Generating a Product of Three Primes with an Unknown Factorization

Dan Boneh and Jeremy Horwitz

Computer Science Department, Stanford University, Stanford, CA 94305-9045
{dabo,horwitz}@cs.stanford.edu

Abstract. We describe protocols for three or more parties to jointly generate a composite $N = pqr$ which is the product of three primes. After our protocols terminate N is publicly known, but neither party knows the factorization of N. Our protocols require the design of a new type of distributed primality test for testing that a given number is a product of three primes. We explain the cryptographic motivation and origin of this problem.

1 Introduction

In this paper, we describe how three (or more) parties can jointly generate an integer N which is the product of three prime numbers ($N = pqr$). At the end of our protocol the product N is publicly known, but neither party knows the factorization of N. Our main contribution is a new type of probabilistic primality test that enables the three parties to jointly test that an integer N is the product of three primes without revealing the factorization of N. Our primality test simultaneously uses two groups: the group $\mathbb{Z}_N^*$ and the twisted multiplicative group $\mathbb{T}_N = (\mathbb{Z}_N[x]/(x^2+1))^*/\mathbb{Z}_N^*$.

The main motivation for this problem comes from cryptography, specifically the sharing of an RSA key. Consider classical RSA: $N = pq$ is a public modulus, e is a public exponent, and d is secret where $de = 1 \bmod \varphi(N)$. At a high level, a digital signature of a message M is obtained by computing $M^d \bmod N$. In some cases, the secret key d is highly sensitive (e.g. the secret key of a Certification Authority) and it is desirable to avoid storing it at a single location. Splitting the key d into a number of pieces and storing each piece at a different location avoids this *single point of failure*. One approach (due to Frankel [7]) is to pick three random numbers satisfying $d = d_1 + d_2 + d_3 \bmod \varphi(N)$ and store each of the shares d_1, d_2, d_3 at one of three different sites. To generate a signature of a message M, site i computes $S_i = M^{d_i} \bmod N$ for $i = 1, 2, 3$ and sends the result to a *combiner*. The combiner multiplies the S_i and obtains the signature $S = S_1 S_2 S_3 = M^d \bmod N$. If one or two of the sites are broken into, no information about the private key is revealed. An important property of this scheme is that it produces standard RSA signatures; the user receiving the signature is totally unaware of the extra precautions taken in protecting the private key. Note that during signature generation the secret key is never reconstructed at a single location.

To provide fault tolerance, one slightly modifies the above technique to enable any two of the three sites to generate a signature. This way if one of the sites is temporarily unavailable the Certification Authority can still generate signatures using the remaining two sites. If the key was only distributed among two sites, the system would be highly vulnerable to faults.

We point out that classic techniques of secret sharing [14] are inadequate in this scenario. Secret sharing requires one to reconstruct the secret at a single location before it can be used, hence introducing a single point of failure. The technique described above of sharing the secret key such that it can be used without reconstruction at a single location is known as *Threshold Cryptography*. See [9] for a succinct survey of these ideas and nontrivial problems associated with them.

An important question left out of the above discussion is key generation. Who generates the RSA modulus N and the shares d_1, d_2, d_3? Previously the answer was a *trusted dealer* would generate N and distribute the shares d_1, d_2, d_3 to the three sites. Clearly this solution is undesirable since it introduces a new single point of failure — the trusted dealer. It knows the factorization of N and the secret key d; if it is compromised, the secret key is revealed. Recently, Boneh and Franklin [2] designed a protocol that enables three (or more) parties to jointly generate an RSA modulus $N = pq$ and shares d_1, d_2, d_3 of a private key. At the end of the protocol the parties are assured that N is indeed the product of two large primes; however, none of them know its factorization. In addition, each party learns exactly one of d_1, d_2, d_3 and has no computational information about the other shares. Thus, there is no need for a trusted dealer. We note that Cocks [5] introduced a heuristic protocol enabling two parties to generate a shared RSA key.

In this paper we design an efficient protocol enabling three (or more) parties to generate a modulus $N = pqr$ such that neither party knows the factorization of N. Once N is generated the same techniques used in [2] can be used to generate shares d_1, d_2, d_3 of a private exponent. For this reason, throughout the paper we focus on the generation of the modulus $N = pqr$ and forgo discussion of the generation of the private key. The methods of [2] do not generalize to generate a modulus with three prime factors and new techniques had to be developed for this purpose. Our main results are described in Section 4

We remark that techniques of *secure circuit evaluation* [1, 4, 16] can also be used to solve this problem; however, these protocols are mostly theoretical and result in extremely inefficient algorithms.

2 Motivation

The problem discussed in the paper is a natural one and thus our solution is of independent interest. Nonetheless, the problem is well motivated by a method for improving the efficiency of shared generation of RSA keys. To understand this we must briefly recall the method used by Boneh and Franklin [2]. We refer to the three parties involved as Alice, Bob, and Carol. At a high level, to generate a modulus $N = pq$, the protocol works as follows:

Step 1. Alice picks two random n bit integers p_a, q_a, Bob picks two random n bit integers p_b, q_b and Carol picks two random n bit integers p_c, q_c. They keep these values secret.

Step 2. Using a private distributed computation they compute the value

$$N = (p_a + p_b + p_c)(q_a + q_b + q_c).$$

At the end of the computation, N is publicly available; however no other information about the private shares is revealed. This last statement is provable in an information-theoretic sense.

Step 3. The three parties perform a distributed primality test to test that N is the product of exactly two primes. As before, this step provably reveals no information about the private shares.

Step (3), the distributed primality test, is a new type of probabilistic primality test which is one of the main contributions of [2]. Step (2) is achieved using an efficient variation of the BGW [1] protocol.

A drawback of the above approach is that both factors of N are simultaneously tested for primality. Hence, the expected number of times step (3) is executed is $O(n^2)$. This is much worse than single-user generation of N where the two primes are first generated separately by testing $O(n)$ candidates and then multiplied together. When generating a 1024-bit modulus, this results in significant slowdown when compared with single-user generation.
To combat this quadratic slowdown, one may try the following alternate approach.

Step 1. Alice picks a random n-bit prime p and a random n-bit integer r_a. Bob picks a random n-bit prime q and a random n-bit integer r_b. Carol picks a random n-bit integer r_c. They keep these values secret.

Step 2. Using a private distributed computation they compute the value

$$N = pq(r_a + r_b + r_c)$$

At the end of the computation, N is publicly available; however no other information about the private shares is revealed.

Step 3. The three parties use the results of this paper to test that N is the product of exactly three primes. This step provably reveals no information about the private shares.

At the end of the protocol neither party knows the full factorization of N. In addition, this approach does not suffer from the quadratic slowdown observed in the previous method. Consequently, it is faster by roughly a factor of 50 (after taking effects of trial division into account). As before, step (2) is carried out by an efficient variant of the BGW protocol.

Instead of solving the specific problem of testing that $N = pq(r_a + r_b + r_c)$ is a product of three primes we solve the more general problem of testing that

$$N = (p_a + p_b + p_c)(q_a + q_b + q_c)(r_a + r_b + r_c)$$

is a product of three primes without revealing any information about the private shares. This primality test is the main topic of this paper.

For the sake of completeness we point out that in standard single-party cryptography there are several advantages to using an RSA modulus $N = pqr$ rather than the usual $N = pq$ (the size of the modulus is the same in both cases). First, signature generation is much faster using the Chinese Remainder Theorem (CRT). When computing $M^d \bmod N$ one only computes $M^{d \bmod p-1} \bmod p$ for all three factors. Since the numbers (and exponents) are smaller, signature generation is about twice as fast as using CRT with $N = pq$. Another advantage is that an attack on RSA due to Wiener [15] becomes less effective when using $N = pqr$. Wiener showed that for $N = pq$ if $d < N^{1/4}$ one can recover the secret key d from the public key. When $N = pqr$ the attack is reduced to $d < N^{1/6}$ and hence it may be possible to use smaller values of d as the secret key. Finally, we note that the fastest factoring methods [12] cannot take advantage of the fact that the factors of $N = pqr$ are smaller than those of a standard RSA modulus $N = pq$.

3 Preliminaries

In this section, we explain the initial setup for our new probabilistic primality test and how it is obtained. We then explain a basic protocol which we use in the later parts of the paper. At first reading, the reader may wish to skip to Section 4 and take on faith that the necessary setup is attainable.

3.1 Communication and Privacy Model

The communication and privacy model assumed by our protocol are as follows:

Full connectivity Any party can communicate with any other party. This is a typical setup on a local network or the Internet.

Private and authenticated channels Messages sent from party A to party B are private and cannot be tampered with en route. This simply states that A and B share a secret key which they can use for encryption and authentications.

Honest parties We assume all parties are honest in their following of the protocol. This is indeed the case when they are truly trying to create a shared key. This assumption is used by both [2] and [5]. We note that some recent work [8] makes the protocol of [2] robust against cheating adversaries at the cost of some slowdown in performance (roughly a factor of 100). These robustness results apply to the protocols described in this paper as well.

Collusion Our protocol is 1-private. That is to say that a single party learns no information about the factorization of $N = pqr$. However, if two of the three parties collude they can recover the factors. For three parties this is fine since our goal is to enable two-out-of-three signature generation. Hence, two parties are always jointly able to recover the secret key. More generally, when k parties participate in our primality test protocol, one can achieve $\lfloor \frac{k-1}{2} \rfloor$-privacy. That is, any minority of parties learns no information about the factors of N.

3.2 Generation of N

In the previous section, we explained that Alice, Bob, and Carol generate N as

$$N = (p_a + p_b + p_c)(q_a + q_b + q_c)(r_a + r_b + r_c),$$

where party i knows p_i, q_i, r_i for $i = a, b, c$ and keeps these shares secret while making N publicly available. To compute N without revealing any other information about the private shares, we use the BGW protocol [1]. For the particular function above, the protocol is quite efficient; it requires three rounds of communication and a total of six messages. The protocol is information-theoretically secure, i.e. other than the value of N, party i has no information about the shares held by other parties. This is to say the protocol is 1-private.

We do not go into the details of how the BGW protocol is used to compute N since it is tangential to the topic of this paper. For our purpose, it suffices to assume that N is public while the private shares are kept secret.

An important point is that our primality test can only be applied when $p_a + p_b + p_c = q_a + q_b + q_c = r_a + r_b + r_c = 3 \bmod 4$. Hence, the parties must coordinate the two lower bits of their shares ahead of time so that the sums are indeed 3 modulo 4. Indeed, this means that a priori each party knows the two least significant bits of the other's shares.

3.3 Sharing of $(p-1)(q-1)(r-1)$ and $(p+1)(q+1)(r+1)$

Let $p = p_a + p_b + p_c$, $q = q_a + q_b + q_c$, and $r = r_a + r_b + r_c$. We define $\hat{\varphi} = (p-1)(q-1)(r-1)$. Since p, q, and r are not necessarily prime, $\hat{\varphi}$ may not equal $\varphi(N)$. Our protocol requires that the value $\hat{\varphi}$ be shared additively among the three parties. That is, $\hat{\varphi} = \varphi_a + \varphi_b + \varphi_c$, where only party i knows φ_i for $i = a, b, c$.

An additive sharing of $\hat{\varphi}$ is achieved by observing that $\hat{\varphi} = N - pq - pr - qr + p + q + r - 1$. To share $\hat{\varphi}$, it suffices to represent $pq + pr + qr$ using an additive sharing $A + B + C$ among the three parties. The additive sharing of $\hat{\varphi}$ is then

$$\varphi_a = N - A + p_a + q_a + r_a - 1 \quad ; \quad \varphi_b = -B + p_b + q_b + r_b \quad ; \quad \varphi_c = -C + p_c + q_c + r_c.$$

The conversion of $pq + pr + qr$ into an additive sharing $A + B + C$ is carried out using a simple variant of the BGW protocol used in the computation of N. The

BGW protocol can be used to compute the value pq; however, instead of making the final result public the BGW variant shares the result additively among the three parties. The details of this variant can be found in [2, Section 6.2].

As before, we do not give the full details of the protocol for converting $pq + pr + qr$ into an additive sharing. Since we wish to focus on the primality test, we assume that an additive sharing of $\hat{\varphi}$ is available in the form of $\varphi_a + \varphi_b + \varphi_c$.

In addition to a sharing of $\hat{\varphi}$, we also require an additive sharing of $\hat{\psi} = (p+1)(q+1)(r+1)$. Once an additive sharing of $pq + pr + qr$ is available it is trivial to generate an additive sharing of $\hat{\psi}$. Simply set

$$\psi_a = N+A+p_a+q_a+r_a+1 \quad ; \quad \psi_b = B+p_b+q_b+r_b \quad ; \quad \psi_c = C+p_c+q_c+r_c.$$

3.4 Comparison Protocol

Our primality test makes use of what we call a *comparison protocol*. Let A be a value known to Alice, B a value known to Bob, and C a value known to Carol. We may assume $A, B, C \in \mathbb{Z}_N^*$. The protocol enables the three parties to test that $ABC = 1 \bmod N$ without revealing any other information about the product ABC. We give the full details of the protocol in this section.

Let $P > N$ be some prime known to all parties. The protocol proceeds as follows:

Step 1. Carol picks a random element $C_1 \in \mathbb{Z}_N^*$ and sets $C_2 = CC_1^{-1} \bmod N$. Clearly $C = C_1C_2 \bmod N$. Carol then sends C_1 to Alice and C_2 to Bob.

Step 2. Alice sets $A' = AC_1$ and Bob sets $B' = (BC_2)^{-1} \bmod N$. Both values A' and B' can be viewed as integers in the range $[0, N)$. The problem is now reduced to testing whether $A' = B'$ (as integers) without revealing any other information about A and B.

Step 3. Alice picks a random $c \in \mathbb{Z}_P^*$ and $d \in \mathbb{Z}_P$. She sends c and d to Bob. Alice then computes $h(A') = cA' + d \bmod P$ and sends the result to Carol. Bob computes $h(B') = cB' + d \bmod P$ and sends the result to Carol.

Step 4. Carol tests if $h(A') = h(B') \bmod P$. If so, she announces that $ABC = 1 \bmod N$. Otherwise she announces $ABC \neq 1 \bmod N$.

The correctness and privacy of the protocol are stated in the next two lemmata. Correctness is elementary and is stated without proof.

Lemma 1. *Let $A, B, C \in \mathbb{Z}_N^*$. At the end of the protocol, the parties correctly determine if $ABC = 1 \bmod N$ or $ABC \neq 1 \bmod N$.*

Lemma 2. *The protocol is 1-private. That is, other than the result of the test, each party learns no new information.*

Proof. To prove the protocol is 1-private, we provide a simulation argument for each party's view of the protocol. Alice's view of the protocol is made up of the values A, C_1, c, d, $h(A')$, and the final result of the test. These values can be easily simulated by picking C_1 at random in $\mathbb{Z}_N^*$, picking c at random in $\mathbb{Z}_P^*$ and

d at random in $\mathbb{Z}_P$. This is a perfect simulation of Alice's view. A simulation argument for Bob is the same, mutatis mutandis.

Simulating Carol's view is more interesting. Carol's view consists of C, C_1, C_2, $h(A')$, $h(B')$, and the result of the test. The key observation we make is that $h(A')$ and $h(B')$ reveal no information about A and B, since they are either equal or are random independent elements of $\mathbb{Z}_P$. Which of the two actually occurs is determined by the result of the test. The independence follows since the family of hash functions $h(x) = cx + d \bmod P$ is a universal family of hash functions (i.e. knowing neither c nor d, the values $h(x)$ and $h(y)$ are independent for any $x, y \in \mathbb{Z}_P$).

To simulate Carol's view, the simulator picks $C_1, C_2 \in \mathbb{Z}_N^*$ at random so that $C = C_1 C_2 \bmod N$. Then, depending on the results of the test, it either picks the same random element of $\mathbb{Z}_P$ twice or picks two random independent elements of $\mathbb{Z}_P$. This is a perfect simulation of Carol's view. This proves that Carol gains no extra information from the protocol since, given the outcome of the test, she can generate the values sent by Alice and Bob herself. $\square$

4 The Probabilistic Primality Test

We are ready to describe our main result, the probabilistic primality test. As discussed in the previous section, our primality test applies once the following setup is achieved:

Shares Each party i has three secret n-bit values p_i, q_i, and r_i for $i = a, b, c$.

The modulus $N = (p_a + p_b + p_c)(q_a + q_b + q_c)(r_a + r_b + r_c)$ is public. We set $p = p_a + p_b + p_c$, $q = q_a + q_b + q_c$, and $r = r_a + r_b + r_c$. Throughout this section, we are assuming that $p = q = r = 3 \bmod 4$. Thus, the parties must a priori coordinate the two least significant bits of their shares so that this condition holds.

Sharing $\hat{\varphi}, \hat{\psi}$ The parties share $(p-1)(q-1)(r-1)$ as $\varphi_a + \varphi_b + \varphi_c$ and $(p+1)(q+1)(r+1)$ as $\psi_a + \psi_b + \psi_c$.

Given this setup, they wish to test that p, q, and r are distinct primes without revealing any of p, q, and r. At this point, nothing is known about p, q, and r other than that $p = q = r = 3 \bmod 4$. Throughout the section, we use the following notation:

$$\hat{\varphi} = \varphi_a + \varphi_b + \varphi_c = (p-1)(q-1)(r-1)$$
$$\hat{\psi} = \psi_a + \psi_b + \psi_c = (p+1)(q+1)(r+1)$$

Clearly, if N is a product of three distinct primes, then $\varphi(N) = \hat{\varphi}$ and $\psi(N) = \hat{\psi}$. Otherwise, these equalities may not hold.

Our primality test is made up of four steps. In the subsequent subsections, we explain how each of these steps is carried out without revealing any information about the factors of N.

Probabilistic test that N is a product of three primes:

Step 1. The parties pick a random $g \in \mathbb{Z}_N^*$ and jointly test that $g^{\varphi_a+\varphi_b+\varphi_c} = 1 \bmod N$. If the test fails, N is rejected. This step reveals no information other than the outcome of the test. We refer to this step as a Fermat test in $\mathbb{Z}_N^*$ (see Section 4.2 for details).

Step 2. The parties perform a Fermat test in the twisted group $\mathbb{T}_N$ defined as $(\mathbb{Z}_N[x]/(x^2+1))^*/\mathbb{Z}_N^*$. Notice that x^2+1 is irreducible modulo N, since $p = q = r = 3 \bmod 4$.
If N is the product of three distinct primes then the order of $\mathbb{T}_N$ is $\psi(N) = (p+1)(q+1)(r+1)$. To carry out the Fermat test in $\mathbb{T}_N$, the parties pick a random $g \in \mathbb{T}_N$ and jointly test that $g^{\psi_a+\psi_b+\psi_c} = 1$ (see Section 4.2 for details). If the test fails, N is rejected. This step reveals no information other than the outcome of the test.

Step 3. The parties jointly test that N is the product of at most three prime powers. The implementation of this step is explained in Section 4.1. If the test fails, N is rejected.

Step 4. The parties jointly test that

$$\gcd(N, p+q+r) = 1.$$

This step reveals no information other than the outcome of the test. The implementation of this step is explained in Section 4.3. If the test fails, N is rejected. Otherwise, N is accepted as the product of three primes.

The following fact about the twisted group $\mathbb{T}_N = (\mathbb{Z}_N[x]/(x^2+1))^*/\mathbb{Z}_N^*$ is helpful in the proof of the primality test.

Fact 1. *Let N be an integer and k a prime such that $k^2 \big| N$. Then, k divides both $\varphi(N)$ and $|\mathbb{T}_N|$.*

Proof. Let $\alpha \geq 2$ be the number of times k divides N, i.e. $N = k^\alpha w$ where $\gcd(k, w) = 1$. Then $\varphi(N) = k^{\alpha-1}(k-1)\varphi(w)$ and, hence, k divides $\varphi(N)$.

To see that k divides $|\mathbb{T}_N|$, note that $\mathbb{T}_N \cong \mathbb{T}_{k^\alpha} \times \mathbb{T}_w$. When $k = 3 \bmod 4$, we know that x^2+1 is irreducible in $\mathbb{Z}_k$ and, hence, $|\mathbb{T}_{k^\alpha}| = k^{\alpha-1}(k+1)$. It follows that k divides $|\mathbb{T}_N|$. When $k = 1 \bmod 4$, we have $|\mathbb{T}_{k^\alpha}| = k^{\alpha-1}(k-1)$ and, again, k divides $|\mathbb{T}_N|$. $\square$

We can now prove that the aforementioned four steps are indeed a probabilistic test for proving that N is a product of three primes.

Theorem 2. *Let $N = pqr = (p_a+p_b+p_c)(q_a+q_b+q_c)(r_a+r_b+r_c)$, where $p = q = r = 3 \bmod 4$ and $\gcd(N, p+q+r) = 1$. If N is a product of three primes, it is always accepted. Otherwise, N is rejected with probability at least half. The probability is over the random choices made in steps 1–4 above.*

Proof. Suppose p, q, and r are distinct primes. Then, steps (1), (2), and (3) clearly succeed. Step (4) succeeds by assumption. Hence, in this case, N always passes the test (as required).

Suppose N is *not* the product of three distinct primes. Assume, for the sake of deriving a contradiction, that N passes all four steps with probability greater than $1/2$. Since N passes step (3) with probability greater than $1/2$ we know that $N = z_1^{\alpha_1} z_2^{\alpha_2} z_3^{\alpha_3}$ for three primes z_1, z_2, and z_3 (not necessarily distinct). Since N passes step (4) we know $\gcd(N, p+q+r) = 1$. Define the following two groups:

$$G = \left\{ g \in \mathbb{Z}_N^* \mid g^{\varphi_a + \varphi_b + \varphi_c} = 1 \right\} \text{ and}$$
$$H = \left\{ g \in \mathbb{T}_N \mid g^{\psi_a + \psi_b + \psi_c} = 1 \right\}.$$

Clearly, G is a subgroup of $\mathbb{Z}_N^*$ and H is a subgroup of the twisted group $\mathbb{T}_N$. By showing that at least one of G or H is a proper subgroup, we will prove that either step (1) or (2) fails with probability at least $1/2$. There are two cases to consider:

Case 1: p, q, and r are not pairwise relatively prime. Without loss of generality, we may assume that $\gcd(p,q) > 1$. Let k be a prime factor of $\gcd(p,q)$. Recall that N is odd, so $k > 2$ (since k divides N).
Since $N = pqr$ we know that $k^2 \big| N$. Hence, by Fact 1, $k \big| \varphi(N)$ and $k \big| |\mathbb{T}_N|$. We claim that either k does not divide $\hat{\varphi}$ or k does not divide $\hat{\psi}$. To see this, observe that if $k \big| \hat{\varphi}$ and $k \big| \hat{\psi}$, then k divides $\hat{\psi} - \hat{\varphi} = p(2q + 2r) + q(2r) + 2$. Since k divides both p and q, we conclude that $k \big| 2$, which contradicts $k > 2$.
First, we examine when k does not divide $\hat{\varphi}$. Since k is a prime factor of $\varphi(N)$, there exists an element $g \in \mathbb{Z}_N^*$ of order k. However, since k does not divide $\hat{\varphi}$ we know that $g^{\hat{\varphi}} \neq 1$. Hence, $g \notin G$ proving that G is a proper subgroup of $\mathbb{Z}_N^*$. If k does not divide $\hat{\psi}$ a similar argument proves that H is a proper subgroup of the twisted group $\mathbb{T}_N$.

Case 2: p, q, and r are pairwise relatively prime. We can write $p = z_1^{\alpha}$, $q = z_2^{\beta}$, and $r = z_3^{\gamma}$ with z_1, z_2, and z_3 distinct primes. By assumption, we know that one of α, β, or γ is greater than 1. Without loss of generality, we may assume $\alpha > 1$.
We first observe that none of the z_i can divide $\gcd(\hat{\varphi}, \hat{\psi})$. Indeed, if this were not the case, then z_i would divide $\hat{\varphi} + \hat{\psi} = 2(N + p + q + r)$. But then, since z_i divides N, it must also divide $p + q + r$, contradicting that $\gcd(N, p+q+r) = 1$ (as tested in step (4)).
We now know that either z_1 does not divide $\hat{\varphi}$ or it does not divide $\hat{\psi}$. However, since z_1^2 divides N, we obtain, by Fact 1, that $z_1 \big| \varphi(N)$ and $z_1 \big| |\mathbb{T}_N|$. We can now proceed as in case (1) to prove that either G is a proper subgroup of $\mathbb{Z}_N^*$ or H is a proper subgroup of $\mathbb{T}_N$.

□

Clearly, most integers N that are not a product of three primes will already fail step (1) of the test. Hence, steps (2–4) are most likely executed only once a good candidate N is found.

Notice that the condition $\gcd(N, p + q + r) = 1$ is necessary. Without it, the theorem is false as can be seen from the following simple example: $p = w^3$,

$q = aw^2 + 1$, and $r = bw^2 - 1$ where w, q, r are three odd primes with $p = q = r = 3 \bmod 4$. In this case, $N = pqr$ will always pass steps 1–3, even though it is not a product of three distinct primes.

4.1 Step 3: Testing that $N = p^\alpha q^\beta r^\gamma$

Our protocol for testing that N is a product of three prime powers borrows from a result of van de Graaf and Peralta [11]. Our protocol works as follows:

Step 0. From our construction of $\hat{\varphi}$, we know that it is divisible by 8. However, the individual shares φ_a, φ_b, and φ_c which sum to $\hat{\varphi}$ may not all be divisible by 8. To correct this, Alice generates two random numbers $a_1, a_2 \in \mathbb{Z}_8$ such that $a_1 + a_2 = \varphi_a \bmod 8$. She sends a_1 to Bob and a_2 to Carol. Alice sets $\varphi_a \leftarrow \varphi_a - a_1 - a_2$, Bob sets $\varphi_b \leftarrow \varphi_b + a_1$ and Carol set $\varphi_c \leftarrow \varphi_c + a_2$. Observe that, at this point,

$$\frac{\hat{\varphi}}{8} = \frac{\varphi_a}{8} + \left\lfloor \frac{\varphi_b}{8} \right\rfloor + \left\lceil \frac{\varphi_c}{8} \right\rceil .$$

Step 1. The parties first agree on eight random numbers $g_1, g_2, \ldots, g_8$ in $\mathbb{Z}_N^*$, all with Jacobi symbol $+1$.

Step 2. For $i, j \in \{1, 2, \ldots, 8\}$, we say that i is equivalent to j (this defines equivalence classes of $\{1, 2, \ldots, 8\}$) if

$$\left[\frac{g_i}{g_j}\right]^{\frac{\varphi_a+\varphi_b+\varphi_c}{8}} = 1 \pmod{N}.$$

Since all three parties know g_i and g_j, they can test if i is equivalent to j as follows:

1. Alice computes $A = (g_i/g_j)^{\varphi_a/8} \bmod N$,
   Bob computes $B = (g_i/g_j)^{\lfloor \varphi_b/8 \rfloor} \bmod N$, and
   Carol computes $C = (g_i/g_j)^{\lceil \varphi_c/8 \rceil} \bmod N$.

2. Using the comparison protocol of Section 3.4, they then test if $ABC = 1 \bmod N$. The comparison protocol reveals no information other than whether or not $ABC = 1 \bmod N$.

Step 3. If the number of equivalence classes is greater than four, N is rejected. Otherwise, N is accepted.

Testing that the number of equivalences classes is at most four requires at most twenty-two invocations of the comparison protocol. Note that we restrict our attention to the elements g_i with Jacobi symbol $+1$ for efficiency's sake. Without this restriction, the number of equivalence classes to check for is eight and, thus, many more applications of the comparison protocol would be necessary.

The following lemma shows that when N is a product of three distinct primes, it is always accepted. When N has more than three prime factors, it is rejected with probability at least $1/2$. Note that if N is a product of three prime powers, it will always be accepted by this protocol. We will use the following notation:

$$J = \left\{ g \in \mathbb{Z}_N^* \;\middle|\; \left(\frac{g}{N}\right) = +1 \right\}$$
$$Q = \left\{ g \in J \;\middle|\; g \text{ is a quadratic residue in } \mathbb{Z}_N^* \right\}$$

The index of Q in J is $2^{d(N)-1}$ or $2^{d(N)}$ (exactly when N is a perfect square), where $d(N)$ is the number of distinct prime factors of N.

Lemma 1. *Let $N = pqr$ be an integer with $p = q = r = 3 \bmod 4$. If p, q, and r are distinct primes, then N is always accepted. If the number of distinct prime factors of N is greater than three, then N is rejected with probability at least half.*

Proof. If N is the product of three distinct primes, then the index of Q in J is four. Two elements $g_1, g_2 \in \mathbb{Z}_N^*$ belong to the same coset of Q in J if and only if g_1/g_2 is a quadratic residue, i.e. if and only if $(g_1/g_2)^{\varphi(N)/8} = 1 \bmod N$. Since, in this case, $\varphi(N) = \hat{\varphi} = \varphi_a + \varphi_b + \varphi_c$, step (2) tests if g_i and g_j are in the same coset of Q. Since the number of cosets is four, there are exactly four equivalence classes and, thus, N is always accepted.

If N contains at least four distinct prime factors, we show that it is rejected with probability at least $1/2$. Define

$$\hat{Q} = \left\{ g \in J \;\middle|\; g^{\hat{\varphi}/8} = 1 \pmod N \right\}.$$

Since, in this case, $\hat{\varphi}$ may not equal $\varphi(N)$, the group $\hat{Q}$ need not be the same as the group Q.

We now show that the index of $\hat{Q}$ in J is at least eight. Since $p = q = r = 3 \bmod 4$, we know that $\hat{\varphi}/8$ is odd (since $\hat{\varphi} = (p-1)(q-1)(r-1)$). Notice that if $g \in J$ satisfies $g^x = 1$ for some odd x, g must be a quadratic residue (a root is $g^{(x+1)/2}$). Hence, $\hat{Q} \subseteq Q$ and hence is a subgroup of Q. Since the index of Q in J is at least eight, it follows that the index of $\hat{Q}$ in J is at least eight.

It remains to show that when the index of $\hat{Q}$ in J is at least eight, N is rejected with probability at least $1/2$. In step (2), two elements $g_1, g_2 \in J$ are said to be equivalent if they belong to the same coset of $\hat{Q}$ in J. Let R be the event that all 8 elements $g_i \in J$ chosen randomly in step (1) fall into only four of the eight cosets. Then

$$\Pr[R] \leq \binom{8}{4} \cdot \left(\frac{1}{2}\right)^8 \approx 0.27 < \frac{1}{2}.$$

N is accepted only when the event R occurs. Since R occurs with probability less than $1/2$, the number N is rejected with probability at least $1/2$ as, required. □

Next, we prove that the protocol leaks no information when N is indeed the product of three distinct primes. In case N is not of this form, the protocol may leak some information; however, in this case, N is discarded and is of no interest. To prove that the protocol leaks no information we rely on a classic cryptographic assumption (see [3]) called *Quadratic Residue Indistinguishability* (QRI). This cryptographic assumption states that when $N = pq$ with $p = q = 3 \bmod 4$, no polynomial time algorithm can distinguish between the groups J and Q defined above. In other words, for any polynomial time algorithm $\mathcal{A}$ and any constant $c > 0$,

$$\left| \Pr_{g \in J}[\mathcal{A}(g) = \text{"yes"}] - \Pr_{g \in Q}[\mathcal{A}(g) = \text{"yes"}] \right| < \frac{1}{(\log N)^c}.$$

Lemma 2. *If N is a product of three distinct primes, then the protocol is 1-private, assuming QRI.*

Proof Sketch. To prove that each party learns no information other than that N is a product of three prime powers, we provide a simulation argument. We show that each party can simulate its view of the protocol; hence, whatever values it receives from its peers, it could have generated itself. By symmetry, we need only consider Alice. Alice's view of the protocol consists of the elements $g_1, g_2, \ldots, g_8$ and bit values $b_{i,j}$ indicating whether $(g_i/g_j)^{\hat{\varphi}} = 1$ (recall that we already gave a simulation algorithm for the comparison protocol in Section 3.4). Thus, Alice learns whether or not each g_i/g_j is a quadratic residue. We argue that, under QRI, this provides no computational information (since it can be simulated). To simulate Alice's view, the simulation algorithm works as follows: it picks eight random elements $g_1, g_2, \ldots, g_8 \in J$. It then randomly associates with each g_i a value in the set $\{0, 1, 2, 3\}$. This value represents the coset of Q in which g_i lies. The simulator then says that g_i/g_j is a quadratic residue if and only if the value associated with g_i is equal to that associated with g_j. Under QRI, the resulting distribution on $g_1, g_2, \ldots, g_8, b_{1,1}, b_{1,2}, \ldots, b_{8,8}$ is computationally indistinguishable from Alice's true view of the protocol.

We note that the value $a_1 \in \mathbb{Z}_8$ that Alice sends Bob in step (0) is an element of $\mathbb{Z}_8$ chosen uniformly at random. Hence, Bob can simulate it trivially. Similarly, Carol can trivially simulate $a_2 \in \mathbb{Z}_8$. □

4.2 Implementing a Fermat Test with No Information Leakage

We briefly show how to implement a Fermat test in $\mathbb{Z}_N^*$ without leaking any extra information about the private shares. The exact same method works in the twisted group $\mathbb{T}_N$ as well.
To check that $g \in \mathbb{Z}_N^*$ satisfies $g^{\varphi_a + \varphi_b + \varphi_c} = 1 \bmod N$, we perform the following steps:

Step 1. Each party computes $R_i = g^{\varphi_i} \bmod N$ $(i = a, b, c)$.
Step 2. They test that $R_a R_b R_c = 1 \bmod N$ by simply revealing the values R_1, R_2, and R_3. Accept N if the test succeeds. Otherwise, reject.

Clearly, the protocol succeeds if and only if $g^{\hat{\varphi}} = 1 \bmod N$. We show that it leaks no other information.

Lemma 3. *If $N = pqr$ is the product of three distinct primes, then the protocol is 2-private.*

Proof. We show that no two parties learn any information about the private share of the third (other than that $g^{\hat{\varphi}} = 1 \bmod N$). By symmetry, we may restrict our attention to Alice and Bob. Since, by assumption, N is the product of three distinct primes, we know that $g^{\hat{\varphi}} = 1 \bmod N$. Hence, $g^{\varphi_a+\varphi_b} = g^{-\varphi_c}$. To simulate the value received from Carol, the simulation algorithm simply computes $R_c = g^{-\varphi_a-\varphi_b}$. Indeed, this is a perfect simulation of Alice and Bob's view. Thus, they learn nothing from Carol's message since they could have generated it themselves. □

4.3 Step 4: Zero-Knowledge Test that $\gcd(N, p + q + r) = 1$

Our protocol for this step is based on a protocol similar to the one used in the computation of N. We proceed as follows:

Step 1. Alice picks a random $y_a \in \mathbb{Z}_N$, Bob picks a random $y_b \in \mathbb{Z}_N$, and Carol picks a random $y_c \in \mathbb{Z}_N$.

Step 2. Using the BGW protocol as in Section 3.2, they compute

$$R = (p_a + q_a + r_a + p_b + q_b + r_b + p_c + q_c + r_c)(y_a + y_b + y_c) \bmod N.$$

At the end of the protocol, R is publicly known; however, no other information about the private shares is revealed.

Step 3. Now that R is public, the parties test that $\gcd(R, N) = 1$. If not, N is rejected. Otherwise, N is accepted.

Lemma 4. *If N is the product of three distinct n-bit primes p, q, and r, with $\gcd(N, p + q + r) = 1$, then N is accepted with probability $1 - \epsilon$ for some $\epsilon < 1/2^{n-3}$. Otherwise, N is always rejected.*

Proof. Clearly, if $\gcd(N, p + q + r) > 1$ then $\gcd(R, N) > 1$ and therefore N is always rejected. If $\gcd(N, p + q + r) = 1$, then N is rejected only if $\gcd(N, y_a + y_b + y_c) > 1$. Since $y_a + y_b + y_c$ is a random element of $\mathbb{Z}_N$, this happens with probability less than $(1/2)^{n-3}$. □

Lemma 5. *If N is the product of three distinct n-bit primes p, q, and r, with $\gcd(N, p + q + r) = 1$, then the protocol is 1-private.*

Proof. Note that, since the BGW protocol is 1-private, the above protocol can be at best 1-private. By symmetry, we need only show how to simulate Alice's view. Alice's view consists of her private shares p_a, q_a, y_a and the number R. Since R is independent of her private shares, the simulator can simulate Alice's view by simply picking R in $\mathbb{Z}_N$ at random. This is a perfect simulation. □

5 Extensions

One can naturally extend our protocols in two ways. First, one may allow more than three parties to generate a product of three primes with an unknown factorization. Second, one may wish to design primality tests for testing that N is a product of k primes for some small k. We briefly discuss both extensions below.

Our protocols easily generalize to allow any number of parties. When k parties are involved, the protocols can be made $\lfloor \frac{k-1}{2} \rfloor$-private. This is optimal in the information-theoretic sense and follows from the privacy properties of the BGW protocol. The only complexities in this extension are the comparison protocol of Section 3.4 and Step (0) of Section 4.1. Both protocols generalize to k parties; however, they require a linear (in k) number of rounds of communication.

Securely testing that N is a product of k primes for some fixed $k > 3$ seems to be harder. Our results apply when $k = 4$ (indeed Theorem 2 remains true in this case). For $k > 4$, more complex algorithms are necessary. This extension may not be of significant interest since it is not well motivated and requires complex protocols.

Another natural question is whether only two parties can generate a product of three primes with an unknown factorization. The answer appears to be yes, although the protocols cannot be information-theoretically secure. Essentially, one needs to replace the BGW protocol for computing N with a two-party private multiplication protocol. This appears to be possible using results of [5].

6 Conclusions and Open Problems

Our main contribution is the design of a probabilistic primality test that enables three (or more) parties to generate a number N with an unknown factorization and test that N is the product of three distinct primes. The correctness of our primality test relies on that we simultaneously work in two different subgroups of $\mathbb{Z}_N[x]/(x^2+1)^*$, namely $\mathbb{Z}_N^*$ and the twisted multiplicative group $\mathbb{T}_N$. Our protocol generalizes to an arbitrary number of parties k and achieves $\lfloor \frac{k-1}{2} \rfloor$-privacy — the best possible in an information-theoretic setting.

Recall that our primality test can be applied to $N = pqr$ whenever $p = q = r = 3 \bmod 4$. We note that simple modifications enable one to apply the test when $p = q = r = 1 \bmod 4$ (essentially, this is done by reversing the roles of $\mathbb{Z}_N^*$ and the twisted group). However, it seems that one of these restrictions is necessary; we do not know how to carry out the test without the assumption that $p = q = r \bmod 4$. The assumption plays a crucial role in the proof of Lemma 1.

A natural question to ask is whether more advanced primality testing techniques can be used to improve the efficiency of our test. For instance, recent elegant techniques due to Grantham [10] may be applicable in our scenario as well.

References

1. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, 1988.
2. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Proceedings of Advances in Cryptology: CRYPTO '97*, pages 425–439. Lecture Notes in Computer Science, Springer-Verlag, New York, 1998.
3. M. Blum and S. Goldwasser. An efficient probabilistic public key encryption scheme that hides all partial information. In *Proceedings of Advances in Cryptology: CRYPTO '84*, pages 289–302. Lecture Notes in Computer Science, Springer-Verlag, New York, 1985.
4. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM Press, 1988.
5. C. Cocks. Split knowledge generation of RSA parameters. Available from the author (`cliff_cocks@cesg.gov.uk`).
6. R. Fagin, M. Naor, and P. Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(5):77–85, May 1996.
7. Y. Frankel. A practical protocol for large group oriented networks. In *Proceedings of Advances in Cryptology: EUROCRYPT '88*, pages 56–61. Lecture Notes in Computer Science, Springer-Verlag, New York, 1990.
8. Y. Frankel, P. MacKenzie, and M. Yung. Robust efficient distributed RSA key generation. Preprint.
9. P. Gemmel. An introduction to threshold cryptography. *CryptoBytes* (a technical newsletter of RSA Laboratories), 2(7), 1997.
10. J. Grantham. A probable prime test with high confidence. Available online (`http://www.clark.net/pub/grantham/pseudo/`).
11. R. Peralta and J. van de Graaf. A simple and secure way to show the validity of your public key. In *Proceedings of Advances in Cryptology: CRYPTO '87*, pages 128–134. Lecture Notes in Computer Science, Springer-Verlag, New York, 1988.
12. A. Lenstra and H. W. Lenstra ed. The development of the number field sieve. Lecture Notes in Computer Science 1554, Springer-Verlag, 1994.
13. H. W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.
14. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.
15. M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553-558, 1990.
16. A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 162–167. IEEE Press, 1986.