings of IEEE Globecom '87*, pp 99–102, 1987.
13. A. Khalili, J. Katz, and W.A. Arbaugh. Toward secure key distribution in truly ad-hoc networks. *IEEE Workshop on Security and Assurance in Ad hoc Networks, in conjunction with the 2003 International Symposium on Applications and the Internet*, Orlando, FL, January 28, 2003.
14. National Bureau of Standards (NBS). *Data Encryption Standard.* Federal Information Processing Standard, Publication 46, NBS, Washington, D.C., January 1977.
15. R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, Vol. 21(12):993–999, December 1978.
16. R.L. Rivest, A, Shamir, and L.M. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, Vol. 21(2):120–126, February 1978.

17. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.
18. C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.
19. A. Yasinsac. Rates of Change in Ad Hoc Networks. To appear in the Proceedings of the Information Resources Management Association International Conference (IRMA2003), Philadelphia, PA, May 18–21, 2003.
20. A.Yasinsac and J.A. Davis. Modeling protocols for secure group communication in adhoc networks. *Tenth International Workshop on Security Protocols*, Cambridge, UK, Apr 17–19, 2002, LNCS.
21. P. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, MA., 1995 (second printing).
22. L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6):24–30, Nov./Dec. 1999.

Protocols for Supporting a Public Key Infrastructure in Ad Hoc Networks (Transcript of Discussion)

Alec Yasinsac

Florida State University, USA

The work that we're going to talk about today is close to the work that I talked about here last year[1], PKI ad hoc networks. But you know ad hoc networks are useless because you can't see any properties of truly ad hoc networks, so you have to assume some kind of properties of these nodes if you are ever to do anything with them at all.

So, the PKI is by nature ad hoc, but we think that we made a little bit of an advance on the certificate based route where no-one can construe the status by a certificate. You'll see how this works later on, but the certificate is signed by the group key. That of course begs the question about this infrastructure environment and how you're going to sign the key. You all know the limitations of systems that have a single sign: it's a single point of failure, it does all the things you have problems with in a secure environment, you get a bottleneck, it's also that single point compromise that you had at the top of the X509 issue. Either you instil complexity into it, or you instil multiple points of compromise. If one person is bad, depending on the scheme, that can make Bad Things Happen. At the very least they may be able to deny a service, but having multiple signing can create a problem property too. With threshold schemes you have the difficulty that maybe it can change in this environment. We think there's a better way, based not on threshold schemes but on dual access structures. The idea is that a threshold scheme and an access structure have a relationship, and that actually the set of access structures for threshold schemes is a pretty important set of structures that helps you understand how they work. Instead of having a single signer you have a single signing key, and you have to have people that are your key signers that collaborate to be able to sign that key in the group. If this is a two of five scheme, then two of the people who are of the qualified set would have to agree to sign that certificate.

This does give you some flexibility. For example, if you have a network of a thousand people then you could create a five of one hundred scheme, or a fifty of one hundred. Some large number of your certifiers can leave the network, and you still have a valid PKIs. If twenty leave you still have eighty certifiers who can potentially be contacted, to sign fifty people in the network. So it's a more robust system than just having a small set of fixed signers, but if your network

[1] A. Yasinsac, J. Davis, "Modelling Protocols for Secure Group Communications in Ad Hoc Networks", LNCS 2845, 189–202.

grows to ten thousand nodes you can have bottlenecks. It's much more difficult to go from a five in a hundred scheme to a fifteen of a thousand scheme, because the mathematics is much harder there.

So what we do here is we promote outside of the group members, once some level of trust has been established in a group of certified holders. The other thing that we have to be able to do is demote group members. We have problems in PKI, right, and we have to be able to demote certifiers; so essentially what a public key infrastructure is for, is to issue and revoke certificates, and to issue and revoke shares in this case, that support our access structure. You'll see in just a moment why it's not a standard threshold scheme.

Frank Stajano: Just to be sure I am following, do you mean that the certificate had to be issued by all the guys in the set signing?

Reply: Yes, in order for this key certifier to be involved in signing certificates, it must collaborate with this node. That's the only way that they can sign.

Frank Stajano: So it is not the case that any five out of your hundred could get together and sign? It's specific subsets, not just any subset of a certain cardinality that will work?

Reply: Exactly, and that's the distinction between the system that we propose, and the standard idea of a threshold access structure. I would propose what we really want is a single group signing key, where the group verification key is public obviously, and we're going to propose a certified key based on general access structures where the signing authority is distributed among these qualified sets. So as I said, this is not a threshold scheme.

And again, we claim that this models the way things are in the real world, some nodes may have more trust certified than other nodes, so we think this is an important contribution to be able to break the scheme down this way.

Now to construct these shares in order to be able to do that, we use the Ito, Saito and Nishizeki algorithm to be able to map a threshold scheme to a general access structure. And then we use those shares that we've constructed in this scheme to be able to map onto the distributed signature scheme that represents the qualified set that we're looking for.

Now we have the other problem of demotion. How do you revoke the certificates once they're issued? Obviously the key certifiers that sign keys can also sign revocation lists, and you also use a revocation list for key certifiers, because each key certifier in this scheme has to know who they can certify with. So if a particular key certifier is found to be dishonest then the good key certifiers know this, and can get together, put their name on a revocation list, and send it out to all the other users. It's essentially a black list.

Matt Blaze: The ones signing a revocation list needn't be the same ones that signed the original certificate, and in fact might be disjoint. So you can get this kind of metastable state of signing, revoking, signing, revoking over and over?

Reply: Yes you certainly could. And it's even a little bit more subtle, if you're thinking about the certificates, because we're trying to revoke shares. Once the share is revoked it's revoked, because in order to re-issue shares you have to re-issue a new scheme.

We talked about revoking shares, so now let's talk about combining them. Let's assume that someone wants to leave the network who is a trusted key certifier. If you're not a trusted certifier and you want to leave the network, then you take your certificate and you go. You still have it with you, but it's of no use to you because you're not in the net so you don't necessarily need to be revoked at all. But for the trusted certifiers, when they leave, it might be nice not so much to revoke them, but rather to allow someone else to pick up their load. So there's a very nice little algorithm to do that. If I'm leaving the group and I have a share to be able to sign a certificate then I just pass it up to Virgil and he can sign anything I can sign, that's one way to do it. It might also be useful to split my share up, because when you're promoting folks you may have less trust in them so you may want to split those shares and give them out to others.

Alastair Beresford: Presumably you'll need some share combining structure as well because otherwise everyone becomes a signer eventually. This is because when someone leaves they split their share in half and pass it on to two other people.

Reply: That could be true, a share combiner essentially would be when I give you my share, and somebody else gives you their shares, and you're set to go. Again, the idea here is that there can be trust assumed among these certifiers, but be aware of the model you want. After I have been in the network long enough - and we're talking about ad hoc networks, network life could be two hours or twenty days, without carrying down those things it's hard to reason strongly about - I understand it may create other weaknesses. For example, if I have all the shares then obviously that's a potential weakness. There's a scheme that prevents me from passing my shares to whoever I want in this mechanism.

Alastair Beresford: I guess if you pass them on randomly then people who stay the longest gather the most shares, and so on.

Reply: Right, if that's your mechanism then it would be. Time would be a factor, that's correct.

Birgit Pfitzmann: I think I didn't understand the revocation. Say there are people in your group that nobody else likes so you want to remove them. I think the revocation list doesn't help you very much, because the signatures that they make are just like everybody else's signature. You wouldn't be able to tell which were the signatures that had been revoked.

Reply: Sorry, the signature mechanism here is that you create your public and your private key, and then you can use the certificate that you receive, with your public key / private key within the group. The revocation that I'm talking about suggests that there has to be some type of identification number with that certificate if you ever revoke it.

Birgit Pfitzmann: Which are the membership keys? But the ability to admit members to the group, you can't really recognise who admitted someone can you, because it looks the same for everybody. With the signatures that they made with the shared key, you wouldn't know it was them. You could say everybody whom they admitted has a different key and immediately revoke them once you noticed that you didn't like them.

Bruce Christianson: If I want to prevent anybody who had been admitted by Virgil from using their shares, how can I do that?

Reply: I would have to have some way either to identify the identity of Virgil, or more likely, have some kind of serial number that goes along with these particular signatures.

Bruce Christianson: So I need to know who contributed the shares for a particular signature, is that right, even though the signature looks the same regardless?

Reply: Essentially, the idea would be that if Virgil had a certificate then if I am a signer and Matt and I are a qualified set, Matt and I would have to agree that we want to remove Virgil's certificate. So then we would have to have a way of identifying his certificate, and issue that in a revocation list to the rest of the group members who would then shun Virgil.

Victoria Stavridou: So they couldn't do anything about rolling back the people that he authenticated before we realised that he was bad? For example, he may have let in a whole bunch of people, and you couldn't go back and revoke their certificates?

Bruce Christianson: Even if I can't do it retrospectively, how do I stop him from contributing to signatures in the future?

Reply: Those are two different questions.

Richard Clayton: I don't think that revocation lists are useful for members. Key-side revocation lists don't buy you anything, because if you've got a group of three and one of them goes bad, then later the other two just have to agree that they will refuse to cooperate. It doesn't matter what else they do. However, if all three of them go bad then they'll instantly issue instructions to the whole group saying, "don't believe anybody else, we three are the only true power." If another group over there says exactly the same thing, then we might as well toss a coin because we've got no way of knowing who's telling the truth. The revocation list for key signers doesn't seem very useful.

Bruce Christianson: How do we stop someone on a revocation list from contributing to the signature if you can't tell from the signature if one was involved?

Reply: Again, there are two different revocation lists. If Matt and I must sign together, and Bruce and Virgil decide that I'm bad, they can sign a revocation list and send it to all the other trusted servers, which would include Matt, so

Matt would now know you two believe I'm bad and he will no longer sign with me, effectively changing the qualified list and revoking my ability to sign.

Matt Blaze: But what happens if both of us get revoked?

Reply: If both you and I get revoked, then we won't qualify as a set.

Matt Blaze: But then we can still sign, we can get together and create signatures.

Bruce Christianson: If a sufficient set goes bad then you're dead, that's one of the rules of the game.

Reply: That's the rule of the game. So you don't ever give me all the power unless you're really sure I'm a really good guy, and you don't give it to just Matt and me unless you're very sure that at least one of us is good.

Bruce Christianson: So in practice there will be other people in the signing group to tell everyone to shun you?

Reply: That's exactly correct. Again, this is complicated beyond the math and the complexities of the Ito et al processes and the generation process. It's all also complicated by the nature of the ad hoc environment where things change.

Victoria Stavridou: But that might be an advantage here, because it helps you correct the problem of say two out of the three going bad. You always have the possibility of making new signers, right? I think that's the trick, threshold with all the signers, then when you've thrown them out and you start a fresh set, you know the new ones who take over, so you can continue providing services.

Virgil Gligor: This is essentially what happens when you also replicate.

Victoria Stavridou: Oh but if you lose all the replicas you're dead. However in this case, because you have available nodes, you could just work through the alphabet and start over. The question is how do you roll back, if for example, I was bad and you didn't know for three or four cycles, I've authenticated a whole bunch of people, how do you go back and figure out who should be on the revocation list.

Reply: OK, that's next year's talk. [Laughter]

What We Can Learn from API Security
(Transcript of Discussion)

Ross Anderson

University of Cambridge

During the period when Mike and Bruce were looking for position papers for this workshop, I was rather busy because of a court case some of you may have heard about, so I'm going to go over the slides that I gave at Roger Needham's farewell do last month[1], with some extra material added. The subject is what API security teaches us in the wider world of protocols; so it's about protocol analysis, composability, computation, and the effects in the real world.

How do we define a security protocol's world? Well that's changing: in the classic literature there are rules for dealing with information used to verify principals' claims to identity, such as passwords, PINs, crypto keys and timestamps. Now it's expanding to include other claims: such as claims to authorisation, or claims to creditworthiness, or claims to have a particular bank balance available for an electronic payment.

Now one of the things that I thought it useful to point out in this talk was that there have been two parallel streams of protocol work, one in academia, and one in industry, and until very recently the two didn't meet at all. I started off working on security in the banking industry in the 1980s, and I think it was about 1990 before I even heard of the existence of conferences such as Crypto and Eurocrypt. There were one or two people in the banking crypto world who were active in both spaces, such as Don Coppersmith, but by and large these were two hermetically sealed worlds and neither knew of the existence of the other.

Now in the academic world, of course, we have things that we know and love such as the Needham-Schroeder protocol, where you set up a key and you try and show that it's fresh by using challenge response rather than timestamps.

From the industrial viewpoint, there are various things you can observe about academic-style crypto that don't require the tools that academics love. For example, an awful lot has been made of the fact that Alice knows the key is fresh but Bob doesn't in Needham-Schroeder; you don't have to use formal methods for this! All you have to do is observe the overall protocol structure – Bob has to rely on Alice for any assurance of freshness, because after he has communicated his challenge to Alice there is no further traffic. That's all you have to observe, you don't have to get involved with fancy formalisms.

So one of the things we can do when we find an academic world and an industrial world in parallel is to go round the 'hermeneutic circle' of constantly

[1] R.J. Anderson and M.K. Bond, "Protocol Analysis, Composability and Computation" in *Computer Systems — Theory, Technology, and Applications* (Ed. Andrew Herbert and Karen Spärck Jones), Springer 2004, ISBN: 0-387-20170-X, pp 15-19.

trying to see what you can learn about one world using the tools of the other– not just in terms of new attacks, but in terms of better understanding of old attacks.

Switching now to the other track, the killer app for commercial crypto was the ATM, and we all understand how this works: you put in the card with the primary account number (which has some of the attributes of a user ID), type in your PIN (which has some of the attributes of the password), and if these match up it gives you some dosh. Now if the ATM is on-line then in theory you can use familiar mechanisms, verify the PIN against an `/etc/passwd` file, and there you are.

But in the old days, because of lousy communications, most ATMs ended up having to work off-line at least some of the time. So there were two ways you could do things. The first was to derive the PIN by encrypting the primary account number, and put the key in the ATM. There's a surprising number of banks that do that to this day. This is something that people are embarrassed about because it means that every bank manager – who maybe has the initialisation keys for the ATM sitting in his correspondence file, perhaps in a box file in the middle of the bank branch – has the means to work out a PIN for every bank customer, and the operation is quite simple and straightforward. We'll look at the implementation detail later.

The second way in which you could do things was to keep an encrypted version of the PIN on the card magnetic strip. The commonest way of doing this used a PIN encryption key that all cash machines know; you take the 4-digit PIN, left-justify, pad it with 12 zeros, encrypt it, bang it on the magnetic strip, and you're done. A slightly more sophisticated way of doing it was to encrypt the account number and the PIN together. But the first method has a protocol failure that led to quite a lot of fraud in Britain about ten years ago.

Matt Blaze: But the first one worked?

Reply: The first one got fielded; it's now ISO 9564 PIN block 3. It has a feature that you can edit the primary account number on your bank card to somebody else's account number, and then take money from his account using your PIN.

ISO PIN block standard 1 is the good one, and was brought in to stop this, but of course didn't cause anybody to change the legacy. By the time it was introduced in about 1985 or so, the majority of the large money-centre banks had already settled with the legacy.

So we end up with a nice harvest of bugs being propagated forward in both communities. And if you look then on the other track, the academic track, we've got all these protocols produced in the 1980s, and essentially they all broke at least once, and a lot of these failures that were found in the academic track involved mechanisms for freshness or for binding. Hence the BAN logic does formal verification of freshness and binding. The way to explain this to the bankers was to say it's an electronic cheque system, you have to make sure you bind together the payee, the drawer, and the amount, and that you link them with some kind of time limit such as a serial number and date in such a way that the cheques can't be altered or duplicated. So there really shouldn't be any rocket science in the BAN logic if you look at it from the point of view of a banker; just take out your cheque book, look at it, contemplate it for a few seconds, and figure out why the cheque has evolved the format that it has over the last eight hundred years. That is a very, very good guide to the design of protocols whether on-line or off-line.

So let's now cross the floor and look at what the bankers were getting up to that was stupid. Well, the bankers spent the 1980s connecting together hundreds of thousands of cash machines into dozens and dozens of networks. It started in America because, in a lot of American states, banks will typically only have one branch. So for the ATM service to be useful, it's always got to either plug into a commercially-operated ATM network, or else share with local banks. How it works is that you put crypto into the cash machine, and there's a series of black-box cryptoprocessors at each bank or switch along the path. At each step the encrypted PIN is decrypted from the key used on the last leg, and it is re-encrypted using the key for the next leg. The idea is that the PIN never appears in the clear in any bank's mainframe, nor do any of the keys involved. And even if you have a double-plus-ungood person working as a programmer at Visa, this person should be able to do no harm because all he's doing is getting cipher text off this link, putting it into the black box, getting fresh cipher text out of the black box, and shoving it out on the next link. There is some reconciliation going on of course, but that's subject to all the usual controls that the banking world has developed and is happy with.

The whole purpose of this is to manage the risk from corrupt bank employees. So there is already a slightly more practical approach, because if we go back to the academic stream, and we look at what went on in the 1990s, we had an enormous number of papers citing the BAN logic. In 1993 I think the BAN logic was the most heavily cited paper in computer science. And there were many follow-on papers, some competitors, and these security protocol workshops that Roger started.

From time to time we thought that we'd run out of protocol stuff to play with, but this didn't happen. More and more real world systems came along, were designed and then broken, and over the course of these workshops we gradually came round to the view that although people talk about the 'Evil Hacker on the Internet', they are actually worried about corrupt insiders. The evil hacker on the Internet eventually got his PhD and now he's got tenure at MIT, so it's a little bit strong to keep on blaming all our research on him. [Laughter] The real spectre at the feast is the dishonest insider.

So how do you then fuse the two strands? Well, two or three years ago I started pointing out that this might be a useful research topic, and I got a student, Mike Bond, to look into it. The threat model looked something like this: you've got the server, and in the server is a wicked person, so you've got a crypto box – a 'security module' – that does all the crypto. So what you have to protect is not the protocol, it's the API – the set of commands the wicked person in the server can send to the security module.

As for how the security module works, many of you have probably seen in the initial API security paper[2], so let's go through all this very quickly. The security module has some master keys in it, and when you want to do crypto you send it a key encrypted under one or other of these local master keys. Which master key you encrypted your working key under tells you what type of key it is: is it a key for encrypting PINs, is it a key for verifying MACs, or what? This in turn determines what you can do with the working key.

As an example, if you want to generate an encrypted PIN, the server sends to the hardware security module a primary account number and a PIN master key, and the PIN master key is encrypted with local master keys 14 and 15, which says it is of type *PIN Master Key.* The security module then sends to the server the PIN – which is the primary account number encrypted under the PIN master key – but encrypted under local master keys 2 and 3. When local master keys 2 and 3 encrypt something they flag it as being of the type *Stored Encrypted PIN.* This transaction is sort of secure by itself, subject to one or two caveats.

However, one of the first spectacular breakdowns that was found, and which was also the first attack on such systems found by the use of formal methods[3], goes as follows. There is a transaction set allowing you to send a *Terminal Communications Key* in clear to the device – this is the key used to verify MACs – and encrypt it under a *Terminal Master Key* – that's a master key that sits in an ATM and is used to encrypt all other keys that are sent to it. The terminal master key is also encrypted under keys 14 and 15, just like a PIN master key. So this transaction returns a terminal comms key encrypted under a terminal master key, and now the interaction between these two protocols (which are secure in themselves) is straightforward. You send to the security module a primary account number saying 'this is a terminal comms key', and you then send a PIN master key encrypted under master keys 14 and 15 saying 'this is a terminal master key'. The security module happily takes the first parameter, and encrypts it under the second parameter, which it decrypts temporarily for the purpose. This is of course, in its first 16 bits, the clear value of the PIN.

So here is a useful case of the composition problem that I referred to yesterday when somebody[4] talked about the composition of two protocols. Here is a composition of two protocols that are instantiated as commands of the security module API. It's instantly fatal to VISA-type security modules.

That is not a new result – it came out in late 2000 or early 2001 – and was part of the first generation of API attacks. However, over the last few months we've generated a second wave of API attacks which focus not just on the way the keys are managed, but on the application level data. This, I think, is going to give an awful lot of fun to an awful lot of people. I'm already seeing derivative works by

[2] R.J. Anderson, "The Correctness of Crypto Transaction Sets", LNCS 2133, 125–141.

[3] M.K. Bond, "Attacks on Cryptoprocessor Transaction Sets", in *CHES 2001*, LNCS 2162, 220–234.

[4] N. Asokan, V. Niemi, K. Nyberg, "Man-in-the-Middle in Tunnelled Authentication Protocols", these proceedings.

other people who have been using these tricks to take apart hardware support for a whole range of shared-key and public-key protocols that are out there.

Now if you put in detail of how PINs are worked out in a bank by encrypting the primary account number to get the PIN, it works as follows: you take the primary account number, you write it as 16 decimal digits. The PIN master key is 16 hex digits, parity adjusted. You encrypt the primary account number with the PIN master key, the output from which we call the raw PIN is written as 16 hex digits. You feed to the device also a PIN length. Assume the PIN length is 4; it then takes off the first four digits of hex and decimalises them. How it decimalises them is by means of a standard lookup table. The great majority of banks simply take the hex digits mod 10. So the decimalisation table is 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5. But the Midland Bank has to be different! For example, if you look at the frequency of digits in Midland Bank customers' PINs, then instead of digits 0 through 5 being twice as common as digits 6 through 9, you find digits 3 through 8 are twice as common as the others, and if you look at eftpos terminals in Denmark, you will see that the digits for 1 through 6 are worn twice as much as the digits for 7 through 0.

Matt Blaze: Surely customers pick their own PIN numbers, I mean, is this a cultural phenomenon?

Reply: It depends on the bank, the bank I use assigns your PIN but you can change it if you feel like it.

Lawrence Paulson: The point is that decimalisation makes some digits twice as likely as others.

John Ioannidis: Only if the bank issues the key, if you change the key, great!

Ross Anderson: I think he probably has a valid point, as the default is pretty common.

Matt Blaze: One explanation for this would be some cultural phenomenon about the numbers that people consider random when they're making them up in their head, and another possibility would be if these were bank-issued PINs that have been decimalised; that that could have a cultural impact.

Reply: There are dozens of papers that can be written on it and work is being done – I mean, looking at wear patterns of ATM keypads is a legitimate scientific activity. [Laughter]

Anyway what happens if the customer can change their PIN is that you've got an offset added to the decimalised PIN. For example, if my bank has been issuing this first off, and then at my first transaction I can change my PIN to, for example, this, and the difference between the two of them is an offset which in off-line systems we write to track 2 and 3 of the magnetic stripe, and in on-line systems, well, it's just on-line.

So one day I'm walking down the corridor in the lab and Mike Bond comes up to me and says, "I've found this new attack on security modules: you can manipulate the decimalisation table", and I say, "yes of course you can, that breaks everything doesn't it". To see why, look at this. This is the version of the attack as I originally, instantly, thought it would be done. What you do is you send a transaction to this security module with a decimalisation table of all 1s, and then the decimalised PIN is going to be 1111, assuming the offset is 0. Then the encrypted PIN that we're going to return, under master keys 2 and 3, is the ciphertext corresponding to the plaintext '1111 0000 0000 0000'. You've now managed to upload a PIN into the system and get a ciphertext out. From then on in it's plain sailing. You can then put in a decimalisation table of 2111 1111 1111 1111, and if the output ciphertext is different, you know there's a 0 in the encrypted account number – in other words in the raw PIN. And depending on how smart you are, it may take you a few dozen tries to work out the complete PIN.

Now the way that Mike had thought of doing this was different, and then somebody else discovered it independently and they also thought a up a different method of doing it, and in fact once you start looking at security module manuals we see that there are many, many, many ways of manipulating the API. For example, as soon as you have one known PIN, you can just track through all the offsets and thus build a dictionary for all ten thousand possible PINs, and all ten thousand corresponding cipher texts. You then download the account master file, and you've got the works. So there are many, many ways of doing this once you've seen it, and the amazing thing about these attacks is that they work again and again and again, against one security module after another, regardless of the manufacturer or the model of the security module. This is unparalleled among previous attacks on cryptographic equipment: that you can code up an attack on the 4758 and it will run on the Thales, it will run on the Atalla, it will run on everything in sight. For practical purposes these machines are virtually bug compatible. [Laughter]

And who's responsible? Well it was IBM, who thought out the scheme back in about 1973 for the 3264, some machine that nobody can remember anymore because it was retired 15 years ago. And it's amazing that you've got such a nasty and pernicious bug which has managed to persist for so long, on so much equipment, all around the world. And now we have the situation where essentially the PIN security of the world's banks depends on the operational controls which surround access (physical and logical) to the server. The cryptographic equipment is performing no function, and if they were doing the encryption in software it would be faster.

And that's absolutely sobering, and it will be even more sobering when some papers currently in the pipeline come out, which show that there are similarly pessimistic results for all sorts of other cryptographic equipment used to support other protocols. If you've got a cryptographic processor, and you think it's doing some work for you, you're probably wrong.

One other interesting thing is that Mike came up with the original idea, and developed it, while the two of us were advising the defence in this case we're not supposed to talk about, and a similar, although not overlapping, set of attacks was discovered by a chap called Jolyon Clulow[5] who was working for

[5] see `http://www.cl.cam.ac.uk/users/jc407/`.

the company that was providing the bank with its expertise in the same case. And there was a gagging order purchased by Citibank, the owner of the Diners Brand, to the effect that if I tell you, on the soil of England or Wales, any of the things that were disclosed during that trial, then I'm a very bad person and I can go to jail for a long time. That's one of the reasons why we rushed these results out as quickly as we possibly could, and saw to it that the technical reports duly appeared on the website before the gagging order was issued.

Matt Blaze: I think whether you are a very bad person is something we can judge independently of the court.

Reply: The curious thing about the ruling was that, when I'm visiting IBM in two weeks time talking to them about this subject, there's absolutely nothing that stops me going into long and gory detail because the court limited its injunction to England and Wales – because of the consideration that if my client lost his case he might end up as a defendant in a criminal trial elsewhere, and so it would be gravely prejudicial to him if the material couldn't be repeated outside the UK.

Bruce Christianson: You are all right north of the border?

Reply: I'm at liberty to tell the gory details of what happened, what was admitted, who dunnit, and the rest of it when I'm in America, but not here – which is strange, isn't it? What sort of due diligence will I have to show in terms of telling a US audience? "Please do not put this on a website where somebody in England or Wales might read it". I don't know, anyway that's for the lawyers.

John Ioannidis: No, it's subtle.

Reply: As I said, there are many, many more variants of this attack, and they're going to have some interesting effects on the commercial world. But here are five scientific lessons I suppose of interest to this community:

Engineering is still leading theory because it's the real-world applications which provide us with the interesting new breaks. When I first thought such an attack might be possible, and in fact described it for our theory lunch back in 1999, I was hoping some of you guys would get your teeth into this. But nobody was interested, in the end it took hands-on hacking of an actual instance of a crypto processor to move the thing ahead.

Second point, perhaps better news to theorists, is that the composition property matters. For years composition was something that caused engineers to yawn and head for the bar whenever it became a topic at Oakland, but here we see that in the real world it is important – and a machine that can do two things with any shared security context is likely to be very suspect indeed.

The third point is that differential analysis now applies to protocols, not just the underlying ciphers, because what we're doing when we feed a large number of slightly varying decimalisation tables into a PIN generation process, and discover those tweaks in the decimalisation tables that don't cause a tweak to the outcoming cipher text, is just the same kind of thing that you do in the differential cyptanalysis of DES: look for input differences that cause no output differences. This is the standard way of looking at a Feistel cipher. And what we have to start bearing in mind is that, in the real world, the bad guy will do the protocol run not just once or twice, but thousands of times with subtly varying input against this piece of kit that you've just designed. Up until now I don't think designers have thought about that possibility very much. This was something for the cryptomathematicians – we're protocol people, we don't do differential stuff! It's the maths department for that stuff.

Fourth, covert channel analysis – this is another thing that caused lots of people to yawn and head for the door, when the few people in the infosec community who obsessed about it start going on. Now this also matters in real life. What happens, for example, if you have a bad person design a security module – so that it halts when given a particular 8-digit PIN, and perhaps outputs the previous customer's PIN? The probability of this being found in real life is as good as nil, but given a modern device that can do a thousand transactions per second, a person in the know can of course easily attack this in an instant with the Balking Channel – the covert channel tied to the Halting Problem that Gus Simmons wrote about[6]. So even if you're leaking only a fraction of a bit of information per transaction, it still can matter[7].

Finally, the next thing for the theory folk is that secure multi-party computation may also suddenly start to come within the reach of protocol analysis. Because what we were trying to do with PIN computation is to perform trusted computation with some untrusted inputs: there are some inputs that the bad guy can manipulate. We all know the theory of how to do that, but this application is in fact a lot more severe than the multi-party computation cases that the theory people have looked at.

So OK, back to the right hand track now, where can we go from here? Well we have a billion cards, and half a million ATMs, in thirty thousand banks, wide open to insider attack. The crypto processors that people thought worked, don't. And there's a lot of counter-productive momentum around. For example, British banks are rushing to introduce PINs to the point of sale so that you will use a PIN when you use your debit card, instead of signing a Switch voucher.

Fixing things is going to be hard because there's a long time lag in getting people to upgrade stuff, maybe four or five years; there's the diversity of platforms; and there's an awful lot of backward compatibility with earlier legacy systems. So what can we do? Well one thing people can do is move to a completely different system like EMV[8], but people have already found API level failures in the EMV protocol suites, so even although this is now being rolled out, it's another of these cases of instant legacy like we talked about in the

[6] G.J. Simmons, "The History of Subliminal Channels" in *Info Hiding 96*, pp 237–256; see also Yvo Desmedt, "Simmons' protocol is not free of subliminal channels" in Ninth IEEE Computer Security Foundations Workshop, 170–175, 1996.

[7] see also M.A. Blaze, "Looking on the Bright Side of Black-Box Cryptography", LNCS 2133, 54–61.

[8] see `http://www.emvco.com/`.

context of mobile phones last night: switch from one insecure system to another insecure system at a cost of eleven figures.

Another possibility, and I think people will come round to this eventually, is that you can completely re-engineer the back end. In other words we keep the infrastructure of security modules, ATMs, and the use of symmetric cryptography, but we redesign the protocols completely in such a way it's incompatible with what there is at the present time, and one way of putting in some incompatibility would be to go to a different size block cipher, like AES. In practice you might use a patented cipher so that you can use IPR to prevent anyone building a security module that handled the new transactions and also the old transactions, because if they did that you'd probably see it fail.

Now there are some possible protocol level fixes for differential attacks. Here's an idea about how you might defeat the differential attack problem. The idea is that with nonces you basically randomise all encryption, so whenever you encrypt something you put a new nonce there, and you never re-use these nonces. So the ATM might send to the enquiring bank a link level nonce, and then an authentication nonce and PIN, which you encrypt and pass along the network to the issuer in the usual way. Then the issuer can send back to the ATM, as a positive authorisation response, this authorisation nonce that we put in the body of the message. We don't have the resources available to compute MACs of all the traffic, so sending out a nonce as the positive authorisation appears to be a cheap and cheerful way to authenticate matters end-to-end. I will not go into that in any more detail; I will be kicking around a number of such ideas and presenting them to some of the people responsible for this.

Let me just wrap up. The seminal papers that we got from Roger in 1978 (the Needham and Schroeder Paper), and the BAN paper in 1989, have led to the emergence of a significant discipline in computer science. What I think we're managing to do with API-level attacks is expand it to cover issues such as composition and multi-party computation, that previously were thought to be different parts of theory not connected with protocols. If, over the next few years, companies like Microsoft, Hewlett Packard, *etc.* are trying to design systems for restricted execution environments – trusted computing, treacherous computing, whatever you want to call it – then this appears to be one of the critical things that you have to get right, and vendors who ignore these issues may well come unstuck.

Bruce Christianson: Ross, I particularly like the idea of treacherous computing instead of trusted computing; I can see a great market for treacherous third parties.

Bruno Crispo: Is your complaint really with IBM, the manufacturer of the 4758, or is it with the author of the software inside the box? So IBM had a bad idea in 1977, that messes up the software inside?

Reply: There is CCA[9] software in the box, but that's also written by IBM.

[9] D.B. Johnson *et al.*, "Common Cryptographic Architecture Application Programming Interface", IBM Systems Journal, pp 130–150, June 1991.

Bruno Crispo: It's the *designer* of the software inside the box that is to blame for all this.

Reply: IBM did that too. The decimalisation table attack is a design error made by IBM, but it was thirty years ago, and if anyone tries to sue them, they'll be told it is too late!

Bruno Crispo: You mentioned EMV for example, that's fine, but it is surely not the vendor of the HSM that must be blamed for that.

Reply: Well the interesting thing is that if there's somebody who got into the HSM business recently, and built an HSM within the last six years, and is therefore available for litigation, perhaps he could end up being liable for a mistake that was actually made by IBM thirty years ago, I mean, who knows the ways of lawyers? Where there's a rich defendant, they'll be . . . [Laughter]

Virgil Gligor: IBM's response to the previous attack was that you need privilege, you require an insider, and . . .

Reply: No, they designed the 4758 specifically to stop the insider getting at the cryptographic keys, that's what hardware crypto processors are for: no insiders means no need for crypto processors.

Virgil Gligor: The original design did not assume insiders, somebody has renamed this equipment who's obviously not the right person.

Reply: I've got material that IBM used to market this stuff twenty years ago, and . . .

Virgil Gligor: I'm not saying that this is unfair criticism, I'm saying that at the time they designed this they did not think about the problem hard enough to be able to come up with better protocols. So my question is different: after seeing this, and they've seen it for a while, if you talk to them after two years, have they changed anything?

Reply: Well with the first lot of attacks, we gave IBM ten months to go and fix it up, before it would appear in print. And those ten months were spent with their Watson labs, which does the hardware, and North Carolina, which does the software, arguing whose fault it was. So in effect IBM did nothing until, thanks to Richard and Mike, there was a slashdot of publicity, and then all of a sudden were there dozens people from IBM downloading our papers and working on a fix to ship to the customers. So this is a very strong argument for openness in security. I concluded, you know, that unless you actually hold the corporation's head down and get them to improve, then they won't actually fix it.

Virgil Gligor: So they fixed this?

Reply: Well they fixed the binding attack, and they fixed one or two other attacks that we found in the previous generation, but this decimalisation table stuff, I don't think it is fixable.

Michael Roe: You can fix the decimalisation table: you can find out the decimalisation table people are actually using, and say, right we're going to put in a patch that says no other decimalisation table is allowed.

Reply: Which is highly unsafe. If you were with a bank that used a non-standard decimalisation table they would no longer serve you? Go into the outer darkness? IBM do not do that. What Thales did was introduce some rules for the decimalisation table. As I recall, it's got to have at least eight different digits and no digit must be present more than five times, something like that.

Richard Clayton: Actually, it's a conceptual problem because if you draw the thing on the board and list the inputs to each operation, and note by the way that we've secured this key, then suddenly it will curse you. Because if you've drawn the right diagram you'll say, well actually you haven't secured the decimalisation table have you, so that ought to be signed and encrypted so bad people can't change it. Well actually the offset shouldn't be changeable, or whatever, so in the end you end up with just the things which are coming from the user which are not secure, and everything else is secure so you do the same computation each time, so that's the real fix. But that means quite a lot of change. You actually change the API, because you have to feed different things into it, you need new functions so that you can generate decimalisation tables and get them signed properly by the box.

Reply: You'd have to get them signed first, or just hack the security module code so that the decimalisation table is always the same regardless of what you feed into it.

Richard Clayton: But you could think of some input which was signed by an administrator in a suitable way.

Reply: No, it will make the transactions too long.

Richard Clayton: Oh yes, "But you asked for security sir." [Laughter]

Reply: You see the fix that Thales has got at the moment doesn't work because it basically says that the decimalisation table should look something like a decimalisation table – it shouldn't be all ones. But that doesn't work: again put your cryptographer's differential hat on and say the decimalisation tables will be 1 2 3 4 5 6 7 N, 1 2 3 4 5 6 7 $N + 1$ ...

Richard Clayton: The tables used in Jolyon's attack looked like pretty normal decimalisation tables.

Bruce Christianson: But so do the vast majority of possible decimalisation tables.

Richard Clayton: Yes.

Reply: So basically, how you actually fix this if you're a bank and you're paranoid right now is to go to your HSM vendor and say: we use only the following transactions, you will comment out everything else; you will nail down that decimalisation table to the following value; we shall never again use an offset; and that's it. In other words, you remove all generality of function from the device and instead of having an off-the-shelf device that costs you $10,000, you've got a custom device that cost you $100,000. Then you will be secure, but you have got all sorts of strategic concerns about using custom kit, and needing specialised knowledge, and so on and so forth.

So it's an unpleasant situation to be in, and there's at least one of the vendors who is responding to this by selling, for extra money, a facility whereby you can turn off some of the transactions. You can download, if you like, a set of access control lists which will cause a number of the transactions to be turned off. So you can pay all this extra money for this feature, and then you presumably have to go and pay them a quarter of a million dollars for consultancy which will tell you exactly which transactions to turn off so that the thing is secure. So this can turn into an enormous moneymaker for the unscrupulous security module vendor, and perhaps in the end the banks will say, ah, to hell with it, we're just going to do the crypto in software.

Wenbo Mao: Would the following way be more secure? Say the trusted hardware module inside has a random generator, it will really follow your instructions saying encrypt this but will randomize it in some standard way, and encrypt it; the result is not what you expected.

Reply: Well as I've intimated, I think you can without significant difficulty design a set of transactions that will perform the same function without any of the known vulnerabilities, the problem is how you roll them out? Visa could say that, as from the 1st April 2005, ACTS will be discontinued in its entirety, there will be no more ISO 8583 standard transactions anywhere in the world, everybody will use the new system from day one, and everybody must move in completely new pieces of security hardware, blah, blah, blah. You know, instant, big bang changeover. Well nobody in the banking world is going to do that. What people are going to want to do is to move from an old system to a new system. There will be procedures for doing this, and penalties, and a bank may be fined $50,000 a month for being late with a security upgrade on a network, for example. So what's going to happen is that some people will move to the new system, and other people will find it inconvenient because their board of directors are barking at them about bringing in a new loan scheme, or a new fancy smartcard, or a didgery doo, or whatever. So they won't migrate in time, and we'll end up being stuck with some banks on the new system, and some banks on the old system, and the banks on the old system who are being fined $50,000 a month will then go to the Visa board and they will vote for the fines to be discontinued on the grounds that they are discriminatory, and you'll end up with an even worse situation than before because all the banks will have security modules that do the old transactions, and the new transactions, with lots of translation between them.

Bruce Christianson: And eventually someone will write a filter...

Bruno Crispo: But the system is that you can run with the old system however long you want, the problem is that we won't cover your loss any more.

Reply: Well the customer covers the loss, that's the public interest issue, and why we should be militant about this sort of stuff.

Virgil Gligor: Ross, you are talking about these wonderful attacks as being real life problems, but you realise the way bank security is broken is mostly through blackmail and intimidation.

Lawrence Paulson: Well I think you need a radical solution to stop this decimalisation bug, and the solution is that people should have hexadecimal PINs. [Laughter]

Bruce Christianson: How about five octal digits? Keep the current keyboard, but don't use 0 or 1.

Addressing New Challenges by Building Security Protocols Around Graphs

Kamil Kulesza and Zbigniew Kotulski

Computer Science Department,
Institute of Fundamental Technological Research,
Polish Academy of Sciences,
ul.Świętokrzyska 21, 00-049, Warsaw, Poland
{kkulesza, zkotulsk}@ippt.gov.pl

Abstract. We propose the use of graphs as basic objects in security protocols. While having all the functionality of their number based counterparts; such protocols can have extended capabilities, especially useful in the field of verification and analysis. The scalability and transitivity for graph related properties allow for addressing protocols of increasing complexity. These features also cater for new challenges in the future, for instance ones resulting from a quantum computing paradigm.

Keywords: security protocols, data security, graph theory, secret sharing, graph colouring, extended capabilities, quantum computing.

1 Introduction

The main goal of this paper is to stimulate discussion on alternatives in security protocol design, which should have a great effect on protocol analysis and verification.

When it comes to cryptology, number theory is king, as it underlies the majority of cryptographic algorithms and protocols. Claude Shannon, in his fundamental Mathematical Theory of Communication ([1]), introduced a description of information as a string of bits. Usually a bit string has been thought of as a binary number. Binary numbers are easy to manipulate, hence this approach underlies virtually all modern computing. Yet, theoretically speaking, binary strings can also represent other objects, like graphs. When they are used to describe the structure of a graph, they actually contain much more information than only about the structure.

In order to benefit from this observation one needs to make practical use of the surplus information. One possibility is to use a conversion between graphs and numbers, for example methods based on Goedel's numbering ([2]). There are other numerous ways to do so (see [3], [4]). When some additional conditions are met, such conversion allows one to use traditional, number theory based cryptographic methods, while also gaining the advantage from graph embedded information. Such opportunity is especially interesting taking into account connections between graph theory and other branches of mathematics like: coding theory, topology and knot theory (see [5]).

Graph theory provides us with a variety of interesting problems, many of which are known to be of NP class (*e.g.*, [3]). Examples are: graph colouring, graph isomorphism and Hamiltonian paths, all of which can be used to provide extended capabilities for security protocols.

One should remember that problems from the same complexity class can be used interchangeably, at least in the graph case. Very good examples are zero-knowledge proofs, that were designed independently for various graph problems, for instance isomorphism (*e.g.*, [6]) and colouring (*e.g.*, [7]). In [6], Goldreich *et al.* have shown that any NP-complete problem can be turned into a zero-knowledge proof. Hence, the choice of problem to work with is only a matter of individual taste.

On the other hand, the above mentioned problems are handy when it comes to the issue of provable security (*e.g.*, [8], [9]). In this paper we wish to concentrate on the most promising field: verification and analysis of the protocols in the provable security context.

In security protocols the role of the graphs has mainly been limited to modelling the data structures. Although significant results have been obtained, with the proposed approach we hope to go much further.

The goal of this paper is to discuss the advantages of using graphs as the building blocks for security protocols. In Section 2 we briefly describe an example of graph colouring based verifiable secret sharing (VSS). Section 3 is meant to stimulate discussion on the interactions between graph object(s) that underlie the protocol and graph based data models. At the outset we outline two graph related properties: scalability and transitivity. Next, we discuss how they can be applied to handle increasing protocol complexity. At the end of the section, we briefly refer to problems resulting from quantum computing methods. The conclusions are provided in Section 4.

2 Graph Colouring Based Verifiable Secret Sharing

In this section an illustrative example, from research into Verifiable Secret Sharing, will be presented. First, we provide some background information on secret sharing.

Everybody knows situations, where permission to trigger certain action requires approval of several selected entities. Equally important is that any other set of entities cannot trigger the action. Secret sharing allows a secret to be split into different pieces, called shares, which are given to the participants, such that only certain groups (authorized sets of participants) can recover the secret.

The access structure is the set of all authorized subsets of participants. Good general references for secret sharing are the books by Stinson [10] and by Pieprzyk *et al.* [11]; an interested reader can also consult the bibliography list by Stinson [12].

Once secret sharing was introduced, it was found that it can be easily compromised by misbehaving parties. Hence, the ability to perform secret consistency verification and detection of cheaters is very important. One of solutions is to use Verifiable Secret Sharing (VSS). The verification capacity usually comes at a price. This fact is related to the paradox stated by David Chaum, that no system can simultaneously provide privacy and integrity.

At ESORICS2002 in Zurich, a verification method that works for any underlying secret sharing scheme was described ([13]). It is based on the concept of verification sets of participants, related to an authorized set of participants. The participants interact (with no third party involvement) in order to check validity of their shares, before they are pooled for secret recovery. Verification efficiency does not depend on the number of faulty participants. One of the pillars of the method is the use of a proper verification function; a very promising one results from the graph colouring check-digit scheme described in [14]. This proposal requires conversion of the given number into a graph and checking its vertex colouring on both sides of the communication channel. The quantitative argument presented shows that the feasibility of the proposed scheme increases with the size of the number whose digits are checked, as well as, overall probability of digits errors.

Joining both results ([13], [14]) produces a graph-based shares verification method. The method depends heavily on graph colouring properties that in turn are handy in the formal security analysis. To some extent it seems even to bypass (or at least weaken) the Chaum paradox. In the case of described method one does not get a free lunch, but at least can have a free starter.

3 Advantages of Graph Based Security Protocols

Before entering into a discussion of what graphs can do for security protocols, we need to outline two graph related properties:

a. Scalability. The fastest informal description is that the level of complication in a graph increases exponentially with the number of vertices, while testing for basic graph properties can be usually done in polynomial time. Very good instances are bounds on chromatic numbers (*e.g.*, [3], [15]).

   Example 1.
   Definitions:
   The degree of vertex in the graph is the number of edges connected to that vertex.

   The complete graph is the graph, such that every vertex is linked with all remaining vertices.

   The odd cycle is a connected graph, having odd number of vertices, with each vertex having degree equal to 2.

   Theorem by Brooks (1941): Let G be a connected graph. If G is neither complete nor an odd cycle, then its chromatic number is smaller or equal to the maximum degree of vertex in G. •

   Scalability is used in the graph colouring check-digit scheme (see [14]). It also works well in formal analysis of access structures in secret sharing protocols. If the graph representing an access structure belongs to some instance graph theoretical class, specified information theoretical properties of the secret sharing method are conserved (*e.g.*, [10]).

b. Transitivity (proliferation). Given any graph, even a small alteration made to the structure (or the way the graph property is described), may have far reaching consequences. This can be understood figuratively and literally – resulting changes can pop up far away in the graph. Again, graph colouring is a very good example. Even a single vertex or edge modification made to the structure can dramatically change colouring (for an example see [4]). On the other hand, for a given colouring of a graph, alteration of one vertex colour can proliferate through an entire system (*e.g.*, [4]).

By its nature, scalability is very useful when managing complexity. This may even start at the protocol design stage. The example given in the Section 2 falls into this category. In this case, the access structure design can be refined, until it represents the best quality in the information theoretical meaning. We present an example coming from the book by Stinson ([10]).

Example 2.
Let's denote:
E as set of edges for the given graph G;
V as set of vertices for the given graph G;
$cl(E)$ as the closure on the set E.
Definitions:
The complete multipartite graph, such that vertices can be partitioned in the finite number of disjoined sets such that:

a. vertices in one set are not linked be the edge;
b. each vertex is linked with all the vertices outside the set.

A secret sharing scheme is ideal if its information rate is 1, so the length of the secret equals to the length of a share held by a participant. The information rate is defined in the Shannon sense (see [1]).

Theorem: Suppose G is a complete multipartite graph. Then there is an ideal scheme realizing the access structure $cl(E)$ on participant set V. •

Yet, more interesting applications of scalability arise when dealing with analysis and verification of the protocols. When the protocol is a complex collection of the interacting parts (or even protocols), there are two basic techniques: cryptoanalytic and formal. Unfortunately both schools rarely interact (*e.g.*, [16]). Scalability, together with the application of graph based protocols, offers an opportunity to reconcile both approaches to the protocol analysis and verification.

The underlying (graph based) parts would be treated in the cryptoanalytic style, while their interaction would be investigated on a higher level. Both levels of analysis would deal with objects of the same type. Hence, merging them into one, more abstract, construct should be possible. On the other hand, use of scalability would facilitate good information theoretical description of the resulting construct.

The second property addresses the issue of how small changes in one place can have an effect on the whole system. This is a handy add-on for the approach outlined above. Adding transitivity into the picture would address the problem of losing the details when trying to abstract some concepts. This natural process has proven to be dangerous in the analysis of the protocols. The introduction of the transitivity allows for taking care of the details that should not be lost during a shift to higher levels of abstraction. Hence, in the analysis of security protocols, transitivity should be used together with scalability, compensating for its possible weaknesses.

A graph based approach provides capabilities for handling potential problems resulting from quantum computing methods. Once quantum algorithms (or protocols) are embedded into the system, apart from the normal considerations, one should take care about resulting quantum effects, like interference or entanglement (*e.g.*, [17]).

Informally speaking, entanglement means that the properties of a composite (quantum) system, even when the components are distant and non-interacting, are linked. In general they cannot be fully expressed by descriptions of the properties of all the component systems. Hence, the system is more than only the sum of all components.

Transitivity seems to be very well suited, at least partially, to addressing issues resulting from the quantum entanglement. It is noteworthy that the same description as for entanglement fits problems in vertex colouring of the graphs.

4 Conclusions

We advocate using graph based security protocols. They result in the situation where one abstract type simultaneously underlies and models the protocol. This, in turn, allows one to see interaction of the protocol parts in a new light. The example from graph colouring and secrets sharing was used in order to illustrate the proposal, as well as to stimulate the discussion. Nevertheless, the graph theory provides much more opportunities. In general terms the most important is the chance for a unified approach to protocol analysis and verification, and also for the emerging quantum computing paradigm.

We hope that graph based protocols will bring new insight into the way that complex systems interact. Other opportunities arise from investigations into the interactions between graph theory and other branches of mathematics (*e.g.*, [5]).

Acknowledgement

The article was completed during a visit to Rhodes University, Grahamstown, South Africa. The authors wish to thank Prof. Patrick Terry for reading the manuscript and for his critical remarks.

References

1. Shannon C.E.: *A mathematical theory of communication.* Bell System Technical Journal, **27** (1948) 379–423
2. Goedel K.: *Uber formal unentscheidbare Satze der Principia Mathematica und verwander Systeme, I.* Monatshefte fur Mathematik und Physik, **38** (1931) 173–198
3. Korte B., Vygen J.: *Combinatorial Optimization, theory and algorithms.* Springer-Verlag, Berlin Heilderberg New York (2000)
4. Molloy M., Reed B.: *Graph Colouring and the Probabilistic Method.* Springer-Verlag, Berlin Heilderberg New York (2002)
5. Beineke L.W., Wilson R.J. (Eds.): *Graph connections; relationships between graph theory and other areas of mathematics.* Oxford University Press, New York (1997)
6. Goldreich O., Micali S., Wigderson A.: *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems.* Journal of the ACM, **38**(1) (1991) 691–729
7. Brassard G., Chaum D., and Crepeau C.: *Minimum disclosure proofs of knowledge.* Journal of Computer and System Science, **37**(2) (1988) 156–189
8. Goldwasser S.: *The search for provably secure cryptosystems.* In: Pomerance C. (Ed.): Cryptology and Computational Number Theory, Proceedings of Symposia in Applied Mathematics **42** (1990) 89-113, American Mathematical Society
9. Maurer U.M.: *Information-theoretically secure secret-key agreement by NOT authenticated public discussion.* In: Fumy W. (Ed.) Advances in Cryptology Eurocrypt '97 , LNCS, Vol. 1233, Springer-Verlag, Berlin Heilderberg New York (1997)
10. Stinson D.R.: *Cryptography, Theory and Practice.* CRC Press, Boca Raton (1995)
11. Pieprzyk J., Hardjono T. and Seberry J.: *Fundamentals of Computer Security.* Springer-Verlag, Berlin Heilderberg New York (2003)
12. Stinson D.R., Wei R.: *Bibliography on Secret Sharing Schemes.* Webpage: http://www.cacr.math.uwaterloo.ca/ dstinson/ssbib.html. Retrieved 30.08.2003.
13. Kulesza K., Kotulski Z., Pieprzyk J.: *On alternative approach for verifiable secret sharing.* ESORICS2002, Zurich. Submitted for publication, available from IACR's Cryptology ePrint Archive (http://eprint.iacr.org/) report 2003/035
14. Kulesza K., Kotulski Z.: *On graph coloring check-digit method.* Submitted for publication, available from the Los Alamos National Laborartory e-print (http://arxiv.org/abs/math.CO/0211317) (2002)
15. Wilson R.A.: *Graphs, Colourings and the Four-colour Theorem.* Oxford University Press, New York (2002)
16. Ryan P.: *Open questions.* In: Christianson B. *et al.* (Eds.): Security protocols, LNCS Vol. 2133, 49–53, Springer-Verlag, Berlin Heilderberg New York (2001)
17. Gruska J.: *Quantum Computing.* McGraw Hill, New York (1999)

From Security Protocols to Systems Security

Brian Monahan

Trusted E-Services Lab, HP Laboratories,
Filton Road, Bristol, BS34 8QZ, UK
brian.monahan@hp.com

Abstract. When applying information security, we need to go beyond the analysis of individual security protocols and consider how they are used within distributed systems, software applications and services. Important as they are, security protocols only form a part of the overall security engineering design for a particular distributed system. The effective use of any security protocol will typically depend upon certain structural data such as key information being *available* for use by some - and at the same time made unavailable to others. Systems need to be designed with requirements like these in mind ([1,2]).

1 Introduction

This report describes work-in-progress that suggests extending the theory and analysis of security protocols towards a modelling approach for describing, analysing and algebraically simulating security aspects of systems.

We would argue that this involves using a software-supportable notation specifically designed for use by security trained professionals to design and explore the security aspects of systems. To this end, we are investigating an experimental notation, called here Systems Security Modelling Language (or SSML) to embody these systems designs. A variety of tools could be contemplated for investigating the security consequences of these designs within particular usage scenarios, to check resistance to particular attacks and to formulate specific trace and state properties which capture more precisely the security-related aspects of a system design.

Such an approach allows the system design, together with its security-related properties, to be iteratively evolved and pragmatically developed as a series of descriptions. The purpose of this evolution is to establish by pragmatic means a certain level of insight and confidence in the resulting final description. At that stage, it should by then be much clearer what the security-related properties of the design have to be and how each part contributes to establishing the security goals of a system. By evolution of a design, we mean a process of exploration and extension of designs, each of which better capture intention and functionality.

In addition to the system's security goals, the design development should have established what the initial assumptions are and have firmed up what structural data needs to be available at each processing stage. Even so, it is only entities contributing somehow to the security aspects of the system that need be modelled - the design needs only to be *sufficient* to establish the security case. Other entities that don't affect or somehow influence the system's security case need not be included in the model. In fact, once some entity has been shown to be irrelevant to the security case, the model could be simplified by removing that entity. Thus, security designs tend not to be completely definitive in every respect, but must be sufficient to capture the overall security situation that the system will find itself within.

To recap, we gain some "lightweight" assurance, via the tools supporting the security-modelling notation, that the evolved design achieves certain overall goals and functionality relevant to security. To gain a stronger degree of assurance and confidence in the design, it may be necessary to formally describe the *kernel* of a system in terms of mathematical logic and then formally prove that the most important security-related properties hold for that kernel.

However, constructing these formal specifications and proofs is a subtle and complex task, requiring significant logic-modelling skills. Furthermore, this will only be typically attempted for a stable system kernel where a highly assured verification of security and correctness is required. As the design has already been captured in the SSML modelling framework and has hopefully been matured into something already having a high level of confidence, this could provide an excellent starting point from which to extract a logically precise formal specification of a suitable system kernel. This formally represented version of the system kernel provides the basis for formally showing the security-related properties claimed for it, perhaps using tools for theorem-proving [3] or symbolic model-checking [4].

Most of the remainder of this report discusses aspects of an experimental security-modelling language and accompanying tools by means of a short example. Before that, we briefly look at protocol descriptions (i.e Message Sequence Charts) and then consider approaches to the security analysis of protocols.

1.1 Protocol Description Notation – Message Sequence Charts

Security protocols have typically been described using the Message Sequence Chart (MSC) notation. For example, Fig 1 contains a classical MSC description of the well-known Needham-Schroeder public key authentication protocol.

```
A --> B : {A, Na}Pub(B)

B --> A : {Na, Nb}Pub(A)

A --> B : {Nb}Pub(B)
```

Fig. 1. Excerpt from the Needham-Schroeder Public Key Authentication Protocol.

This intuitively conveys what the message flow is and broadly what each principal needs to do at each stage of the protocol. However, as recognised by

Abadi in [5], this does not adequately document certain important aspects of the protocol such as:

- The goals of the protocol and what each participant expects to be jointly achieved.
- The initial assumptions made by each principal.
- What parts of each message were freshly generated and by whom.

Lowe showed in [6] that the particular protocol illustrated above is possible by essentially allowing the attacker to be an 'insider' (i.e. have the same access to authentication keys as the honest principles). This permits the attacker to perform a classic middle-person attack. However, as pointed out in [7], the original protocol design assumed that attacks would be mounted by external agents. This nicely illustrates how flaws in complex systems can arise, even for the best designed systems - a subsystem intended for one set of circumstances may eventually get used in another set of circumstances unintended or foreseen by the designers, leading to inappropriate behaviour. This further suggests that complex, distributed systems may need to be adaptive or evolutionary in their design, if they are to be successful in a changing environment.

The variation of a protocols' initial assumptions may easily lead to different attacks, and this was illustrated in [8] using an automated protocol analysis tool. This emphasises how security in complex, distributed systems depends upon context and the manner in which protocols are deployed and used.

1.2 Approaches to Security Protocol Analysis

There are currently two main approaches to the attack and analysis of security protocols. Both of these approaches start from the classic Needham-Schroeder paper [1] on authentication in large networks. This landmark paper contains many of the basic principles and underlying issues of security protocol analysis from which all modern work has flowed.

Probability and Complexity-based Analysis. In the probabilistic and complexity-theoretic approach favoured by cryptographers (e.g. [9]), the attacker tries to exploit protocol interactions in some way so as to reveal covert information about the underlying crypto primitives - the objective is to break the crypto primitives by subversion or by extracting key information. Whichever way is taken, if successful, the attacker would achieve full control.

The outcome of such an analysis is typically some estimate of the *work factor* required by the attacker in order to achieve their goal. This work factor may be the probability of extracting useful information, expressed in terms of a function of a given number of trials (i.e. attack runs) or it may be the number of trials required to extract particular information. A typical feature of this type of analysis has been the use of the Random Oracle assumption in which hash functions are modelled as functions randomly chosen from a uniformly distribution.

Algebraic and Formal Logic-based Analysis In the algebraic and formal logic-based approach, begun by Dolev and Yao [10], and continued by many researchers e.g. [11,3,5,12,13,14], the attacker tries to gain *reward* or *advantage* by playing different protocol players off against other protocol players, each of which may be using differing protocols. This usually involves some deceptive use of an honest principals role, where the attacker can exploit some inherent ambiguity within the protocols. Relative to given keys, the attacker can analyse and synthesise arbitrary messages, based upon what the attacker explicitly knows and subsequently uncovers. However, the attacker is only permitted to do cryptographic operations (e.g. encryption, signing) using those keys it has got *explicit* access to.

Process algebraic and trace theoretical approaches to protocol security have played a fundamental role in defining many important security concepts [15,5,3,16,12,17,18,19,20,21,22,23], such as determinism, non-interference and correspondence properties. Security-related properties via typing within the Spi calculus framework have been investigated in [15,24,25]. The Multiset Rewriting approach is another emerging formalism for cryptographic protocols [18,26]. A good recent tutorial survey and overview for this entire research area maybe found in the proceedings of FoSAD 2000 [27,28,29,30,31,32].

Reconciliation of Approaches. Abadi and Rogaway began in [33] to draw these approaches more together, emphasising that assumptions of each model have corresponding assumptions in the other. Further work in combining these approaches has been actively pursued by Pfitzmann and Waidner [34].

It is also known, however, that the analytic techniques dealing with security protocols mentioned above have their limitations. For example, it is widely acknowledged that the BAN logic approach does not adequately capture causation and emphasises what is known (*believed*) by the principles involved. Equally, there are digital signature schemes that have been proven secure within the Random Oracle approach, but where every implementation is equally insecure in their sense [9]. Proofs of security for a protocol only typically give *partial* assurance concerning integrity.

It is clear that currently no *single* approach to security protocol analysis has yet emerged that adequately captures the full richness and diversity of the way that systems are used and the security threats they face. This suggests that further work is needed to combine the efforts across several approaches when analysing and assessing systems security.

2 The Classic Web Service Example

We now give the example we use to motivate the security-modelling notation. This example is a simplified variant of the well-known "single sign-on" problem, depicted in the form of a Web service. In the informal description, there are three distinct phases of operation:

Registration: Client registers and joins the Web Service, by interacting with the Registration Service.

Let S be a Web service offering some online services over the Internet. Each potential customer C must firstly register their membership to be able to access this service in future via a registration service RS.

Following registration, C gains access to the service by fully proving their membership at the beginning of a session. Customer C then receives a time-limited token (a "cookie") that permits re-admittance to S quickly without going through the full access dialogue. Once logged in, C has a set of access privileges allowing them to retrieve and/or update information.

The appropriate informal use-cases are diagrammed here:

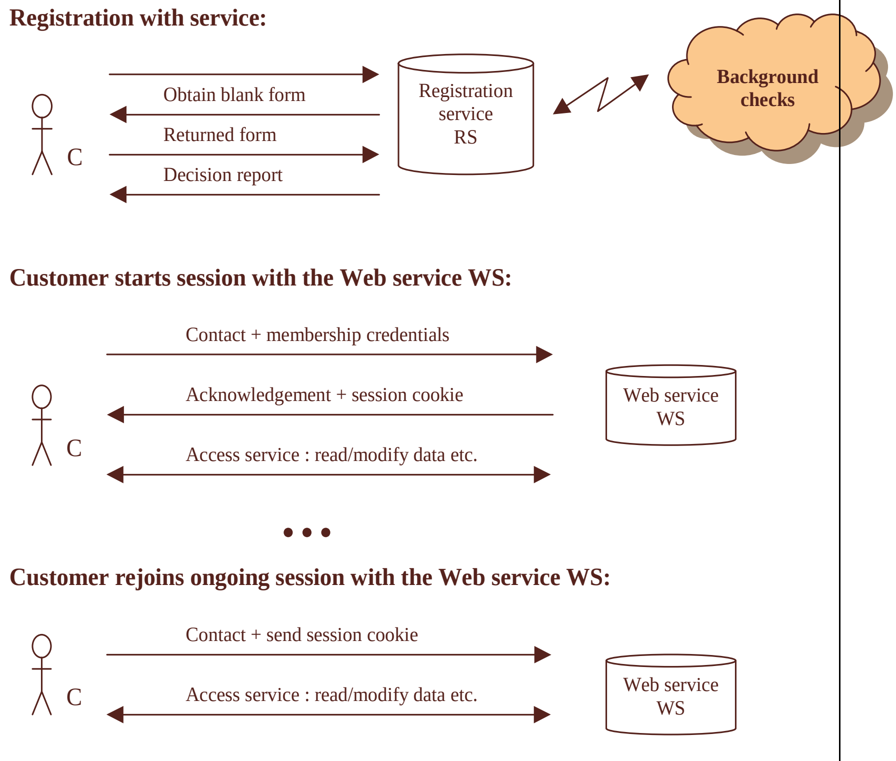

Registration with service:
Obtain blank form
Returned form
Decision report
C
Registration service RS
Background checks
Customer starts session with the Web service WS:
Contact + membership credentials
Acknowledgement + session cookie
Access service : read/modify data etc.
C
Web service WS
Customer rejoins ongoing session with the Web service WS:
Contact + send session cookie
Access service : read/modify data etc.
C
Web service WS

Fig. 2. Informal description of the Web Service

Start a new session with Web Service: Client has to prove their membership in order to initiate a long-lived, but interruptible, session with the Web Service. The Client is given a time-limited token (a cookie) that allows them to rejoin the session later on.

Continue an existing session with Web Service: Client continues an existing session by sending the cookie back to the Web Service. If the cookie has time-expired, the session is closed.

The example given in Fig 2 contains a number of ambiguities and undefined notions that are typical of requirements descriptions. By their nature, requirements descriptions are necessarily informal - they contain the “germ” of the need to be fulfilled by some (designed) solution. The act of design itself involves imposing some order and structure to create a space of potential solutions, some of which can hopefully fulfil the spirit of the requirement that was originally expressed informally.

This example was chosen because it is somewhat typical of an applied service occurring naturally in its “raw” state, rather than something pre-packaged that is already well understood by the security protocols community, such as Kerberos.

A partial answer to that question is to provide ways for exploring and investigating security designs. Our approach is to capture the security aspects of a design in terms of a model that can be analysed in an effective way, supported by software tools. This approach involves capturing the *transactional behaviour* of the design in terms of structured data, events and so on. The purpose of any design modelling activity is to gain insight and establish confidence in the content of the design. It is important to try and keep the model as simple as possible - while retaining its essential relevance.

3 SSML – A Notation for Security Modelling

Our notation, System Security Modelling Language, allows us to define entities representing systems in terms of their behaviour. Although SSML is superficially close in appearance to well-known programming languages (such as Java and Standard ML), its semantics does owe something to the process calculus tradition of CSP and CCS. Further discussion of the semantics underlying SSML is beyond the scope of this paper.

The intention is to capture aspects of the structural features of a security design together with pertinent aspects of dynamic behaviour. Because we are focussing here on security aspects, we choose to emphasise certain details and de-emphasise others. For example, we de-emphasise fine-grained conditional branching and case analysis, but focus instead upon the use of keys, encryption and so on. Such aspects, although de-emphasised for security analysis purposes here, are of considerable interest and importance, but at other stages of systems design and development.

For lack of space in this report, we are unable to present a full account of either the notation, tools or of the complete example.

There are three main groups of entity that make up an SSML description. These are:

1. **Contexts and types:** These entities represent structured data types and are also used to represent the state-space of each principal item that makes up a system.
2. **Events, methods and functions:** Each event is located at an instance of some context representing a principal and typically consists of straight-line

code with no branching i.e. neither conditionals nor case analysis. Events can optionally receive an input, send an output, do both or even neither. Methods generally represent state-modifying operations that may yield a result. Pure functions can be specified algebraically in terms of conditional equations.

3. **Traces and system behaviours:** Traces are finite sequences of events, the message exchanges that are accepted and the state transitions that arise as a result. A system is defined in terms of (accepting) finite sets of traces.

We now move to discussing our Web Service example and illustrating the features mentioned above. In our design, the state-space for the Client, the Registration Service and the Web Service themselves will be represented in terms of appropriate *contexts*. Associated with these contexts will be a number of located events that represent some part of the interaction with another principal. These events process the input and output messages between principals. The overall *system behaviours* linking each of the Client's, the Registration Service and the Web Service are then defined in terms of traces.

We continue below with a more detailed description of the three main elements of SSML: contexts, events and systems behaviours.

3.1 Contexts

Describing complex systems involves structured data, and rather than involve full-blown objects and their semantic complexity (e.g. inheritance, overriding, etc.), we introduce **contexts** to represent this. A context is a collection of named, typed entities. An example is given in Fig 3 to define the Client's context.

The context consists of a number of named fields, followed by an (optional) constraint part. Each field has a couple of classifiers. The first classifier is one of **fixed**, **volatile** or **persistent** - which briefly indicates how the field is used and initialised (or not). The second classifier is one of **visible**, **internal** or **secret**, and this determines security visibility characteristics. These classifiers are explained further in Appendix B. Each field is strongly typed - the types `Identity`, `Asymkey`, `Nonce`, `String` and `Uint` are built-in (there are others). The types `String` and `Uint` are conventional and stand for the type of strings and unsigned integers, as you would expect. The other types represent security-relevant *roles*.

For example, values of type Identity represent "standard" names for entities - in practice this might be a string containing a URI or some other systematically defined moniker. If A is a well-typed value, then ident A is a value of type Identity. To be clear, Identity values are plaintext address-like references-they have no inherent security characteristics. We show later how these identity values can be used to access useful publicly visible information - such as public keys - bound to these identifiers. Values of type Asymkey and Nonce represent asymmetric keys and nonces respectively. The user-defined type RegForm is not further specified but is intended to represent a Registration Form of some kind (typically implemented in terms of XML or HTML). The user-defined type

```
context
  Client  =  {
    fixed
      visible  String   name;       // Client's name (not an Identity)

      visible  Asymkey  custK;      // Client's public key
      secret   Asymkey  privCustK;  // Client's private key

    volatile
      internal RegForm form;        // Registration form
      internal Nonce   n;           // A nonce value
      internal Cookie  ck;          // A "cookie" value, for Web Service

    persistent
      internal Identity registerID; // Registration Service Identity
      internal Identity serviceID;  // Web Service Identity

      internal Uint mem  = undef;   // Membership Number

    constraint
        (custK keypair privCustK);  // Key relationship
  }

type RegForm = unspecified; type Cookie = unspecified;
```

Fig. 3. Defining the Client's context

Cookie represents cookie values that are exchanged between Client and the Web Service and is not further specified. The order in which material is introduced is not significant - in general, there should be no cyclic dependencies between definitions.

Finally, note how the constraint part states the relationship between the customer's public key custK and their private key privCustK in terms of the binary key-pairing relationship, **keypair**. In general, this relation is regarded as being effectively intractable - even though it is classically definable. This relation is statically restricted to occurrences within verification predicates like the constraint part here - it is not permitted elsewhere.

3.2 Events

The dynamic behaviour of a security design is given in terms of events. Basically, an event represents a small chunk of state-dependant behaviour, and can input or output messages using the anonymous broadcast communication model, discussed further in Appendix B. Events are then strung together to form traces - where sets of traces describe the dynamic behaviour of a system.

```
event
  [Client A] acceptMembership {
    Identity regID, rID, cID;
    Binary encData;
    Nonce nA;

    input ["registered", regID, encData];

    [nA, rID, cID, A.mem] =
      decrypt ( encData ) with A.privCustK;

    check nA == A.n  &  rID == regID  &  cID == ident A;
  }
```

Fig. 4. Defining the acceptMembership event for the Client.

A simple illustrative example of an event is given in Fig 4. This event is located at instances of a Client context - as specified by the "[Client A]" part. This effectively allows the event access to all of the components of its context state, here an instance of Client, during its execution. What the above says is that an input message can be accepted if it has the specified "shape" - it looks like: ["registered", regID, encData] This message has three components - the first being the string literal "registered", the second is an Identity value assignable to regID and the third is a Binary value assignable to encData. The effect of a successful pattern-match is to match the string literal and to overwrite the two variables.

The second statement says that, using Client A's private key privCustK, we can correctly decrypt the encData Binary value to give a message containing four elements [nA, rID, cID, A.mem], where nA is a Nonce value, rID and cID are both Identity values and A.mem is a field of type Uint associated with Client A (go back to Fig 3 for the definition of mem). Any previous values in the variables are ignored and overwritten if the whole statement successfully matches, with failure otherwise. Thus, partially successful matching does not cause any effective corruption.

Finally, the check statement checks the logical predicate to see if it is true - with failure if not. In this case, we check that:

1. The encrypted nonce nA is the same as A.n, a nonce sent previously and retained by Client A.
2. The identity rID is the same as regID, the identity quoted unencrypted in the main message.
3. The identity cID is the same as ident A, the identity corresponding to this Client.

An event can have at most one input statement and at most one output statement - typically, an event will only use either an input or an output.

```
event
  [StateContext ST] eventName {
    statements;

    input message;

    statements;

    output message;
  }
```

Fig. 5. General form of an event.

The general form of an event is given in Fig 5. In effect, events are (partial) state functions that may (optionally) input or output messages, while performing a constrained side effect upon the specified state context. If an event inputs a message, this may be accepted, refused or rejected. Rejection means failure, whereas refusal means the message could be accepted by some other event. Events are given by simple straight-line pieces of code, having no recursion, no loops, no case/switch statements and no (explicit) conditional branching via if statements familiar from programming languages like C and Java.

Typically, the input statement acts like a guard, allowing the selection of events that match the current message - they are different from the check statements mentioned earlier that are used to reject, thus yielding an (unrecoverable) failure. Other statement forms available include traditional assignment in both conventional and pattern-matching variants. Data operations in the form of pure functions can also be specified by conditional equations, and this includes recursive definition and conditional evaluation. However, we do not encourage this level of detailed description, since the intention is not to completely define all functional aspects of a system's behaviour, but to just focus on matters impinging upon security.

As described here, these events may appear somewhat isolated and unrelated to each other. What will link events together are system behaviours - and these are described in the next section.

4 System Behaviours

As mentioned earlier, we specify the behaviour of security systems, consisting of a number of different principals or players, in terms of sets of traces - where a trace consists of a finite sequence of located events. As an example of this,

```
system RegistrationService (Client C, Registry RS) {
  init C.registerID == ident RS;

  ( [C]  sendHello;
    [RS] receiveHelloSendForm;
    [C]  receiveForm;
    [C]  returnForm;
    [RS] acceptProcessForm;
    [C]  acceptMembership
  )
}
```

Fig. 6. System behaviour corresponding to the Registration Service

we define the (intended) system behaviour corresponding to the Registration Service in Fig 6.

Corresponding to each sequence of events, there are also the sequences of messages that are output and subsequently input. Accordingly, the above is essentially a (partial) function from the participants (i.e. specified by the Client and Registry types) satisfying the init predicate to potentially accepting traces of the system. If we instantiate this by giving particular instances of the Client and Registry, we obtain a sequence of events - which may or may not succeed. Thus we recognise that there could be instances that, although having the above form, will nonetheless lead to non-acceptance (rejection) due to failure.

A system description like the above is considered to be (weakly) well-formed if there is at least one accepting trace for some choice of arguments, and strongly well-formed if there is at least one accepting trace for every possible choice of arguments. In effect, strong well-formedness says that the system's description forms a total function.

There are other useful trace operators in addition to sequential composition $(T_1; T_2)$, such as simple choice $(T_1|T_2)$ and concurrent merge $(T_1||T_2)$. These are useful in defining complex trace sequences used in system behaviours and in trace predicates (equivalence and containment).

In the next section, we briefly consider how designs in SSML could be analysed using static analysis concepts and ideas from algebraic simulation.

4.1 Analysing the Design

The purpose of a security analysis of a design is to determine if the design achieves certain security goals. Such goals are typically stated in terms of high-level concepts such as confidentiality, authentication, non-repudiation, non-interference and so on. Although such concepts can be mathematically defined in terms of trace, state and correspondence properties (e.g. [19,5,12,3]) translating these into precise, verifiable statements that are pertinent to particular designs is not so straightforward.

For us, analysis involves two distinct phases:

- **Checking of static integrity:** The objective of this phase is to ensure that the design satisfies certain static integrity constraints - and if not, to determine diagnostic information aiding the user in fixing any violations. The analysis involves techniques such as type-checking the usage of data items and flow analysis/abstract interpretation to ensure that information flows are suitably constrained [35,25,36,37].

  Important aspects of this checking includes:

  1. Checking that system designs can accept some traces - a system is malformed if it ends up rejecting every trace (i.e. it has no behaviour of interest). This involves checking that message flows are feasible - that for a message to be accepted as input by an event, it must have been output by a preceding event. Additionally, the volatile fields must have been initialised before they are used in each possible run. However, it is also clear that these trace acceptance conditions could well involve general constraint satisfaction and hence need theorem-proving support.
  2. Calculating the static set of valuable items as used by the system - for example, these are entities that are declared secret or only transferred solely in an encrypted form and not deliberately exposed. More subtly, this also includes indirectly valuable items because they can be used to gain access to other material already known to be valuable.

     In this fashion, we can detect when a system design makes certain kinds of security exposures. In addition, the set of valuable items represents those items that could form targets for attackers and thus need to be protected.
- **Exploration of dynamic behaviour:** System behaviour are defined here by sets of traces – where the traces include state transitions and the messages sent and received. The objectives of this phase are to examine, explore and where possible verify the dynamic characteristics of a system in combination with an environment. For us, an environment is a system corresponding to "the rest of the system" (i.e. the context) in which the system of interest operates.

  In particular, the attacks come from the environment – a successful attack being any coordinated set of actions controlled by the environment that provokes the entire system (i.e. environment and system together) to enter a bad state or to perform an illegal act.

  This type of analysis can usefully be viewed as exploring whether the system or the environment "wins" a certain contest between them – the environment "wins" if it can obtain reward and advantage in some systematic way, and the system of interest "wins" if it always defeats the best efforts of the environment to cause it to fail - i.e. the system wins if it doesn't lose.

We intend to use algebraic-based simulation and manipulation techniques to examine this behaviour and investigate the security of certain combinations of environment and system.

The CAPSL Integrated Protocol Environment described in [38], has been developed by Jonathan Millen and his team at SRI. This appears to be currently one of the most extensively developed security protocol analysis tools. CAPSL[1] provides an algebraic notation and tools for protocol analysis, which are based upon an algebraic rewriting logic. Although CAPSL's protocol notation essentially provides an algebraic form of Message Sequence Charts, it also provides ways to precisely state the protocol goals and initial assumptions for each principal. CAPSL descriptions are firstly translated into an intermediate logic-based form called CIL (CAPSL Intermediate Language) [39] which is in turn related to the MultiSet Rewriting formalism [18]. CIL is principally used to provide a canonical protocol representation, allowing CIL descriptions of protocols to be exchanged between a variety of algebraic formal analysis tools, such as PVS and Maude.

The CAPSL translator tools are currently available for Unix platforms.

5 Conclusions and Further Work

The purpose of this paper is to motivate interest in security-focused systems analysis and the need for security modelling and simulation tools, using the example of a Web Service to illustrate the idea.

There is considerable natural scepticism about unconditional security or correctness arguments that ignore the operating context or environment. Usually, a system design can only be shown "correct" (i.e. meets its correctness obligations) under some assumptions about its mode of use and context - and the same certainly holds of security-related properties. For this reason, exploring and studying the combination of systems operating within particular environments, from a security point of view, is of great interest. In other contexts, similar process-focused modelling approaches have also been advocated [40,41].

We should be prepared to view a much richer landscape in which a system in one context is acceptably secure and performs a useful functional role. However, when used as a subsystem in some other context, the same system behaves insecurely and fails to operate as expected. Thus, the task of assessing the security of a system is closely associated with assessing those classes of environments in which a system operates safely.

Concerning the present form of SSML, some basic support tools already exist and a GUI-based analysis tool is currently under development. The purpose of such tools are to (1) provide ways of capturing models focusing upon the security aspects and (2) to explore their behaviour within particular environments using algebraic simulation and static analysis techniques. We hope to include goal statements that describe security-related properties, in terms of trace, state and

[1] CAPSL stands for Common Authentication Protocol Specification Language.

correspondence predicates, and allow user-directed exploration of what happens under certain initial conditions (i.e. "what-if" scenarios). An extended version of this paper (with Appendices) is available as a technical report [42].

Acknowledgements

I thank my colleagues Adrian Baldwin, Liqun Chen, Jonathan Griffin, Antonio Lain, Simon Shiu and Mike Wray at HP Labs for their helpful remarks and comments relating to this work. In addition, I am grateful for conversations with various protocol workshop participants which directly led to a number of improvements.

References

1. R. Needham, M. Schroeder: Using encryption for authentication in large networks of computers. CACM **12** (1978) 993–999
2. Anderson, R.: Security Engineering. Wiley (2001)
3. Paulson, L.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security **6** (1998) 85–128
4. Huth, M.: Secure Communicating Systems. Cambridge (2001)
5. Abadi, M.: Security Protocols and their Properties. In: Foundations of Secure Computation (F.L. Bauer and R. Steinbrueggen, eds.). NATO Science Series, Marktoberdorf, Germany, IOS Press (2000) 39–60
6. Lowe, G.: An attack on the Needham-Schroeder public key authentication protocol. Information Processing Letters **56** (1995) 131–136
7. Gollmann, D.: What do we mean by Entity Authentication? In: Proc. IEEE Symposium on Security and Privacy 1996, IEEE Computer Society (1996)
8. Monahan, B.: Introducing ASPECT - a tool for checking protocol security. Technical Report HPL-2002-246, HP Labs (2002) `http://www.hpl.hp.com/techreports/2002/HPL-2002-246`.
9. R.Cannetti, O.Goldreich, S.Halevi: The random oracle methodology, revisited (preliminary version). In: Proc. 30th Annual ACM Symp. On Theory of Computing, Perugia, Italy, ACM Press (1998) 209–218
10. D. Dolev, A. Yao: On the security of public key protocols. Technical Report STAN-CS-81-854, Dept. of Computer Science, Stanford University (1981) Also in *Transactions on Information Theory*, 29(2):198-208, 1983.
11. M. Burrows, M. Abadi, R. Needham: A logic of authentication. In: Proceedings of the Royal Society of London A. Volume 426., Royal Society (1989) 233–271 Also publ. (condensed) in ACM Transactions on Computer Systems, 8(1): 18-36, February 1990.
12. F. J. Thayer Fábrega, J. C. Herzog, J. D. Guttman: Strand Spaces: Proving Security Protocols Correct. Journal of Computer Security **7** (1999) 191–230
13. P. Ryan, S. Schneider: Modelling and Analysis of Security Protocols. Addison-Wesley (2001)
14. M. Rusinowitch, M. Turuani: Protocol Insecurity with Finite Number of Sessions is NP-complete. In: Proc. 14th IEEE Computer Security Foundations Workshop, IEEE (2001) 174–187

15. M. Abadi, A. Gordon: A calculus for cryptographic protocols: the Spi Calculus. Technical Report SRC-149, DEC-SRC (1998)
16. A. W. Roscoe, M.H. Goldsmith: What is intransitive non-interference? In: Proc. of 1999 IEEE Computer Security Foundations Workshop, IEEE Computer Society Press (1999)
17. P.Y.A. Ryan, S. Schneider: Process algebra and non-interference. In: Proc. 1999 IEEE Computer Security Foundations Workshop, Mordano, Italy, IEEE Press (1999)
18. N. A. Durgin, P. D. Lincoln, J. C. Mitchell, Scedrov, A.: Undecidability of bounded security protocols. Proc. FLOC Workshop on Formal Methods in Security Protocols (1999)
19. Ryan, P.: A CSP formulation of non-interference. Cipher, IEEE Computer Society Press (1991) 19–27
20. R. Focardi, R. Gorreri, F. Martinelli: Secrecy in Security Protocols as Non Interference. In: DERA/RHUL Workshop on Secure Architectures and Information Flow (S. Schneider and P. Ryan ed.). Volume 32 of Electronic Notes in Theoretical Computer Science., Elsevier, (1999)
21. J. Millen, V. Shmatikov: Constraint solving for bounded process cryptographic protocol analysis. In: Proc. 8th ACM Conference on Computer and Communications Security, ACM (2001)
22. S. N. Foley: A Non-Functional Approach to System Integrity. IEEE Journal on Selected Areas in Communications (2003)
23. F.Martinelli: Analysis of security protocols as open systems. TCS **290** (2003) 1057–1106
24. A.D.Gordon, A. Jeffrey: Types and effects for asymmetric cryptographic protocols. In: Proc. 15th IEEE Computer Security Foundations Workshop (CSFW 2002), IEEE (2002) 77–91
25. C. Bodei, P. Degano, H. R. Nielson, F. Nielson: Static analysis for secrecy and non-interference in networks of processes. In: Proc. PACT'01. Volume 2127 of LNCS., Springer (2001)
26. I. Cervesato, N. Durgin, J. C. Mitchell, P. Lincoln, A. Scedrov: Relating Strands and Multiset Rewriting for Security Protocol Analysis. In: Proc. 15th IEEE Computer Security Foundations Workshop, IEEE (2000) 35–51
27. P.Y.A. Ryan: Mathematical Models of Computer Security. In: FOSAD 2000. Volume 2171 of LNCS., Springer (2001) 1–62
28. P. Syverson, I. Cervesato: The Logic of Authentication Protocols. In: FOSAD 2000. Volume 2171 of LNCS., Springer (2001) 63–137
29. P. Samurati, S. de Capitani di Vimercati: Access Control: Policies, Models and Mechanism. In: FOSAD 2000. Volume 2171 of LNCS., Springer (2001) 137–196
30. J. D. Guttman: Security Goals: Packet Trajectories and Strand Spaces. In: FOSAD 2000. Volume 2171 of LNCS., Springer (2001) 197–261
31. A. D. Gordon: Notes on Nominal Calculi for Security and Mobility. In: FOSAD 2000. Volume 2171 of LNCS., Springer (2001) 262–330
32. R. Focardi, R. Gorreri: Classification of Security Properties (Part 1: Information Flow). In: FOSAD 2000. Volume 2171 of LNCS., Springer (2001) 331–396
33. M. Abadi, P. Rogaway: Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). Journal of Cryptology **15** (2002) 103–127
34. B. Pfitzmann, M. Waidner: A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In: Proc. of IEEE Symposium on Security and Privacy, Oakland, California, IEEE Computer Society Press (2001) 184–200

35. F. Nielson, H.R. Nielson, C. Hankin: Principles of Program Analysis. Springer (1999)
36. D. Clark, C. Hankin, S. Hunt: Information flow for Algol-like languages. Journal of Computer Languages (2002)
37. A. Sabelfeld, A. C. Myers: Language Based Information Flow Security. IEEE Journal on selected areas in Communications **21** (2003)
38. G. Denker, J.K. Millen, H. Rueß: The CAPSL Integrated Protocol Environment. Technical Report SRI-CSL-2000-02, SRI, Menlo Park, California, USA (2000)
39. G. Denker, J.Millen: CAPSL intermediate language. In: Proc. of FLoC Workshop on Formal Methods and Security Protocols. (1999)
40. Bruns, G.: Distributed Systems Analysis with CCS. Prentice-Hall (1997)
41. G. Birtwistle, C. Tofts: Relating Operational and Denotational Descriptions of Demos. Simulation Practice and Theory **5** (1997) 1–33
42. Monahan, B.: From Security Protocols to Systems Security - Making a case for systems security modelling. Technical Report HPL-2003-147, HP Labs (2003) `http://www.hpl.hp.com/techreports/2003/HPL-2003-147`.

From Security Protocols to Systems Security (Transcript of Discussion)

Brian Monahan

HP Laboratories, UK

Pekka Nikander: Do you have any feeling for how much of this system you can model?

Reply: It's a moveable feast: you can choose the boundary. But if you don't have any boundary at all then I don't think you've got enough context. . . you need to talk about conditions that are established through combinations of subprotocols, and they have to be in context. I think it's not just a simple matter of fact, of saying "OK, we'll just work with the protocol and hope everything else will go away." It won't. And I don't yet know where to stop.

Bruce Christianson: When you say "the protocol" I assume you're using that in a narrow sense of just meaning the messages and the state of the protocol itself, is that right? It's clear that you need to model some of the other aspects of those systems on which the protocols run.

Reply: Yes, that's where I was coming from. I originally started doing this work on very constrained message sequence charts. When people asked if I could prove completeness, I said that there may be a key to prove completeness for this very narrow class of protocols, but what was the point, because the real problems were all around the side. You need to broaden out the model and then completeness questions probably disappear.

Birgit Pfitzmann: You know, detailed semantics would help us.

Reply: Yes, absolutely. All of this can be explained in terms of layer transitions. So I'm sure there is a relationship between this and the work you've done.

Mads Dam: If you start with the message sequence charts, there are a lot of incomplete descriptions. Modelling a complete API or something like that would need some persistent state information, and you would need to talk about the actual communication structure.

Reply: I'm sure we will, but we don't have a complete story just yet. The ability to play I think is very important, so that implies some tools. You want people to be able to see these consequences, so we'll try to expose these attacks a lot more clearly, rather than just turn it into a formal game that's probably useless.

Biometrics to Enhance Smartcard Security*

Giampaolo Bella[1,2], Stefano Bistarelli[3,4], and Fabio Martinelli[4]

[1] Computer Laboratory, University of Cambridge, UK
giampaolo.bella@cl.cam.ac.uk
[2] Dipartimento di Matematica e Informatica, Università di Catania, Italy
giamp@dmi.unict.it
[3] Dipartimento di Scienze, Università "G. D'annunzio" di Chieti-Pescara, Italy
bista@sci.unich.it
[4] Istituto di Informatica e Telematica, CNR, Pisa, Italy
{stefano.bistarelli, fabio.martinelli}@iit.cnr.it

Abstract. A novel protocol is proposed to address the problem of user authentication to smartcards by means of devices that are currently inexpensive. The protocol emulates expensive *Match On Card* (MOC) smartcards, which can compute a biometric match, by cheap *Template on Card* (TOC) smartcards, which only store a biometric template. The actual match is delegated to an extension of the cryptographic module running on the card host, which is called *Cryptoki* according to the PKCS#11[9] standard. Compliance to such a standard increases the portability of the protocol. Informal reasoning confirms the protocol strenghts, though its formal verification in terms of established equational techniques appears to be at hand.

1 Overview

Smartcards are currently used as a secure and tamper-proof device to store sensitive information such as digital certificates, private keys and personal information. Access to smartcards has historically been regulated by a trivial means of authentication: the Personal Identification Number (PIN). A user gains access to a card if he enters the right PIN.

Experience shows that PINs are weak secrets [6] in the sense that they are often poorly chosen, and that they are easy to loose. Moreover, actual implementations using PINs often consider the channel between host and smartcard secure, so they simply send PINs in the clear. This allows many simple attacks [6]: a simple Trojan on the host, for example, could easily sniff the PIN and store it for future use (we succeeded in eavesdropping the PIN using four different, commercially available, smartcards).

Biometric technologies have been proposed to strengthen authentication mechanisms in general by matching a stored biometric template to a live biometric template [2,3,4]. In the case of authentication to smartcards, intuition imposes

* This work was partially supported by MIUR project "Tools, techniques and methodologies for the information society" and by a CSP grant for project "SeTAPS II".

the smartcard to perform the match. But this is not always possible because of the complexity of biometric information such as fingerprints or iris scans, and because of the yet limited computational resourses offered by currently available smartcards.

In general, three strategies of biometric authentication can be identified.

Template on Card (TOC). The biometric template is stored on a hardware security module (smartcard or USB token). It must be retrieved and transmitted to a different system that matches it to the live template acquired by special scanners from the user. Cheap *memory-cards* with no or small operating systems and onboard applications are generally sufficient for this purpose.

Match on Card (MOC). The biometric template is stored on a hardware security module, which also performs the matching with the live template. Therefore, a *microprocessor* smartcard is necessary, which must be endowed with an operating system running suitable match applications.

System on Card (SOC). This is a combination of the previous two technologies. The biometric template is stored on a hardware security module, which also performs the matching with the live template, and hosts the biometric scanner to acquire, select, and process the live template.

Our research aims at the development of a viable and easy-to-deploy protocol to allow secure authentication of users to smartcards, enforced by biometric matching. Clearly, if we could cast a spell, we would adopt the third of the strategies sketched above. Indeed SOC cards appear to be the best in terms of security, as everything takes place on card. Unfortunately, the existence of SOC cards is still confined to research laboratories at present [8,11]. The benefits deriving from MOC cards are valuable themselves, although these cards must still be considered rare and expensive. Reasonably cheap microprocessor cards do come with a cryptographic processor, which unfortunately is too specialised to perform biometric matches. Going down the hierarchy of prices, we find that TOC cards are obviously inadequate.

In the present setting, how can we implement biometric authentication on smartcards that are commercially available? We address this issue by developing a novel protocol that employs inexpensive TOC cards as if they were MOC cards. The actual match is carried out by a module added to the *Cryptoki* application running on the card according to the PKCS#11 standard defined by RSA laboratories [9]. In principle, if we trust the Cryptoki module, then we should protect the extra exchange that we expect is necessary to transmit the stored template from the card to the Cryptoki. By doing so, the resulting protocol should not be more vulnerable than the corresponding protocol for MOC cards. Our ongoing work on formally analysing the proposed protocol appears to confirm such a claim.

The presentation of the protocol messages may reveal possible optimizations. The current design is imposed by the restricted set of functions that we have to implement in the applet that the smartcard shall load. We have a working alpha implementation of the protocol, whose details will be published in upcoming papers [1]. Another important remark is that our design is obtained incrementally from a shorter design for MOC cards. Hence, it will be easy to simplify the protocol into one for MOC cards, as these become more affordable.

After a brief introduction to PKCS#11 Cryptoki (§2), this paper describes our novel protocol (§3), sketches its ongoing formal analysis (§4), and concludes (§5).

2 PKCS#11: Cryptographic Token Interface Standard

The PKCS#11 standard specifies an API, called "Cryptoki" (cryptographic token interface), to devices that hold cryptographic information and perform cryptographic functions [9]. Cryptoki follows a simple object-based approach, addressing the goals of technology independence (any kind of device) and resource sharing (multiple applications accessing multiple devices), presenting to applications a common, logical view of the device called a "cryptographic token". The primary goal of Cryptoki is a lower-level programming interface that abstracts the details of the devices, and presents to the application a common model of the cryptographic device, called a cryptographic token (or simply token).

Cryptoki's general model is illustrated in Figure 1. The model begins with one or more applications that need to perform certain cryptographic operations, and ends with one or more cryptographic devices, on which some or all of the operations are actually performed. A user may or may not be associated with an application. Usually, a cryptographic device can perform some cryptographic operations, following a certain command set. For example, a smartcard can generate a nonce, but in practice cannot perform any operation by itself. To have the smartcard generate a nonce, the application should invoke function `C_GenerateRandom(...)` of the Cryptoki library. Figure 2 represents such interactions. Hence, to let the smartcard participate in the protocol, we will load on a javacard compliant smartcard a small applet responsible to run the protocol

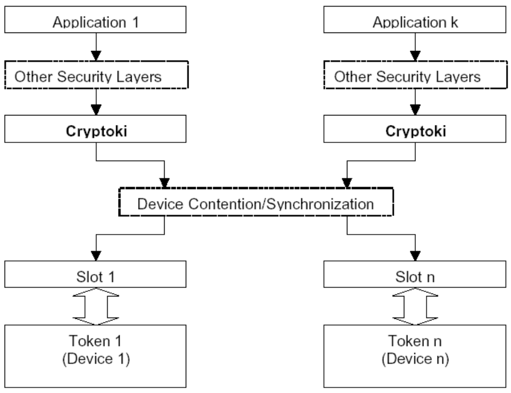


Fig. 1. A general Cryptoki model

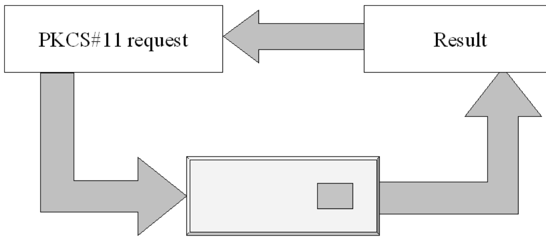


Fig. 2. The request-answer PKCS#11 model

with the Cryptoki. For example, a crucial operation that our applet will implement using the PKCS#11 interface is the login request to initiate a protocol session. It will be expressed using a PKCS#11 compliant $login(NULL)$ request.

3 Simulating MOC Using TOC

This section presents our protocol.

We follow an incremental design. First, we build a (PKCS#11 compliant) protocol for MOC cards. Then, we extend this protocol to function with the cheaper TOC cards. The price paid is one extra round of messages. While the latter protocol can be used at present, the former is its obvious "modification" for the time when MOC cards become cheaper.

3.1 A MOC Protocol

A MOC protocol can be defined as in Figure 3. In short, when the application sends the $login(NULL)$ request to the criptoky, it starts the protocol with the smartcard asking for a nonce. This will be used for a challenge-response phase to authenticate the Cryptoki to the smartcard (messages 1, 2 and 3). In message 3 the Cryptoki also transmits the just-acquired live template to the card, which checks the match and acknowledges (message 4).The session key generated by the smartcard could be used for all subsequent communication between the host and the smartcard. The live template is sent encrypted with a session key to ensure user privacy. A fuller description is given below, where this protocol is extended to delegate the biometric match to the Cryptoki. The key K_{sess} is the session key, N_{SC} is the nonce, K_{CK} and K_{CK}^{-1} are the public and private keys of the Cryptoki, respectively.

3.2 A MOC-Using-TOC Protocol

While the previous protocol requires MOC cards, the protocol presented here can do with cheap TOC cards. Figure 4 shows the new protocol, which extends the previous with two additional steps between 3 and 4. The additional steps essentially delegate the matching to a special Match Module of the Cryptoki. The keys K_{MM} and K_{MM}^{-1} are the public and private key of the Match Module, respectively.

$$\begin{array}{lll}
1.\ \mathsf{Cryptoki} & \rightarrow \mathsf{SmartCard}: & hello_{MOC} \\
2.\ \mathsf{SmartCard} & \rightarrow \mathsf{Cryptoki}: & \{\!|\{\!|N_{SC}, K_{sess}|\!\}_{PIN}|\!\}_{K_{CK}} \\
3.\ \mathsf{Cryptoki} & \rightarrow \mathsf{SmartCard}: & \{\!|\{\!|N_{SC}, Template_{live}|\!\}_{K_{CK}^{-1}}|\!\}_{K_{sess}} \\
4.\ \mathsf{SmartCard} & \rightarrow \mathsf{Cryptoki}: & \{\!|N_{SC}, Hash(Template_{live})|\!\}_{K_{CK}}
\end{array}$$

Fig. 3. A MOC protocol

$$\begin{array}{llll}
1. & \mathsf{Cryptoki} & \rightarrow \mathsf{SmartCard} & : \{\!|hello_{MOC} \\
2. & \mathsf{SmartCard} & \rightarrow \mathsf{Cryptoki} & : \{\!|\{\!|N_{SC}, K_{sess}|\!\}_{PIN}|\!\}_{K_{CK}} \\
3. & \mathsf{Cryptoki} & \rightarrow \mathsf{SmartCard} & : \{\!|\{\!|N_{SC}, Template_{live}|\!\}_{K_{CK}^{-1}}|\!\}_{K_{sess}} \\
3_1. & \mathsf{SmartCard} & \rightarrow \mathsf{Cryptoki_{match}} & : \{\!|Template_{live}, Template_{storedAC}|\!\}_{K_{MM}} \\
3_2. & \mathsf{Cryptoki_{match}} & \rightarrow \mathsf{SmartCard} & : \{\!|Hash(Template_{live}, Template_{storedAC})|\!\}_{K_{MM}^{-1}} \\
4. & \mathsf{SmartCard} & \rightarrow \mathsf{Cryptoki} & : \{\!|N_{SC}, Hash(Template_{live})|\!\}_{K_{CK}}
\end{array}$$

Fig. 4. A MOC-using-TOC protocol

First Phase: Cryptoki's Authentication to SmartCard, and Transmission of the Live Template

$$\begin{array}{lll}
1.\ \mathsf{Cryptoki} & \rightarrow \mathsf{SmartCard}: & hello_{MOC} \\
2.\ \mathsf{SmartCard} & \rightarrow \mathsf{Cryptoki}: & \{\!|\{\!|N_{SC}, K_{sess}|\!\}_{PIN}|\!\}_{K_{CK}} \\
3.\ \mathsf{Cryptoki} & \rightarrow \mathsf{SmartCard}: & \{\!|\{\!|N_{SC}, Template_{live}|\!\}_{K_{CK}^{-1}}|\!\}_{K_{sess}}
\end{array}$$

In this phase a challenge-response authentication between SmartCard and Cryptoki takes place. The SmartCard issues a fresh nonce and the Cryptoki returns it. The challenge response phase implies the presence of the user who knows the PIN associated to the smartcard. In fact, only with the knowledge of the PIN can message 2 be decrypted to obtain the nonce (this is a first step of authentication where a *what I know* feature of the protocol is used).

The next step is to create a message representing the biometric information of the user (a *what I am* feature). A *live* and *trusted* template is generated. The idea is to accept a biometric template only if coming from a trusted device (the signature shows this) and only if acquired during the current authentication session (the presence of the nonce ensures this). We can assume that the Cryptoki public key is already available to the smart card, otherwise the Cryptoki should just send the appropriate digital certificate with message 1.

The session key sent to the Cryptoki from the SmartCard will be useful to encrypt the biometric template for privacy reasons. The main reason to use a session key is because the user may want to secure his biometric information from eaves-droppers for a number of reasons. For example, one is that the fingerprint can reveal certain diseases. The same session key could be used for all the communication following the login phase.

Second Phase. SmartCard's Transmission of Stored Template to Cryptoki Matching Module for the Match.

$$3_1.\ \mathsf{SmartCard} \rightarrow \mathsf{Cryptoki_{match}} : \{\!|\mathit{Template}_{live}, \mathit{Template}_{storedAC}|\!\}_{K_{MM}}$$

$$3_2.\ \mathsf{Cryptoki_{match}} \rightarrow \mathsf{SmartCard} : \{\!|\mathit{Hash}(\mathit{Template}_{live}, \mathit{Template}_{storedAC})|\!\}_{K_{MM}^{-1}}$$

The SmartCard sends the stored template to the Cryptoki Matching Module, who is delegated the responsibility of doing the match. This step is secured under the key of the Cryptoki Matching Module for privacy reasons. If the live template matches the one stored in the attribute certificate $\mathit{Template}_{storedAC}$ then the Match module issues a signed message confirming such a match[1]. The use of a hash function ensures that no information related to the fingerprint template is leaked (as before for privacy reasons).

Last Phase: Successful Login

$$4.\ \mathsf{SmartCard} \rightarrow \mathsf{Cryptoki} : \{\!|N_{SC}, \mathit{Hash}(\mathit{Template}_{live})|\!\}_{K_{CK}}$$

If the SmartCard generates the last message, it means that a successful biometric match was performed and the *login* is successful. The message 4 containing the nonce and the (hashed) live template is also used to respond to the authentication challenge sent by the Cryptoki in message 3.

4 Towards a Formal Analysis of the MOC-Using-TOC Protocol

We can relate the MOC-using-TOC protocol to the MOC protocol using two criteria: functional behaviour and security features. We may use a variant of the CCS process algebra of Milner [7] to formally describe the behaviour of the components involved in the protocol, i.e. the SmartCard (S), the Cryptoki (C) and the match module of the Cryptoki (M).

The functional behaviour of the MOC-using-TOC clearly extends that of the MOC protocol with steps 3_1 and 3_2. Indeed, the specification of the whole Cryptoki consists of the specification of C in parallel with M, i.e. $C \mid M$. Note that this is possible because C and M do not need to communicate with each other or to share knowledge to continue the protocol. Indeed, the Criptoki needs

[1] If we also assume that the acquisition of the live template by the Cryptoki is performed in a confidential way, this message may embody a nice form of challenge/response authentication, using the pair $(\mathit{Template}_{live}, \mathit{Template}_{storedAC})$ as a nonce, due to the unpredictable nature of the $\mathit{Template}_{live}$. Such a form of freshness, is in principle not crucial for the security of the protocol because the result of the matching among the live and the stored templates should always be the same (thus replay attacks are not worthy). It may, however, turn out useful in a real implementation if we imagine that the accuracy of the matching algorithm may change over time (e.g., due to legislative acts).

to know the PIN (it could ask the user to provide it), while its matching module does not need it. In particular, the matching module could in turn delegate the match to another (remote) module.

Thus, we may expect that future Cryptoki applications implementing MOC may easily be adapted from those implementing MOC-using-TOC. For example, a MOC SmartCard may decide whether to perform itself the match between $\mathit{Template}_{live}$ and $\mathit{Template}_{storedAC}$ or to delegate it by taking steps 3_1 and 3_2. It may be reasonable to assume that the smart card will ask for either a MOC-using-TOC protocol or a MOC one depending on its internal features. The Cryptoki matching module will act according to this request. In the case of a MOC protocol, it will simply avoid to take steps 3_1 and 3_2. This is formally described by the fact that the set of possible execution traces of C is contained in those of $C \mid M$.

The security features of the MOC protocol are more interesting. The MOC protocol guarantees a mutual authentication of a SmartCard that knows a PIN with a Cryptoki that also knows the PIN and has a certificate. We assume that the $\mathit{Template}_{live}$ is acquired by a secure device and so no-one except the identified person is able to provide the Cryptoki with the fingerprint. (The $\mathit{Template}_{live}$ could also be acquired by a device trusted by the SmartCard.) We also assume that the Cryptoki is honest. Then, what is done is an authentication of the Cryptoki to the Smartcard, followed by the creation of a session key. This session key may be used for securing the subsequent connections.

We are interested in checking the following: whenever the Smartcard authenticates the Cryptoki, then the Cryptoki wanted to be authenticated by the Smartcard; the $\mathit{Template}_{live}$ provided by the Cryptoki equals the $\mathit{Template}_{storedAC}$ stored in the Smartcard that authenticates the Cryptoki; both SmartCard and Cryptoki share the PIN. From a theoretical point of view, the correctness of the protocol depends on secrecy of the PIN and of the Criptoki's private key.

Thus, the security property is a form of agreement on the message sent (the template) between the Smartcard and the Cryptoki. This may be modelled through common techniques such as the correspondence analysis and the inductive analysis [5,7,10]. We take the approach based on non-interference developed in [7]: a system is secure if its observable behaviour cannot be significantly altered by an enemy. Clearly, such a definition relies on the notion of *observable* behaviour. Indeed, honest agents in the protocol are able to take special actions denoting their beliefs on the current status of the computation. For example, a user A may raise an action $\mathit{start}(A, B, m)$ meaning that she is starting a run of the protocol with B for exchanging the message/template m. In the following, the observable behaviour only consists of this kind of actions.

A way to check that the protocol MOC-using-TOC is correct is to show that MOC is correct and that the security features of MOC-using-TOC are simulated by (or equal to) those of MOC-using-TOC.

The scenario of the analysis is as follows. Consider an external enemy X, described through a process X, which does not hold either the private key of the Cryptoki or the PIN (but could know the $Template_{live}$). Suppose also that the Cryptoki issues the control action $start(C, S, Template_{live})$ when it starts a communication with the SmartCard for a user that prompts the live template $Template_{live}$. Assume that the SmartCard issues the action $finish$ $(S, C, Template_{storedAC})$ whenever it considers a run of the protocol as terminated. This happens when the SmartCard concludes that an honest Cryptoki provided a live template that matches the one stored in the attribute certificate $Template_{storedAC}$.

Note that MOC is such that, for each enemy X that satisfies our analysis assumptions, we have

$$(S \mid C \mid X) =_{Obs\_traces} start(C, S, Template_{live}).finish(S, C, Template_{storedAC})$$

where $Template_{live}$ and $Template_{storedAC}$ match.

We may show that when we are interested in such a property then MOC and MOC-using-TOC agree on their observable behaviour (that is to say they enjoy the same property)

$$(S \mid C \mid M \mid X) =_{Obs\_traces} (S \mid C \mid X)$$

Here, *Obs_traces* denotes the equalities of set of traces where only the set of observable actions is taken into account. This analysis may be defined for each security property of interest.

5 Conclusions

Modern, inexpensive TOC smartcards cannot compute a biometric match as MOC smartcards. We have developed a protocol which simulates the MOC strategy through the use of TOC cards. In practice the actual match is delegated to the Cryptoki module of the card. The computational burden that the protocol puts on the smartcards amounts to a classical PIN match.

Our protocol achieves the goal of authenticating a user to a smartcard by means of the user's biometric information, although the smartcard does not perform the actual match. The design presented here has been fully implemented using the PKCS#11 standard, which is the most common interface language for current smartcards.

Acknowledgements

We would like to thank the workshop attendees for the exciting discussion environment and, in particular, for pointing out a potential weakness of a previous version of our protocol due to a non appropriate use of weak secrets.

We are also indebted to Luca Bechelli and Stefano Frassi for valuable suggestions about the PKCS#11 implementation constraints.

References

1. L. Bechelli, S. Bistarelli, and S. Frassi. A protocol for simulating match-on-card authentication through the use of a template-on-card technology. *Working draft.*
2. L. Bechelli, S. Bistarelli, F. Martinelli, M. Petrocchi, and A. Vaccarelli. Integrating biometric techniques with electronic signature for remote authentication. *ERCIM News*, (49), 2002.
3. L. Bechelli, S. Bistarelli, and A. Vaccarelli. Biometrics authentication with smartcard. Technical Report 08-2002, CNR, Istituto di Informatica e Telematica, Pisa, 2002.
4. L. Bechelli, S. Bistarelli, and A. Vaccarelli. Certificati per impronte digitali memorizzati su smartcard. Technical Report 09-2002, CNR, Istituto di Informatica e Telematica, Pisa, 2002. In Italian.
5. G. Bella. Inductive Verification of Smart Card Protocols. 11(1):87–132, 2003.
6. Mike Bond and Piotr Zielinski. Decimalisation table attacks for pin cracking. Technical Report UCAM-CL-TR-560, University of Cambridge, Computer Laboratory, feb 2003.
7. R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proc. of FM'99*, volume LNCS 1708, pages 794–813. Springer, 1999.
8. N. J. Henderson and P. H. Hartel. Pressure sequence - a novel method of protecting smart cards. In J. Domingo-Ferrer, D. Chan, and A. Watson, editors, *Proceedings 4th Int. IFIP wg 8.8 Conf. Smart card research and advanced application (CARDIS), Bristol, UK*, pages 241–256. Kluwer, 2000.
9. "RSA Security Inc.". Pkcs#11 v2.11: Cryptographic token interface standard. Technical Report PKCS#11 v2.11 r1, RSA Security Inc., nov 2001. RSA Security Inc. Public-Key Cryptographic Standard (PKCS).
10. L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.
11. B. Struif. Use of biometrics for user verification in electronic signature smartcards. In *Smart Card Programming and Security - Proceedings of the International Conference on Research in Smart Cards (E-smart), Cannes, France*, pages 220–228, sep 2001.

Biometrics to Enhance Smartcard Security
(Transcript of Discussion)

Fabio Martinelli

Istituto di Informatica e Telematica, Italy

Matt Blaze: What are the semantics of match-on-card? In what way does the host computer trust it?

Reply: The host computer in our case is the crypto key. For the matching card the crypto key is also a crypto key match module, in this case, because it's another trusted module. The thing is that the smartcard trusts this additional confidence. It is a new component, to perform the match.

Matt Blaze: OK, so that's the trusted component.

Reply: Yes, all these components are trusted in our analysis, it's true. After the markers see the file index, they give the information to the match model of the crypto key. This signed key we need to encrypt with the public key of the module, but again this is only for privacy reasons. Note that here we have no freshness, and there's no necessity to have freshness on a matching card for this information. However in a real implementation, where there is some constraint about how the position of the match module will change, you will need some nonces here, and these have been provided.

Mike Roe: Why is this more efficient than just doing the match-on-card?

Reply: No, this is no more efficient. The match-on-card requires translations, notations, so it requires some operations that the smart marker doesn't actually provide. This is the point.

Mike Roe: Oh yes, sorry.

Reply: So actually this is exactly the job of this marker. The nice thing is that for us is that you seem to have to insert two steps. All the rest of the protocols remain the same from this side of the application.

Richard Clayton: I'm a little puzzled because if you're assuming the crypto module can do the comparison between the templates with biometric stuff, then why don't you use that for the PIN as well, rather than what you do at the beginning? It's actually brute forcible. Attack that line 2 there because the PIN is short.

Reply: Yes, I know that the PIN here is a very weak secret, usually what to do is not to have the biometric control of the key, but to see if we can provide two alternative mechanisms; you log onto this marker with a key or you go through authentication. In our minds we said we wanted one mechanism on top of the other so in order to connect these mechanisms you have to know both the key, and the private key of the crypto key. Certainly the limitation that we have in mind is that it is possible that the crypto key should ask for the user to sign. We can imagine also that this message comes from a trusted device. Then what you have is that the crypto key knows nothing about the security of the protocol, because the problem is not at this end, and the authentication of the message is this second message. So I know that's encrypted with a key. It's not a good way, because usually they are very weak, but first of all you can decide the length of this scheme at the application layer, and then it's definitely something that you want on top of the key security.

Matt Blaze: You seem to be assuming that the card has blocked for some layer...

James Malcolm: It's a template I think.

Matt Blaze: Oh, OK, fine, that's good. So the liveness?

Reply: The life is a nonce. These always match. If you assume that the crypto key match module is honest, and never signs two messages if they do not match, then you don't need to worry. But as I told you, you could intercept or avoid attacks if you wish. For instance if the law changes, or something like that. You want to require that these two messages come exactly from that kind of authority.

Matt Blaze: If the crypto module that's doing the match is trusted, what benefit is there of doing any computation on the party? Intuitively it seems that you just send a certified template to the trusted module that is certified, maybe encrypted in some way, that the module can verify within that template. What am I more secure against by having a smartcard as opposed to a memory device?

Reply: You are not more secure, you can simply add the same level of security as a marker in a way that is less expensive. I want to provide that anyone can improvise legally the life template. The real problem is the matching card problem.

Matt Blaze: I can prevent somebody from providing a template just by certifying the template that is running certificates to the signature. So does life need the parties present?

Reply: No life means that this information is public information and is being acquired by a cluster device at this moment. You could also consider that someone is able to send back their public information to you, so you need to ensure that the information being acquired by a clustered device is new information.

Richard Clayton: Supposing I do have some computing resources and I eavesdrop on messages one, two and three, then by brute force I can learn the PIN. I have then learnt the session key, and the nonce.

Reply: But it won't verify the key.

Richard Clayton: No, I don't have the private key, but the other thing is that the session key doesn't seem to be doing anything. All it's doing is obscuring the signed thing of the T line. The T line, it seems to me, is not a secret at all, it's just what the fingerprint reader has seen today. What is the session key doing in the protocol?

Reply: At the moment the session key does nothing, although in many cases you have to protect the nonce.

Richard Clayton: Why? It's not used again, except right down the bottom. If you're going to assume that you've got the public key of the client sitting on the card anyway, then why don't you use that for the second message, or use the keys for that rather that encrypting them.

Reply: At the end of the problem you use your session key to keep sensitive information with the marker. Actually this is perhaps misleading, it's a secret to establish between the two markers and the crypto key. The smart card has to then send sensitive information to the crypto key.

Bruce Christianson: The way the protocol is written there the session key is no stronger than the template. But you could leverage a fresh strong secret from the weak secret, we know how to do that[1].

Reply: What you're saying is that usually we perform authentication at this level with the key node. I don't see there is something more in terms of defence of the key, but this means that you're able to provide this message.

Bruce Christianson: Oh, even if you don't want to get a stronger secret you can still leverage freshness in a way that doesn't reveal the PIN to a passive attack. This is purely a technical interjection to the secure protocol you have there. But your approach to the action the user would take is very interesting, you've got that right whereas a lot of people haven't.

Reply: There's a few aspects of this stuff which I think we could usefully discuss elsewhere.

[1] Christianson, Roe, Wheeler, "Secure Sessions from Weak Secrets", these proceedings.

Blind Publication: A Copyright Library Without Publication or Trust

James A. Malcolm

University of Hertfordshire

1 Introduction

Digital documents, be they text, image, audio or video, are very easy to copy. There is therefore considerable interest on the part of some originators of digital documents in trying to control the copying of such documents. Of course, this is an impossible task (without special hardware), but it does not stop people trying. But if we cannot prevent illegal copies, we would like to be able to identify them.

2 Printed Works

A copy of each copyright book published in the UK is placed in each of a small number of copyright libraries. The approach is that the document whose copyright is asserted will be placed into the library in order to resolve any future dispute as to whether the document is original or not. Not only is the document physically placed in the library, but there is also some physical record of when it was put there, and it is there for all to see.

One problem with a copyright library is that the secret is published: documents placed there are publicly accessible and thus easy to copy. A particular example of this dilemma is faced by administrators of student Project courses: should a student's Project report be placed in the university library or is this only making life easier for those who wish to commit plagiarism? In general, though, (and especially before the invention of the photocopier) the difficulty of copying made this issue relatively unimportant in regard to physical books.

An alternative is that a document is given to some kind of notary and is kept secret. In this case the problem is that we have to trust the notary both to preserve the evidence and equally to provide the evidence we require should a dispute arise.

Ideally, we would like to be able to register a document in some kind of system without having to trust that system both to keep the content secret and to take the correctside in any future dispute.

3 Digital Documents

If the document we want to protect is digital, then it is hard to verify the original. Bit patterns have no provenance [1].

We can embed some kind of source identifier and other details about the document within it. Unfortunately such meta-data can be modified or removed. Watermarking techniques attempt to make it harder to examine or to modify the meta-data, but it is not entirely impossible. Again there is no provenance in the bit patterns used to mark the document so the fact that there is a watermark there doesn't determine when the watermark was added to the document nor does it state whether the watermark was placed there by the originator. It is also hard to see how to add a watermark to plain (ascii) text in the general case.

Alternatively, a hash of the document can be published and so used to prove the authenticity of a document at a later date, but a very tiny change to that document means that the signature will not match. As far as any digital signature process is concerned, it is no longer the same document. So hash functions are not the answer to the problem of copyright violation: small changes give very different hash values. Also, you and I can each publish a hash of a document, and log the date, but only by seeing the document can we verify what was signed. I cannot tell whether your document is very similar to mine or completely different. In fact neither of us can check the other's claim unless the text is revealed.

We can view this as a requirement for an application dependent hash function, where similar documents hash to similar values (contrary to usual principles of hashing).

4 What Are We Trying to Protect?

So, from a consideration of digital signatures, it becomes apparent that in many cases it is not the *bits* of the document that should be protected by a copyright marking scheme, but the salient features of that document. Two simple examples of this will suffice. Firstly if we take a well-known story and change one word it is usually still the same story but a digital signature of the document would be quite different. Similarly an image of a well-known painting rendered in red on a blue background may be a very different bit pattern from that representing the original image, but it would still be a violation of the original's copyright.

What the salient features should be probably depends on the nature of the application, but it is clear from water marking and digital signatures that simply protecting the bit pattern is really not solving the problem that matters.

If we can extract salient features of a document, hash each feature, and construct a feature set of these values; then this feature set can be placed in a library, possibly widely distributed. But now a new document can similarly have its features extracted and this feature set can be compared with feature sets already in the library. Similar documents will have common features, and so the intersection of the two sets will be large. Any document found to have a similar feature set will be deemed to be a possible copy.

It is important to note that similarity of feature sets identified at this stage is not proof of copying, but it greatly reduces the number of cases that would have to be more closely examined. The point is that similar documents should have similar fingerprints. This is quite opposite to the position with digital signatures, where ideally changing just one bit in a document should change about half of the bits in the signature.

The requirements therefore are:

– that going from document to feature set should be straightforward,
– that reverse engineering from a feature set to a document should be hard,
– that the feature set should be reasonably compact, certainly not much larger than document itself,
– and that the set of all possible features should be large.

5 Efficient Realisation of Feature Sets

The original ordering of the elements in the feature vector is discarded – it is indeed a feature *set*, with the set cardinality equal to the document length. The number of possible features is very large, but the number in any document is much smaller. We could represent the feature set as a bit array and compress it, or hash it to a much smaller number of bits, and accept collisions, or possibly use a Bloom filter [2]. The representation should not be too large, so as to avoid leaking too much information. If this leads to a false positive, and hence a dispute, we still don't need to publish the documents whose feature sets are similar, as we can repeat the publication process using a different hash algorithm and/or a larger feature set.

6 What Features?

For an image, extracting the features that characterise it is a hard problem. Is an ASCII art Mona Lisa a copyright violation? An image can be translated, cropped, scaled, subset, colour mapped, etc, etc. If one takes a specific image file format, and restricts the manipulations against which one wants the system to be resilient, then the problem becomes easier. For example, from a JPEG file one might extract the DCT coefficients of each block and use these as the features. Any particular image therefore would contain only a small fraction of the possible features, and (I hypothesise) every picture a different subset. It is not obvious, though, whether these features would reflect the Scene, the Image, or the implementation of the compression algorithm.

Audio can be converted to a midi-like note sequence (using FFT) and the sequence of intervals used as the basis for a feature set. Again, we have to ask how effective these would be (though there is evidence that very simple features uniquely characterise musical themes). One issue with music is; what are you protecting – the tune or the performance? Rachmaninov or Rattle? And what about the protection of mixed ownership digital artefacts, such as a bootleg recording of Hendrix playing Dylan?

Questions such as this will influence what feature set is appropriate.

7 Text Feature Extraction

For text, the lexical features that should be extracted need to preserve some information about the ordering of words in the text (because vocabulary alone will not adequately differentiate one document from another). On the other had choosing a feature that preserves too much information about word order will make it easier for small and trivial changes to a copied document to fool the system into thinking it to be original.

The feature that we use is a three word trigram. Note that the order of these trigrams is not important; they characterise the text in whatever order they are presented. In a sense this set of trigrams can be called a fingerprint, except that a fingerprint is a small identifier of a larger body, while this set of trigrams is usually larger than the original text.

The well-known Zipfian distribution of words [3] captures the fact that in any text a few words are very common, but most words are rare. This distinctive distribution is more pronounced for word bigrams, and even more marked for trigrams, as is illustrated by statistics on the frequency of trigrams in the Wall Street Journal corpus [4] (page 258). In this corpus of nearly 39 million words, about 14 million different trigrams appear, of which nearly 11 million appear only once in the entire corpus. This leaves about 3.2 million non-unique trigrams that appear on average 8.7 times each in the corpus, but the average frequency of these non-singletons grows very slowly as the corpus size increases. We can get an idea of how little overlap can be expected between independently written texts if we hypothesise for a moment that all the non-unique trigrams are equally common: this would give each trigram only about 9 repetitions in the entire corpus, so the chances of many matching trigrams occurring in two independent samples is still very small.

The characterization by trigrams typifies a single text, and levels of chance matches with other texts are very low, even when the texts are on the same subject. For example, consider the two short passages below in which different journalists describe the same event [5]. Although they quite probably had access to the same source material, and the texts used are very close semantically, the actual words are substantially different and only four trigrams match out of a total of 29 in one passage, 32 in the other.

Four North Korean fighter jets intercepted a US spy plane over the Sea of Japan at the weekend and one used its radar in a way that indicated an imminent attack ... *Example 1: 31 words, 29 trigrams, 4 matching trigrams*
A large US airforce surveillance aircraft was intercepted by **four North Korean** MiG fighters on Sunday **over the Sea of Japan**, with one of the fighters locking its firing radar on to the aircraft ... *Example 2: 34 words, 32 trigrams, 4 matching trigrams*

8 Conclusion

A fingerprint based on hashing the triples in a document can be published to provide a way of identifying similar documents which does not require access to the document text.

Further work is required to see whether this approach can be extended to the much more complicated cases of images and sounds.

References

1. B. Christianson and J. A. Malcolm. Binding bit patterns to real world entities. In *Security Protocols: 5th International Workshop on Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 105-113. Springer Verlag, 1998.
2. B. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, **13**(7) pages 422-426, 1970.
3. G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.
4. D. Gibbon, R. Moore, and R. Winski, editors. *Handbook of Standards and Resources for Spoken Language Systems.* Mouton de Gruyter, 1997.
5. T. Reid. North Korean MiGs intercept spy plane. *The Times*, 4th March 2003.

Blind Publication: A Copyright Library Without Publication or Trust
(Transcript of Discussion)

James A. Malcolm

University of Hertfordshire

Good morning. The title, Blind Publication, may not convey too much about what I'm actually going to talk about: it's a simple idea which I hope will have some merit, and it really started when I started thinking about copyright. Now all sorts of people are thinking about copyright, digital rights, *etc*, but they're looking at movies, images, all the sorts of stuff that Hollywood thinks is valuable, and I thought, well let's look at something much simpler, let's look at plain text and see if we can get any good ideas by looking at a much simpler situation. So that's the area that I want to look at, but I thought we'd start with a picture as an analogy of what I want to do.

Essentially what I want to do is to look at these three pictures, but you can't see them properly, they're partially hidden, they've not been fully published, and then to identify that two of these pictures are similar in some sense, and one is different. And you get kind of a clue that there are some features in the first picture which are also present in the second. They're clearly very different, they're clearly not the same picture, but there's some commonality: there's a feature (the smile) in image 1 and image 2 which is common. Now what I've discovered, through looking at text, is that perhaps if can we extract features from copyright documents we may be able to find a way of comparing documents without actually publishing them. This is the basic idea. Essentially I want to leak just enough information from one copyright document so that you can tell whether or not the other copyright document is the same, without actually looking at the entire copyright document.

Now feature extraction might seem to be a hard AI kind of problem, but certainly for text it turns out be relatively straightforward, and a fairly dumb brute force sort of approach works quite well. I'm going to talk about the general principle, have a brief look at how it might be extended from text to things like graphics and music, and then look at how it actually works, I think quite well, in plain text.

Victoria Stavridou: What kind of text are you talking about? The characters, or the words, or ...

Reply: Essentially I'm looking at the words, which can be extracted from a typical word-processor document, HTML file, or whatever. Maybe the numbers as well, but probably just the words.

Victoria Stavridou: In all languages? In Japanese as well?

Reply: Well European languages, I've not got as far as Japanese[1].

Birgit Pfitzmann: And is it time stamping, so that for example, I might want to prove that it was my document and I already had it, despite somebody stealing it and trying to change it and make it a little bit different so that it seems to be another one.

Reply: Yes, I want to be able to say, "I published this document" in an open way that somebody else can verify, so that later on if somebody has managed to take a copy my document (because I gave it to some friends one of whom turned out to be untrustworthy), then I'd like to have some way of confirming that the document was in fact mine[2]. I think it will become clear shortly.

The problem with digital documents is that they are easy to copy, so we want to control copying in some way. We could say, let's try and prevent people copying with DRM, and then we find people download electronic books, take screen shots, and then print them. So there are ways round some of these technologies, but people try.

Or we can identify a document and say we'll put a watermark or something in the copy so that people can later find the copy and say, look it's got my watermark in it. But one of the problems is you can't tell which is the original, because there is no provenance information in digital documents.

So let's have a look at what happens with conventional printed works. There are two approaches: the main one with printed works is you have copyright libraries; anything that's important and is published ends up in the University Library, and there's a slip of paper glued into the book, we know when the book arrives, it's an open process, we know when that copyright volume was put into the University Library. But the secret is published. So, for example, with student projects, and we've got hundreds and hundreds of student projects every year,

[1] There is some evidence that this approach *would* work for Chinese, as Chinese characters seem to show the same Zipfian frequency distribution as words in European languages.

[2] A bit like Hooke's *'ut tensio, sic vis'* published as an anagram.

it would not be very nice to put them all in the library, so that students could benefit from what their predecessors had done, and learnt, because of course this encourages copying. So having a situation where our secret is actually published for everybody to see, has got some weaknesses.

The alternative approach is you can give the document some kind of notary, you can keep it secret, but then you have to trust the notary to take your side if there is a dispute.

But there is an intermediate situation where you could register the document to some kind of system without having to trust it to keep the secret, by revealing just enough information.

The insight here, which started from looking at text but it applies in some sense to other kinds of documents as well, is that what we're trying to protect in a copyright system isn't actually bits. Because if you change one bit, then using hashes or digital signatures, you've got a completely different digital signature, a completely different hash. And so we really want to protect not the bits but the salient features, you want to know this is a picture of the Mona Lisa, not that it's got this particular bit pattern. Two documents might be completely different bit patterns, but the same picture.

Victoria Stavridou: Kind of like a fingerprint, rather than just bits?

Reply: Yes, but you don't need the whole fingerprint to identify the criminal.

Matt Blaze: To get a little more precise, the reason just hashing the document isn't sufficient for your purposes is that we want to be able to look at two hashes and tell if they were similar before they were hashed, that is without going back and looking at the original documents. So we want the library to say, "hey wait a minute, somebody already submitted that hash".

Reply: Absolutely. It would show that something similar has been submitted before. I mean, it's not proof, but it is raising a worry, and then you can investigate.

So what we need to do, I believe, is extract the features from the existing document, whatever those might be, take the hash of each feature, and create sets of all the hash values. Similar documents are going to have common features and therefore the intersection of the two feature sets will be large, and therefore we suspect there may be a problem, but if the intersection is small we can be confident that there is no problem.

Obviously if we just published a huge collection of hash values it's going to be an enormous collection of bits, so instead we use a bit vector with a bit on for each feature that is present.

Victoria Stavridou: Are the features published?

Reply: The algorithm would be public, so in a security sense they are, but there are a lot of collisions. The number of possible features is very large, and the number in any document is much smaller, so when you hash each feature you can represent it as a bit array where a considerable number of hash values map to the same bit.

Victoria Stavridou: In the case of plagiarism detection, a student who knows how this works is just going to avoid those features to make sure that the intersection is real small.

Reply: What turns out to be suitable with text is that every single small group of words is a feature, and so in practice that attack is impossible. It's more a trouble than writing it yourself to change every third word, and when we explained to students exactly how a plagiarism detection system based on these ideas works, one of the students in the audience said, "all we need to do is to change every third word", and we said, "yes, that's fine".

George Danezis: But I think the whole problem reduces to one of signal to noise detection, because there are some triples that occur naturally in almost all documents, that you don't want to detect as collisions, while there are more rare triplets that you really want to use as a fingerprint.

Reply: Yes, it turns out that most triples are rare.

George Danezis: Are they?

Reply: Almost all word triples are rare[3]. For example, when I was looking for a specific case of student plagiarism I searched for "router manufacturers happy". Now "router manufacturers" is quite a common string and "manufacturers happy" probably is too, but "router manufacturers happy" is a three word string that occurs exactly six times on Google. All six are actually the same document: one is the original, there are three *bona fide* copies (where the source has been cited properly), and the other two are student projects. [Laughter]

Richard Clayton: Isn't there a trust issue here, because you take your document which you've written and you push it through this magic function which produces this magic number, and you publish that number. I steal your work, right, and I write my document, how are you going to make sure I push it through the *correct* system in order to make sure that number I publish is in fact representative of my document, because I'm not going to show you my document either?

Reply: Indeed, that's a big weakness, I accept that.

Kamil Kulesza: As for the triples, I'm afraid that it's not content sensitive, because if you search Google for "verifiable secret sharing", I am positive that you will receive thousands of matches.

Reply: Absolutely.

[3] We need to distinguish conceptually between a triple *instance* and a triple *type*. A document may contain many instances of a particular triple type. The assertion is about triple types; the overwhelming majority have no instances at all in any document, and the majority of the remainder have just one instance in total.

Kamil Kulesza: And this is a triple, right.

Reply: Suppose you take all the triples in the document. What people tend to assume when they think about this is, yes, there'll be some common triples, and some less common triples, so as we go through the triples in rank order we imagine a gradual tailing off in the frequency. What actually happens from observation - and it's not a new thing, and not just by me, but by other people doing work in speech processing, and analysing huge volumes of text, like the Wall Street Journal corpus of 50 million words - is that most triples, most three words strings, only occur once.

John Ioannidis: Like "Wall Street Journal"? [Laughter]

Reply: Some triples *are* common, but most of these features only occur once, and therefore if most of the features occur once, then most of the documents will contain mostly unique triples.

George Danezis: Are you excluding function words?

Reply: No, it's amazing, but you don't need to exclude function words to make this work.

George Danezis: Because things like "verifiable secret sharing" are actually quite a good feature because many documents will have it but many more will not, but things like, "in order to", are not very good features.

Reply: Yes there are quite a few poor features, but there are many, many good ones, and the number of good features outweighs the number of poor features so substantially that the idea works very well.

Matt Blaze: This sounds very similar to the problem that the anti-spam people are trying to solve, the practice of being able to tell whether or not two email messages are substantially similar in order to be able to tell whether or not a message is coming from anybody else. Is it in fact a spam, or is it unique. I'm not intimately familiar with the literature in that area, but it would almost certainly would be a place that you can either draw from, or contribute to.

John Ioannidis: Another question is, are you more worried about false positives than false negatives? In the case of spam for example, I would rather get some more spam than miss a really important message.

Ross Anderson: One of the most impressive plagiarism detection papers appeared some years ago, Nevin Heintze wrote a paper[4] on this, and his technique was to match the documents by looking for the rarest words in each document, and having got the dozen or so words in each document, which were very rare in common English usage, he used these as an index, and this turns out to be very powerful. Now what you are doing by using triples is you're creating a very, very large number of very rare words by setting the word boundaries differently,

[4] "Scalable Document Fingerprinting", Second USENIX Electronic Commerce Workshop, pp 191-200, 1996.

and you don't have the convenience of the Heintze mechanism. You've a very large number of rare "words" namely most of the triples in each document, and so the issue that that would then raise is, what's the computational cost of this. Alright, you've given us an algorithm whereby I can write some scripts that will get Google to find in order of the ten closest documents in the world to the document that I am now looking at, but it's going to require an awful lot of references to Google to do that, and if everybody starts doing that, then Google might slow down.

Michael Roe: I've seen something similar to this before, one of the tricks you can do is you can say, hash each triple and have a rule that the hashes that have a certain pattern, like the first couple of bits equal to this value, I would use that as a pseudo random way of choosing which features in the many features of the documents you then put in your repository, select those features for the reference document and then try and compare something you've got.

Reply: Yes, Andrei Broder did something vaguely similar[5] but he was using ten word sequences and I think he only kept every tenth one to reduce storage requirements. But (a) it's not necessary to keep such a long string, and (b) if you're looking at ten word sequences it is only necessary to change every tenth word to make a different document.

Ross Anderson: If what we need is an ordering, we might as well use an order for the triples, the product of the probabilities of each of these words in general text rather than doing something pseudo-random, because that actually gives you some leverage over the probabilities.

Reply: I'm really not looking at the actual probabilities, just pointing out that most of these triples had not occurred before. When you sit down and write a document, mostly what you type is original[6] even if you're writing about the same thing that you wrote about yesterday, unless you've got a very good memory.

Birgit Pfitzmann: What about strong secrecy? I don't know if you want it at all, but suppose I am interested whether you are continuing to work in on-line publishing or if he is continuing to work on verifiable secret sharing. So I put "verifiable secret sharing" into the hash function, look what the hash would be, and then I look if it occurs in your set of hashes, so I could find out a certain amount of information about what a document is about. Does that cause a problem to your scheme?

Reply: Yes, I believe that's possible, but I don't believe it's possible to work out what the whole document says because there are too many next words. What is the next triple after "verifiable secret sharing"? Is it "secret sharing is", "secret sharing can", ... There are so many possible next words.

I say "the cat sat on the", and you think "mat", but only because you are plagiarising a well known sentence. "The cat sat on the not very comfortable chair".

[5] A.Z. Broder et al. Computer Networks and ISDN Systems **29** pp 1157-1166, 1997.

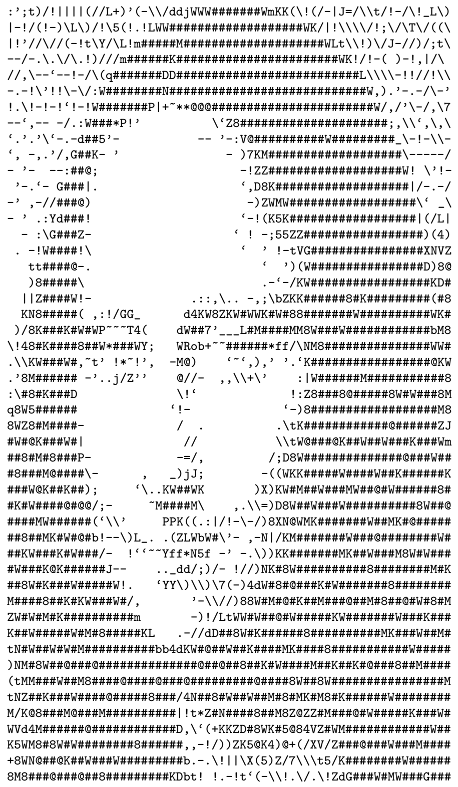
[6] i.e. the majority of the triple *types* have never occurred before.

If you do get a false positive, there's still no need to publish the document because we can repeat the publishing process with different hash function or a larger feature set, and also you can combine features.

As an example of this applied to text, two newspapers described the same event, probably from the same news wire as they probably weren't actually up there themselves, and they came up with very different ways of saying the same thing, giving only four matching trigrams in a short piece of text. In more substantial documents the proportion of matching trigrams drops.

Richard Clayton: Reuters are experimenting with a watermarking scheme at the moment whereby different people get slightly different feeds with the words changed around by Reuters in order to try and detect whether or not they're leaking the stuff when they shouldn't.

Reply: Yes, these kinds of ideas have possible applications in a number of areas. Images are harder. You probably recognise the following image, but how do you extract appropriate features from this, and compare them with the features in the Mona Lisa that we saw earlier. It's clearly very hard, but if one was looking specifically at JPEG files I think one might get some mileage by extracting DCT coefficients, but that's a bit of a half-baked idea as there are many possible attacks on that.



With audio I think there are features that we can extract: using FFT and similar techniques you can actually extract the notes from a piece of recorded music, and from the notes you can extract the intervals that were used, and it's fairly clear from work that's been done by other people in the past that quite simple features will characterise musical themes. Now there's a lot of noise in a piece of music: I don't mean background noise, but the various accompaniments as well as the tune. But certainly the tune can be characterised quite simply by whether the note is going up or down without even recording the intervals, so if we record more information, then the feature set might be better.

Matt Blaze: I think that in the case of music there are really two separate and equally interesting problems, one is to determine if two distinct performances are of the same music, and the other is to determine whether you have two distinct recordings of the same performance, and those are probably different problems.

Reply: So what feature set we use depends on the nature of what we're trying to protect, but in the applications I am considering it's not the raw bits, it's something different.

So, to conclude, if we hash the word triples in a document, then a bit vector representing those hash values can be published allowing us to identify similar documents without access to the document texts. If neither author is willing to reveal his text[7], then as Richard pointed out, we require trust in the publishing process, which is a weakness I haven't resolved[8]. It would be interesting to see to what extent this can be extended to images, probably pretty hard, and sounds, which I think relatively straightforward.

Matt Blaze: Is it a weakness that if you're using only word triples, and the dictionary you've got is small, I could build a table of all the word triples to act as an index for this; it would be large though computationally probably within range.

Reply: Yes, there's going to be a lot of collisions, so there's a lot of communal words.

Matt Blaze: Somewhere between five thousand and half a million.

Reply: It's not just dictionary words, because you have to take into account spelling mistakes particularly when you're looking for plagiarism.

Matt Blaze: Well you can crack DES.

Bruce Christianson: The use of hashing reduces that problem doesn't it? This is collision rich hashing.

Reply: Yes, you can hash onto much smaller bit vectors so that there would be a lot of collisions.

[7] LNCS 1796, p. 228.

[8] But the author can verify what the publishing module does, so hardware support might give us this trust.

Matt Blaze: What I'm concerned with is that, if what I am publishing is a genuinely secret document, it's possible that a triple in that document might itself be sensitive, and if I can build the reverse hash table that might in some applications be a weakness.

Simon Foley: There's an analogy I think between what you are doing with copyright and work on intrusion detection systems where experiments show that system-call execution profiles of executing programs can be characterised by a small number of n-grams[9].

Audience: As a useful pointer, have you talked with the people from Phillips? They're trying to solve this problem for audio[10].

John Ioannidis: When you showed the ASCII-art picture of the Mona Lisa, the way, I think, people realise that that was a picture of the Mona Lisa, the way they extract the features, is to employ a low pass filter. In fact if I take my glasses off I would recognise it as the Mona Lisa more easily.

Similarly the music comparison of seeing the pitch goes up or down, that is a very crude form of a low pass filter, looking at some very low frequency component of the music. Could we get some equivalent to a low pass filter and know what that means for text?

Audience: Word lengths.

Reply: Word lengths[11] might work, you'd have to try it. But the details of what works for any particular medium is not really what I wanted to get over; what I wanted to convey is that what certainly works for text is just to gather as many features as you can, and simply count how many of these features are common between two works.

John Ioannidis: The question is, how do we formalise what we mean by a feature in any particular medium?

Reply: Potentially much more work to do. Thank you.

Victoria Stavridou: What do your students think? Have you applied this?

Reply: Oh yes.

[9] S.A. Hofmeyr, S. Forrest and A. Somayaji: "Intrusion Detection Using Sequences of System Calls", Journal of Computer Security, **6**, 151–180, 1998.

[10] Jaap Haitsma, Ton Kalker and Job Oostveen: Robust Audio Hashing for Content Identification, CBMI 2001.

[11] Or T-line shorthand?

And Finally...

A Word from Our Sponsor

```
     0 0
    (---)
 /\( | | )/\
^^  ^^ ^^  ^^
```

```
                               W  W
                                \ \
                                 \ \
                                  \ \
                                 (    )
                                  (   )
                                 / 0 0 \
                                 \      \
                                 ^^      ^^
```

```
                                         Ribbit
                                            \

                                              0 0
                                             (---)
                                          /\( | | )/\
                                         ^^  ^^ ^^  ^^
```

Author Index