

# Efficient Generation of Shared RSA Keys

(Extended Abstract)

Dan Boneh<sup>1</sup>  
dabo@bellcore.com

Matthew Franklin<sup>2</sup>  
franklin@research.att.com

<sup>1</sup> Bellcore, 445 South St., Morristown, NJ, 07960, USA

<sup>2</sup> AT&T Labs, 180 Park Ave., Florham-Park, NJ, 07932, USA

**Abstract.** We describe efficient techniques for three (or more) parties to jointly generate an RSA key. At the end of the protocol an RSA modulus  $N = pq$  is publicly known. None of the parties know the factorization of  $N$ . In addition a public encryption exponent is publicly known and each party holds a share of the private exponent that enables threshold decryption. Our protocols are efficient in computation and communication.

**Keywords:** RSA, Threshold Cryptography, Primality testing, Multiparty computation.

## 1 Introduction

We propose efficient protocols for three (or more) parties to jointly generate an RSA modulus  $N = pq$  where  $p, q$  are prime. At the end of the computation the parties are convinced that  $N$  is indeed a product of two large primes. However, none of the parties know the factorization of  $N$ . We then show how the parties can proceed to compute a public exponent  $e$  and shares of the corresponding private exponent.

Our techniques require a number of steps including a new distributed primality test. The test enables two (or more) parties to test that a random integer  $N$  is a product of two large primes without revealing the primes themselves.

Several cryptographic protocols require an RSA modulus  $N$  for which none of the participants know the factorization. For examples see [11, 12, 14, 19, 20, 21]. Usually this is done by asking a dealer to generate  $N$ . Clearly, the dealer must be trusted not to reveal the factorization of  $N$ . Our results eliminate the need for a trusted dealer since the parties can generate the modulus  $N$  by themselves.

Threshold cryptography is a concrete example where shared generation of RSA keys is very useful. We give a brief motivating discussion and refer to [9] for a survey. A threshold RSA signature scheme involves  $k$  parties and enables any subset of  $t$  of them to generate an RSA signature of a given message. No subset of  $t - 1$  parties can generate a signature. A complete solution to this problem was given in [8]. Unfortunately, the modulus  $N$  and the shares of the private key were assumed to be generated by a dealer. The dealer, or anyone who

compromises the dealer, can forge signatures. Our results eliminate the need for a trusted dealer (as long as  $t < \lceil k/2 \rceil$ ) since the  $k$  parties can generate  $N$  and the private shares themselves. Such results were previously known for the ElGamal public key system [22], but not for RSA.

We note that generic secure circuit evaluation techniques, e.g. [26, 17, 3, 6] can also be used to generate shared RSA keys. After all, a primality test can be represented as a boolean circuit. However, such general techniques are usually too inefficient.

Our protocols are useful even when only two parties are involved. However, some steps of the protocol require the parties to interact with a third “helper” party we call Henry. At the end of the protocol Henry learns nothing, but the value of  $N$  which is public. To simplify the exposition we first describe our results for the case of two parties with a third helper (Sections 2-6). In Section 7, we explain how our methods generalize to more parties. An overview of our techniques is given in Section 2, and the various stages of the protocol are given in Sections 3-6.

## 2 Overview

In this section we give a high level overview of the protocol. The parties are Alice and Bob, with a third helper party Henry (see Section 7 for a generalization to more parties). Alice and Bob wish to generate a shared RSA key. More precisely, they wish to generate an RSA modulus  $N = pq$  and a public/private pair of exponents  $e, d$ . At the end of the computation  $N$  and  $e$  are public, and  $d$  is shared between Alice and Bob in a way which enables threshold decryption. Alice and Bob should be convinced that  $N$  is indeed a product of two primes, but neither of them know the factorization of  $N$ .

We assume a model of passive adversaries, i.e. all three parties follow the protocol as required. At the end of the protocol no party is able to factor  $N$ . We discuss the case of active adversaries at the end of the paper.

At a high level the protocol works as follows:

- (1) **pick candidates:** The following two steps are repeated twice.
  - (a) **secret choice:** Alice and Bob pick random  $n$ -bit integers  $p_a$  and  $p_b$  respectively, and keep them secret.
  - (b) **trial division:** Using a private distributed computation Alice, Bob and Henry determine that  $p_a + p_b$  is not divisible by small primes. If this step fails repeat step (a).

Denote the secret values picked at the first iteration by  $p_a, p_b$ , and at the second iteration by  $q_a, q_b$ .

- (2) **compute  $N$ :** Using a private distributed computation Alice, Bob and Henry compute

$$N = (p_a + p_b) \cdot (q_a + q_b)$$

Other than the value of  $N$ , this step reveals no further information about the secret values  $p_a, q_a, p_b, q_b$ .

- (3) **primality test:** Alice and Bob (without Henry) engage in a private distributed computation to test that  $N$  is indeed the product of two primes. If the test fails, then the protocol is restarted from step 1.
- (4) **key generation:** Alice and Bob engage in a private distributed computation to generate a public encryption exponent  $e$  and a shared secret decryption exponent  $d$ .

**Notation** Throughout the paper we adhere to the following notation: the RSA modulus is denoted by  $N$  and is a product of two  $n$  bit primes  $p, q$ . When  $p = \sum p_i$  we denote by  $p_i$  the share in possession of party  $i$ . Similarly for  $q_i$ . When the  $p_i$ 's themselves are shared among the parties we denote by  $p_{i,j}$  the share of  $p_i$  that is sent to party  $j$ .

**Performance issues** Our protocol generates two random numbers and tests that  $N = pq$  is a product of two primes. By the prime number theorem the probability that both  $p$  and  $q$  are prime is asymptotically  $1/n^2$ . Therefore, naively one has to perform  $n^2$  probes on average until a suitable  $N$  is found. This is somewhat worse than the expected  $2n$  probes needed in traditional generation of an RSA modulus (one first generates one prime using  $n$  probes and then a second prime using another  $n$  probes). This  $n/2$  degradation in performance is usually unacceptable (typically  $n = 512$ ).

Fortunately, thanks to trial division things aren't so bad. Our trial division tests each prime individually. Therefore, to analyze our protocol we must analyze the effectiveness of trial division. Suppose a random  $n$ -bit number  $p$  passes the trial division test where all primes less than  $B$  are tested. We take  $B = c \cdot n$  for some constant  $c$ . How likely is  $p$  to be prime? Using a classic result due to Mertens, DeBruijn [7] shows that asymptotically

$$\Pr[p \text{ prime} \mid \text{trial division up to } B] = e^\gamma \frac{\ln B}{\ln 2^n} (1 + o(1/n)) = 2.57 \frac{\ln B}{n} (1 + o(1/n))$$

Hence, when  $n = 512$  bits and  $\ln B = 9$  (i.e.  $B = 8103$ ) the probability that  $p$  is prime is approximately  $1/22$ . Consequently, traditional RSA modulus generation requires 44 probes while our protocol requires 484 probes. This eleven fold degradation in performance is unfortunate, but manageable.

**Generation of shares** In step (1) of the protocol each of Alice and Bob uniformly picked a random  $n$  bit integer  $p_a, p_b$  as its secret share. The prime  $p$  was taken to be the sum of these shares. Since the sum of uniform independent random variables over the integers is *not* uniformly distributed,  $p$  is picked from a distribution with slightly less entropy than uniform. We show that this is not a problem. For the generalization to  $k$  parties, each party  $i$  uniformly picks a random  $n$  bit integer  $p_i$ . Then  $p = \sum_i p_i$  is at most an  $n + \log k$  bit number. One can easily show that  $p$  is chosen from a distribution with at least  $n$  bits of entropy (since the  $n$  least significant bits of  $p$  are a uniformly chosen  $n$  bit string). Intuitively, these  $\log k$  bits of "lost" entropy can not help an adversary, since they can be easily guessed (the number of parties  $k$  is small, certainly  $k < n$ ). This is formally stated in the next lemma. We note that by allowing some

communication between the parties it is possible to ensure that  $p$  is uniformly distributed among “most”  $n$  bit integers.

A second issue is the fact that the shares themselves leak some information about the factors of  $N$ . For instance, party  $i$  knows that  $p > p_i$ . We argue that this information does not help an adversary either.

The two issues raised above are dealt with in the following lemma. Due to space limitation we leave the proof for the full version. Let  $\mathbb{Z}_n^{(2)}$  be the set of RSA moduli  $N = pq$  that can be output by our protocol above when  $k$  parties are involved. We assume  $k < \log N$ .

**Lemma 1.** *Suppose there exists a polynomial time algorithm  $\mathcal{A}$  that given a random  $N \in \mathbb{Z}_n^{(2)}$  chosen from the distribution above and the shares  $\{p_i\}$  of  $k-1$  parties, factors  $N$  with probability at least  $1/n^d$ . Then there exists an expected polynomial time algorithm  $\mathcal{B}$  that factors  $1/n^{d+2}$  of the integers in  $\mathbb{Z}_n^{(2)}$ .*

### 3 Distributed primality test

We now consider the distributed primality test. We describe our protocol for the case of two parties, and discuss the case of  $k > 2$  parties in Section 7.

In the case of two parties, Alice and Bob possess integers  $p_a, q_a$  and  $p_b, q_b$  respectively. Both parties know  $N$ , where  $N = (p_a + p_b)(q_a + q_b)$ . They wish to determine if  $N$  is the product of two primes. The primality test is a mix of the Solovay-Strassen [24] and the Rabin-Miller [23] primality tests. We assume that the secret values chosen by the parties satisfy  $p_a = q_a = 3 \pmod 4$  and  $p_b = q_b = 0 \pmod 4$ . This can be agreed upon before hand and causes the resulting modulus  $N$  to be a Blum integer<sup>3</sup>, since  $p \equiv q \equiv 3 \pmod 4$ . The test is as follows:

1. Alice and Bob agree on a random  $g \in \mathbb{Z}_N^*$ .
2. Alice computes the Jacobi symbol of  $g$  over  $N$ . If  $(\frac{g}{N}) \neq 1$  the protocol is restarted at step (1).
3. Otherwise, Alice computes  $v_a = g^{(N-p_a-q_a+1)/4} \pmod N$ , and Bob computes  $v_b = g^{(p_b+q_b)/4} \pmod N$ . They exchange these values, and verify that

$$v_a = \pm v_b \pmod N$$

If the test fails then the parties declare that  $N$  is not a product of two primes. Otherwise they declare success.

Since  $p_a = q_a = 3 \pmod 4$  and  $p_b = q_b = 0 \pmod 4$  both exponents in the computation of  $v_a, v_b$  are integers after division by 4. The correctness and privacy of the protocol is proved in the next two lemmas.

<sup>3</sup> The primality test described in this section is best suited for Blum integers. For non-Blum moduli the test may leak a few bits of information depending on the power of two dividing  $\text{lcm}(p-1, q-1)$ . For non-Blum integers  $N = pq$  with  $p = q = 1 \pmod 4$  these problems can be avoided by performing the test in a different group (i.e. not in  $\mathbb{Z}_N^*$ ). We use a quadratic extension of  $\mathbb{Z}_N$ .

**Lemma 2.** *Let  $N = pq$  be an integer with  $p \equiv q \equiv 3 \pmod 4$ . If  $N$  is a product of two distinct primes then success is declared in all invocations of the protocol. Otherwise, for all but an exponentially small fraction of  $N$ , the parties declare that  $N$  is not a product of two primes with probability at least  $\frac{1}{2}$  (over the choice of  $g$ ).*

*Proof.* Observe that the test in step (3) of the protocol checks that  $g^{(N-p-q+1)/4} \equiv \pm 1 \pmod N$ .

Suppose  $p$  and  $q$  are prime. In step (2) we verify that  $\left(\frac{g}{N}\right) = 1$ . This implies  $\left(\frac{g}{p}\right) = \left(\frac{g}{q}\right)$ . Also, since  $\frac{q-1}{2}$  and  $\frac{p-1}{2}$  are odd we have,

$$\begin{cases} g^{\phi(N)/4} = \left(g^{\frac{p-1}{2}}\right)^{\frac{q-1}{2}} \equiv \left(\frac{g}{p}\right)^{\frac{q-1}{2}} = \left(\frac{g}{p}\right) \pmod p \\ g^{\phi(N)/4} = \left(g^{\frac{q-1}{2}}\right)^{\frac{p-1}{2}} \equiv \left(\frac{g}{q}\right)^{\frac{p-1}{2}} = \left(\frac{g}{q}\right) \pmod q \end{cases}$$

Since  $\left(\frac{g}{p}\right) = \left(\frac{g}{q}\right)$  it follows that  $g^{\phi(N)/4} \equiv \pm 1 \pmod N$ . Since  $\phi(N) = N - p - q + 1$  when  $p$  and  $q$  are prime, it follows that the test in step 3 always succeeds.

Suppose at least one of  $p, q$  is not prime. That is,  $N = r_1^{d_1} \dots r_s^{d_s}$  is a non-trivial factorization of  $N$  with  $\sum d_i \geq 3$  and  $s \geq 1$ . Set  $e = (N - p - q + 1)/4 = (p - 1)(q - 1)/4$  to be the exponent used in step (3). Note that  $e$  is odd since  $p \equiv q \equiv 3 \pmod 4$ . Define the following two subgroups of  $\mathbb{Z}_N^*$ :

$$G = \{g \in \mathbb{Z}_N \text{ s.t. } \left(\frac{g}{N}\right) = 1\} \quad \text{and} \quad H = \{g \in G \text{ s.t. } g^e = \pm 1 \pmod N\}$$

To prove the lemma we show that  $|H| \leq \frac{1}{2}|G|$ . Since  $H$  is a subgroup of  $G$  it suffices to prove proper containment of  $H$  in  $G$ , i.e. prove the existence of  $g \in G \setminus H$ . There are four cases to consider.

**Case 1.** Suppose  $s \geq 3$ . Let  $a$  be a quadratic non-residue modulo  $r_3$ . Define  $g \in \mathbb{Z}_N$  to be an element satisfying

$$g \equiv 1 \pmod{r_1} \quad \text{and} \quad g \equiv -1 \pmod{r_2} \quad \text{and} \quad g \equiv \begin{cases} 1 \pmod{r_3} \text{ if } \left(\frac{-1}{r_2}\right) = 1 \\ a \pmod{r_3} \text{ if } \left(\frac{-1}{r_2}\right) = -1 \end{cases}$$

and  $g \equiv 1 \pmod{r_i}$  for  $i > 3$ . Observe that  $g \in G$ . Since  $e$  is odd  $g^e = g = 1 \pmod{r_1}$  and  $g^e = g = -1 \pmod{r_2}$ . Consequently,  $g^e \not\equiv \pm 1 \pmod N$  i.e.  $g \notin H$ .

**Case 2.** Suppose  $\gcd(p, q) > 1$ . Then there exists an odd prime  $r$  such that  $r$  divides both  $p$  and  $q$ . Then  $r^2$  divides  $N$  implying that  $r$  divides  $\phi(N)$ . It follows that in  $\mathbb{Z}_N^*$  there exists an element  $g$  of order  $r$ . Since  $r$  is odd we have  $\left(\frac{g}{N}\right) = \left(\frac{g^r}{N}\right) = \left(\frac{1}{N}\right) = 1$ , i.e.  $g \in G$ . Since  $r$  divides both  $p$  and  $q$  we know that  $r$  does not divide  $N - p - q + 1 = 4e$ . Consequently  $g^{4e} \not\equiv 1 \pmod N$  implying that  $g^e \not\equiv \pm 1 \pmod N$ . Hence,  $g \notin H$ .

**Case 3.** The only way  $N = pq$  does not fall into both cases above is if  $p = r_1^{d_1}$  and  $q = r_2^{d_2}$  where  $r_1, r_2$  are distinct primes and at least one of  $d_1, d_2$  is bigger than 1 (case 2 handles  $N$  that are a prime power  $N = r^d$ ). By symmetry we may assume  $d_1 > 1$ . Since  $\mathbf{Z}_p^*$  is a cyclic group of order  $r_1^{d_1-1}(r_1 - 1)$  it contains an element of order  $r_1^{d_1-1}$ . It follows that  $\mathbf{Z}_N^*$  also contains an element  $g$  of order  $r_1^{d_1-1}$ . As before,  $\left(\frac{g}{N}\right) = 1$ , i.e.  $g \in G$ . If  $q \neq 1 \pmod{r_1^{d_1-1}}$  then  $4e = N - p - q + 1$  is not divisible by  $r_1^{d_1-1}$ . Consequently,  $g^{4e} \neq 1 \pmod{N}$ , i.e.  $g \notin H$ .

**Case 4.** We are left with the case  $N = pq$  with  $p = r_1^{d_1}$ ,  $q = r_2^{d_2}$ ,  $d_1 > 1$  as above and  $q \equiv 1 \pmod{r_1^{d_1-1}}$ . In this case it may indeed happen that  $H = G$ . For example,  $p = 3^n$  and  $q = 2 \cdot 3^{n-1} + 1$  with  $n$  odd and  $q$  prime. Observe that  $r_1^{d_1-1} \geq \sqrt{p} \geq 2^{n/2}$ . Consequently, since  $p$  and  $q$  are chosen independently the probability of  $q \equiv 1 \pmod{r_1^{d_1-1}}$  is less than  $1/2^{n/2}$ . In addition,  $p$  has to be a prime power which happens with probability less than  $n/2^{n/2}$ . The probability that both events happen is less than  $n/2^n$ . Hence, this case occurs with exponentially small probability.  $\square$

Integers  $N$  that fall into Case 4 above incorrectly pass the test. This can be rectified by adding a fourth step to the protocol to filter out these integers. With this extra fourth step our protocol becomes a complete probabilistic test for proving that  $N$  is a product of two primes. Due to space limitation we only give a high level description in the next subsection.

**Lemma 3.** *Suppose  $p, q$  are prime. Then either party can simulate the transcript of the primality testing protocol. Consequently, neither party learns anything about the factors of  $N$  from this protocol.*

*Proof Sketch.* Since  $p, q$  are prime we know that  $v_a = \pm v_b \pmod{N}$  where  $v_a, v_b$  are defined as in step (3) of the protocol. Consequently, given either of  $v_a$  or  $v_b$ , the simulator can compute the other one up to sign. If  $v_a = v_b$  then  $\left(\frac{g}{p}\right) = \left(\frac{g}{q}\right) = 1$ , and if  $v_a = -v_b$  then  $\left(\frac{g}{p}\right) = \left(\frac{g}{q}\right) = -1$ . That is, the sign determines whether  $g$  is a quadratic residue or not modulo  $N$ . If the simulator chooses the sign according to the flip of an unbiased coin, the resulting distribution is indistinguishable from the true distribution assuming the hardness of quadratic residuosity modulo a Blum integer.  $\square$

We note that step (2) of the protocol is crucial. Without it the condition of step (3) might fail (and reveal the factorization) even when  $p$  and  $q$  are prime. We also note that in practice the probability that a non RSA modulus passes even one iteration of this test is actually much less than a half.

### 3.1 A complete probabilistic primality test

Integers  $N$  that fall into Case 4 of Lemma 2 can be filtered out by adding an extra fourth step to our protocol. Due to lack of space we only give a high level description. There are two alternatives:

1. Let  $K$  be the group  $K = (\mathbb{Z}_N[x]/(x^2 + 1))^* / \mathbb{Z}_N^*$ . When  $N = pq$  is a product of two distinct primes  $K$  contains  $(p + 1)(q + 1)$  elements (recall  $p = q = 3 \pmod{4}$ ). In this case all  $g \in K$  satisfy  $g^{(p+1)(q+1)} = 1$ . One can show that when  $N$  falls into Case 4, at least half the elements in  $K$  do not satisfy the above condition. Hence, by picking a random  $g \in K$  and jointly testing that  $g^{(p+1)(q+1)} = 1$  Alice and Bob can eliminate all such  $N$ .
2. Alternatively, observe that  $N$  that fall into Case 4 satisfy  $\gcd(N, p+q-1) > 1$ . The parties can easily test this with the help of a third party, Henry. Alice picks a random  $r_a$ . Bob picks a random  $r_b$ . Using the protocol of the next section they compute  $z = (r_a + r_b)(p_a + q_a + p_b + q_b - 1) \pmod{N}$  and test that  $\gcd(z, N) > 1$ . If so, then  $N$  is rejected. Unfortunately this test also eliminates a few valid RSA moduli, i.e. moduli  $N = pq$  with  $p, q$  prime and  $q = 1 \pmod{p}$ .

## 4 Distributed computation of $N$

We now turn our attention to the computation of  $N$ . We describe our protocols for the case of two parties with a helper and discuss the case of  $k > 2$  parties in Section 7.

In the case of two parties, Alice and Bob possess integers  $p_a, q_a$  and  $p_b, q_b$  respectively. They wish to compute the integer  $N = (p_a + p_b)(q_a + q_b)$  such that at the end of the computation Alice has no information about  $p_b, q_b$  beyond what is revealed by the knowledge of  $N$ . The same should hold for Bob. To make the protocol secure in the information theoretic sense we require the help of a third “helper” party called Henry. Henry has no information about either  $p_i$  nor  $q_i$  (for  $i = a, b$ ) and the same should hold at the end of the protocol. Clearly, Henry learns  $N$  (since  $N$  is public) but he learns nothing more.

BenOr, Goldwasser and Wigderson [3] (and similarly Chaum, Crépeau and Damgård [6]) describe an elegant protocol for private evaluation of general functions for three or more parties. Their full technique is an overkill for the simple function we have in mind. We adapt and optimize their protocol in several ways so as to minimize the amount of computation and communication between the parties. From here on, let  $P > N$  be some prime. Unless otherwise stated, all arithmetic operations are done modulo  $P$ . The protocol works as follows:

**Alice:** Alice picks two random lines that intersect the  $y$  axis at  $p_a, q_a$  respectively. This is done by picking two integers  $c_a, d_a \in \mathbb{Z}_P^*$  and using the lines  $c_a x + p_a$  and  $d_a x + q_a$ . She evaluates each line at three points  $x_a = 1, x_b = 2, x_h = 3$ . Let  $p_{a,i} = c_a x_i + p_a$  and  $q_{a,i} = d_a x_i + q_a$  for  $i = a, b, h$ . Next, Alice picks two random numbers  $p_{b,a}, q_{b,a}$  and a random quadratic polynomial  $r(x)$  such that  $r(0) = 0$ . Set  $r_i = r(x_i)$  for  $i = a, b, h$ . She computes  $N_a = (p_{a,a} + p_{b,a})(q_{a,a} + q_{b,a}) + r_a$ . Finally, she sends  $p_{a,b}, q_{a,b}$  and  $p_{b,a}, q_{b,a}$  and  $r_b$  to Bob. She sends  $p_{a,h}, q_{a,h}, r_h$  and  $N_a$  to Henry.

**Bob:** Bob computes  $c_b = (p_{b,a} - p_b)/x_a$  and  $d_b = (q_{b,a} - q_b)/x_a$ . Note that the two lines  $c_b x + p_b$  and  $d_b x + q_b$  intersect the  $y$ -axis at  $p_b, q_b$  respectively and evaluate to  $p_{b,a}, q_{b,a}$  at  $x_a$ .

Next, Bob computes  $p_{b,i} = c_b x_i + p_b$  and  $q_{b,i} = d_b x_i + q_b$  for  $i = b, h$ . He computes  $N_b = (p_{a,b} + p_{b,b})(q_{a,b} + q_{b,b}) + r_b$  and sends  $p_{b,h}, q_{b,h}$  and  $N_b$  to Henry.

**Henry:** Henry computes  $N_h = (p_{a,h} + p_{b,h})(q_{a,h} + q_{b,h}) + r_h$ . He then interpolates the quadratic polynomial  $\alpha(x)$  that passes through the points  $(x_a, N_a)$ ;  $(x_b, N_b)$ ;  $(x_h, N_h)$ . We have  $\alpha(0) = N$ . Henry sends  $N$  to Alice and Bob.

To see that  $\alpha(0) = N$  observe that the polynomial  $\alpha(x)$  satisfies

$$\alpha(x) = ((c_a x + p_a) + (c_b x + p_b)) \cdot ((d_a x + q_a) + (d_b x + q_b)) + r(x)$$

Indeed  $\alpha(x_i) = N_i$  for  $i = a, b, h$ .

**Lemma 4.** *Given  $N$ , Alice, Bob and Henry can each simulate the transcript of the protocol. Consequently, they learn nothing more than the value of  $N$ .*

*Proof Sketch.* This is clear for Alice and Bob. To simulate Henry's view the simulator picks  $p_{a,h}, q_{a,h}, p_{b,h}, q_{b,h}, r_h$  at random and computes  $N_h = (p_{a,h} + p_{b,h})(q_{a,h} + q_{b,h}) + r_h$ . It then picks a random quadratic polynomial  $\alpha(x)$  satisfying  $\alpha(0) = N$  and  $\alpha(x_h) = N_h$ . It computes  $N_a = \alpha(x_a)$  and  $N_b = \alpha(x_b)$ . These values are a perfect simulation of Henry's view.  $\square$

The protocol's communication pattern is very simple: initially Alice sends one message to Bob and one to Henry. Then Bob sends a message to Henry. Finally, Henry publishes the value of  $N$ . Hence, during the protocol only three messages are sent. The protocol is also efficient in computation since only three multi-precision multiplications are performed.

The protocol differs from the BGW protocol in two ways. First, there is no need for a truncation step. Second, to minimize the number of messages we let Alice pick her shares  $p_{b,a}$  and  $q_{b,a}$  of Bob's secret. Bob then picks his polynomial to be consistent with Alice's choice.

## 5 Trial division

In this section, we consider the trial division step. We describe our protocol for the case of two parties with a helper and discuss the case of  $k > 2$  parties in Section 7.

Let  $q$  be some random number. The first step in testing the primality of  $q$  is trial division, which tests if  $q$  is divisible by any small prime. In our case  $q = q_a + q_b$  where Alice knows  $q_a$  and Bob knows  $q_b$ . Let  $p_1, \dots, p_j$  be the set of small primes to be considered. Together they wish to test that  $q \not\equiv 0 \pmod{p_i}$

for all  $i$ ,  $1 \leq i \leq j$ , without revealing any other information about  $q_a, q_b$ . This is equivalent to testing that  $q_a \bmod p_i \neq -q_b \bmod p_i$  for all  $i$ ,  $1 \leq i \leq j$ . A number of simple protocols have been proposed for privately evaluating the equality predicate [15], including one with a third helper party, based on universal classes of hash functions [5, 25] (attributed to Noga Alon in [15]). Using this equality test, the trial division protocol is as follows:

**Alice** Pick random  $c_i \in \mathbb{Z}_p$ , and  $d_i \in \mathbb{Z}_{p_i}^*$ . Compute  $u_i = c_i + d_i q_a \bmod p_i$ , for all  $i$ ,  $1 \leq i \leq j$ . Send  $c_1, d_1, \dots, c_j, d_j$  to Bob and  $u_1, \dots, u_j$  to Henry.

**Bob** Compute  $v_i = c_i - d_i q_b \bmod p_i$  for all  $i$ ,  $1 \leq i \leq j$ . Send  $v_1, \dots, v_j$  to Henry.

**Henry** Output "pass" if  $u_i \neq v_i$  for all  $i$ ,  $1 \leq i \leq j$ . Otherwise, output "fail".

**Lemma 5.** *The output of the protocol is "pass" if and only if  $q \neq 0 \bmod p_i$  for all  $i$ ,  $1 \leq i \leq j$ .*

**Lemma 6.** *When the output is "pass", each party can simulate its view of the transcript of the protocol. Consequently, when the output is "pass", the parties learn nothing about  $q$  other than the fact that  $q \neq 0 \bmod p_i$  for all  $i$ ,  $1 \leq i \leq j$ .*

## 6 Shared generation of public/private keys

In this section, we consider the step of key generation. We describe our protocol for the case of two parties and discuss the case of  $k > 2$  parties in Section 7.

Suppose Alice and Bob have successfully computed  $N = pq = (p_a + p_b)(q_a + q_b)$ . They wish to compute shares of  $d = e^{-1} \bmod \phi(N)$  for some agreed upon value of  $e$ . We have two approaches for computing shares of  $d$ . The first only works for small  $e$  (say  $e < 1000$ ) but is very efficient requiring very little communication between the parties. The second works for any  $e$  and is still efficient, however it requires the help of Henry and takes more rounds of communication (but still constant).

### 6.1 Small public exponent

We begin by describing an efficient technique for generating shares of  $d$  when the public exponent  $e$  is small. For simplicity throughout the section we assume  $e = 3$ .

First, Alice and Bob compute  $\phi(N) \bmod 3$ , by exchanging  $p_a + q_a \bmod 3$  and  $p_b + q_b \bmod 3$ . This reveals some little information (less than two bits) about  $\phi(N)$ ; this information is of no use since it can be easily guessed. Observe that<sup>4</sup>:

$$\begin{cases} d = [\phi(N) + 1]/3 = \frac{1}{3}[N + 2 - (p_a + p_b + q_a + q_b)] & \text{if } \phi(N) = 2 \bmod 3 \\ d = [2\phi(N) + 1]/3 = \frac{2}{3}[N - (p_a + p_b + q_a + q_b)] + 1 & \text{if } \phi(N) = 1 \bmod 3 \end{cases}$$

<sup>4</sup> The case  $\phi(N) = 0 \bmod 3$  is of no interest since in that case  $e = 3$  can not be used as a public RSA exponent.

Consequently, knowing  $\phi(N) \bmod 3$  enables Alice and Bob to locally compute shares of the decryption exponent  $d$ : If  $\phi(N) \bmod 3 = 1$ , then Alice sets her share to be  $d_a = \lfloor \frac{N-2p_a-2q_a}{3} \rfloor + 1$  and Bob sets his share to be  $d_b = \lceil \frac{N-2p_b-2q_b}{3} \rceil$ . If  $\phi(N) \bmod 3 = 2$ , then  $d_a = \lfloor \frac{N-p_a-q_a+2}{3} \rfloor$  and  $d_b = \lceil \frac{-p_b-q_b}{3} \rceil$ . Either way  $d = d_a + d_b \bmod \phi(N)$ . This enables threshold decryption as described in [13], i.e.,  $c^d \equiv c^{d_a} c^{d_b} \bmod N$ .

## 6.2 Arbitrary public exponent

Unlike the previous technique, our second method for generating shares of  $d$  works for arbitrary public exponent  $e$  and leaks no information. However, it requires the help of Henry.

Recall that the public modulus  $N = (p_a + p_b)(q_a + q_b)$  satisfies  $\phi(N) = (N - p_a - q_a + 1) - (p_b + q_b)$ . We set  $\phi_a = N - p_a - q_a + 1$  and  $\phi_b = -p_b - q_b$ . Then  $\phi(N) = \phi_a + \phi_b$  is a sharing of  $\phi(N)$  between Alice and Bob. The private exponent  $d$  is the inverse of  $e \bmod \phi_a + \phi_b$ . Unfortunately, traditional inversion algorithms, e.g. extended gcd, involve computations modulo  $\phi_a + \phi_b$ . When  $\phi = \phi_a + \phi_b$  is shared among two users we do not know how to efficiently perform these computations. We therefore develop an inversion algorithm for computing  $e^{-1} \bmod \phi$  that avoids any computation modulo  $\phi$ .

When only a single user is involved the inversion algorithm works as follows:

- (1) Compute  $\zeta = \phi^{-1} \bmod e$ .
- (2) Set  $T = -\zeta \cdot \phi + 1$ . Observe that  $T \equiv 0 \bmod e$ .
- (3) Set  $d = T/e$ . Then  $d = e^{-1} \bmod \phi$  since  $d \cdot e \equiv 1 \bmod \phi$ . Notice that the algorithm made no reductions modulo  $\phi$ . Our inversion algorithm made use of the fact that  $e^{-1} \bmod \phi$  can be immediately deduced from  $\phi^{-1} \bmod e$ .

We now show how the above inversion algorithm can be used to compute shares  $d_a + d_b = e^{-1} \bmod \phi_a + \phi_b$ . Clearly we may assume  $\gcd(\phi(N), e) = 1$ .

**Step 1.** Alice picks a random  $r_a \in \mathbb{Z}_e$ . Bob picks a random  $r_b \in \mathbb{Z}_e$ .

**Step 2.** Using the protocol of Section 4 compute  $\Psi = (r_a + r_b) \cdot (\phi_a + \phi_b) \bmod e$ .

Since  $e$  is odd ( $\gcd(\phi(N), e) = 1$ ) all the required Lagrange coefficients indeed exist. At this point  $\Psi$  is known to both Alice and Bob. If  $\Psi$  is not invertible modulo  $e$  the protocol is restarted at Step 1.

**Step 3.** Alice sets  $\zeta_a = r_a \Psi^{-1} \bmod e$ . Bob sets  $\zeta_b = r_b \Psi^{-1} \bmod e$ .

Observe  $\zeta_a + \zeta_b = (r_a + r_b) \Psi^{-1} = \phi^{-1} \bmod e$ .

**Step 4.** Next they fix an arbitrary odd integer  $P > 2N^2e$ , e.g.  $P = 2N^2e + 1$ .

They then regard the shares  $0 \leq \zeta_a, \zeta_b < e$  as elements of  $\mathbb{Z}_P$ . Using a modification of the BGW protocol of Section 4 they compute a sharing of

$$A + B = -(\zeta_a + \zeta_b)(\phi_a + \phi_b) + 1 \bmod P$$

such that Alice knows  $A$  and Bob knows  $B$ . Recall that in Section 4 Alice uses a random quadratic  $r(x)$  such that  $r(0) = 0$ . Instead, Alice will choose a truly random quadratic  $r(x)$ . Then the final result computed by Henry is offset from the desired result by an additive factor of  $r(0)$ , where only Alice knows  $r(0)$ . If Henry gives his final result to Bob, then Alice and Bob

have additive shares of the desired result. These shares could then be re-randomized if Alice adds, and Bob subtracts, an agreed-upon random value unknown to Henry.

**Step 5.** From here on we regard  $A$  and  $B$  as integers  $0 \leq A, B < P$ . Our objective is to ensure that over the integers

$$A + B = -(\zeta_a + \zeta_b)(\phi_a + \phi_b) + 1 \quad (1)$$

Observe that  $0 \leq A + B \bmod P < P/N$  (since  $\zeta_a + \zeta_b < 2e$  and  $\phi(N) < N$ ). It follows that  $A + B > P$  with probability more than  $1 - \frac{1}{N}$  (the only way that  $A + B < P$  is if both  $A$  and  $B$  are less than  $P/N$ ). Therefore, if Alice sets  $A \leftarrow A - P$  then equation (1) holds over the integers. In the very unlikely event (that occurs with probability  $1/N$ ) that the relation doesn't hold over the integers, the wrong sharing of the private key will be generated. This will be detected when the parties do a trial decryption.

**Step 6.** At this point  $e$  divides  $A + B$  since

$$A + B = (\zeta_a + \zeta_b)(\phi_a + \phi_b) + 1 = -(\phi_a + \phi_b)^{-1}(\phi_a + \phi_b) + 1 \equiv 0 \pmod{e}$$

Therefore  $d = (A + B)/e$  since  $de = A + B = k\phi(N) + 1 \equiv 1 \pmod{\phi(N)}$ . Consequently, Alice sets  $d_a = \lfloor A/e \rfloor$  and Bob sets  $d_b = \lceil B/e \rceil$ . Clearly  $d = d_a + d_b$ .

Notice that the value  $P$  we use in step 4 is quite large. As a result the shares  $d_a, d_b$  are of the order of  $N^2$ . In actual implementations there is no need for this to happen. The only reason  $P$  has to be this large is to ensure that step 5 succeeds with overwhelming probability. If one is willing to tolerate leakage of one bit in step 5 then the parties can use a much smaller  $P$ , e.g.  $P = 2Ne + 1$ . If the resulting  $A, B$  satisfy  $A + B > P$  then the correct sharing of  $d$  is obtained. Otherwise, trial decryption will fail and the parties learn that  $A + B < P$ . In this case, Alice adds  $P$  back to her share  $A$  and step 6 is repeated again. The correct sharing of  $d$  is now obtained. This results in shares  $d_a, d_b$  of order  $N$ .

The computation of  $\phi^{-1} \bmod e$  (steps 1-3 above) is based on a technique due to Beaver [2].

## 7 Generalizations to $k$ parties

Our results thus far show how two parties can generate an RSA modulus  $N = (p_a + p_b)(q_a + q_b)$  with the help of a third neutral party. In this section we discuss how these results generalize to the case of three or more parties. In this case, the  $k$  parties will be generating an RSA modulus  $N = (p_1 + \dots + p_k)(q_1 + \dots + q_k)$ , where each party  $i$  knows  $p_i, q_i$ . Afterwards, assuming that the parties follow the protocol as required, no coalition of  $\lceil k/2 \rceil - 1$  parties can factor  $N$ .

The primality test from Section 3 generalizes easily to  $k > 2$  parties. Assume that the secret values chosen by the parties satisfy  $p_1 = q_1 = 3 \bmod 4$  while for all other parties  $p_i = q_i = 0 \bmod 4$ . Then party 1 computes  $v_1 = g^{\frac{N - p_1 - q_1 + 1}{4}} \bmod$

$N$ . Party  $i$  computes  $v_i = g^{\frac{p_i + q_i}{4}}$ ,  $2 \leq i \leq k$ . They all publish their values and verify that  $v_1 \equiv \pm v_2 v_3 \dots v_k \pmod{N}$ . The arguments for correctness and privacy are essentially the same. The resulting protocol is  $k$ -private.

To generalize the distributed computation of  $N$  of Section 4 to  $k > 2$  parties use the BGW protocol with higher degree polynomials (rather than linear). The BGW protocol can be made private (i.e. no information about the  $p_i, q_i$  is leaked) even when  $\lceil k/2 \rceil - 1$  parties collude.

Trial division (Section 5) with  $k > 2$  parties can be done  $\lceil k/2 \rceil - 1$  privately, but a different protocol must be used. We adapt an idea due to Beaver [2]. Let  $q = q_1 + \dots + q_k$  be an integer shared among  $k$  parties. Let  $p$  be a small prime. To test if  $p$  divides  $q$  each party picks a random  $r_i \in \mathbb{Z}_p$ . Using the BGW protocol they compute  $qr = (\sum q_i)(\sum r_i) \pmod{p}$ . If  $qr \neq 0$  then  $p$  does not divide  $q$ . Furthermore, since  $r$  is unknown to any minority of parties,  $qr$  provides no other information about  $q$ . Note that if  $qr = 0 \pmod{p}$  it could still be the case that  $p$  does not divide  $q$ . However, if the test is repeated twice for each small prime  $p$ , the probability that a good candidate is rejected is at most  $1 - \prod_{p < B} (1 - \frac{1}{p^2}) < \frac{1}{2}$ .

The first key generation protocol of Section 6 immediately generalizes to produce a  $k$ -out-of- $k$  sharing of a private key  $d$  among  $k > 2$  parties. The second protocol can produce a  $k$ -out-of- $k$  sharing  $d = \sum d_i$ ; however the computation is based on BGW and is therefore only  $\lceil k/2 \rceil - 1$  private.

The more difficult case of  $t$ -out-of- $k$  sharing of a private key among  $k > 2$  parties is treated in the next subsection.

## 7.1 $t$ -out-of- $k$ sharing

Since the computation of  $N$  in Section 4 relies on the BGW protocol [3] we are a priori restricted to threshold  $t$  satisfying  $t < \lceil k/2 \rceil$ . A coalition of more than  $k/2$  parties can already factor  $N$ . We show how to achieve any threshold  $t < \lceil k/2 \rceil$ .

To achieve  $t$ -out-of- $k$  sharing of  $d$ , first share  $d$  using a  $k$ -out-of- $k$  scheme as described above, i.e. each party computes a share  $d_i$  such that  $d = \sum d_i \pmod{\phi(N)}$ . Then each party  $i$  shares its share  $d_i$  with all other parties using a  $t$ -out-of- $k$  scheme. We denote the share of  $d_i$  sent to party  $j$  by  $d_{i,j}$ . A coalition  $\mathcal{C}$  of  $t$  parties can do threshold decryption using its shares of  $d$  and its shares  $d_{i,j}$  for  $i \notin \mathcal{C}$ . Thus, we are left with the problem of showing how party  $i$  generates the  $d_{i,j}$  given  $d_i$ . Secret sharing modulo  $\phi(N)$  is not easy. An elegant solution was given in [8] where the authors show how a trusted dealer (who knows the factorization of  $N$ ) can generate shares  $d_{i,j}$  as required. We can show that when  $N = (\sum p_i)(\sum q_i)$  where party  $i$  only knows  $p_i, q_i$ , there is no need for a trusted dealer. That is, the parties can engage in a multi-party protocol to compute the same shares  $d_{i,j}$  that were generated by the dealer in [8]. Unfortunately, this requires multiple invocations of the BGW protocol described in Section 4.

Since we are mainly concerned with efficient solutions we describe an alternate approach which works well when the threshold  $t$  is small. When  $t$  is small

$t$ -out-of- $k$  sharing can be achieved through  $t$ -out-of- $t$  sharing. Naively this can be done by giving each of the  $\binom{k}{t}$  coalitions a  $t$ -out-of- $t$  sharing of the secret. Other techniques [1] can reduce  $t$ -out-of- $k$  sharing to  $t$ -out-of- $t$  far more efficiently<sup>5</sup>. However, it is essential for these reductions that the instances of  $t$ -out-of- $t$  sharing be independent. Because it is difficult to compute reduction modulo  $\phi(N)$  efficiently without revealing  $\phi(N)$ , ordinary techniques for generating new sharing instances cannot be used.

We propose the following procedure for party  $i$  to generate many independent  $t$ -out-of- $t$  sharings of  $d_i$ . To avoid unnecessary indices we refer to  $d_i$  as  $s$ . Pick  $t - 1$  random integers  $s_1, \dots, s_{t-1} \in_R [-B, B]$  for some large  $B$  and compute  $s_t = s - \sum_{j=1}^{t-1} s_j$  (where addition is over the integers). We show that  $s_1, \dots, s_t$  is a private  $t$ -out-of- $t$  sharing of  $s$  for suitable choice of  $B$ . Note that this sharing scheme is at least as secure as the scheme where every share is a random elements in  $[-B, B]$  and  $i$  publishes the difference between the secret and the sum of all the shares. When  $s \in [1, b]$  the following lemma establishes that this scheme is sufficiently private when  $B > tb^{2+\epsilon}$  for any fixed  $\epsilon > 0$ .

**Lemma 7.** *Let  $s \in [1 \dots b]$ , and let  $p_x = \text{prob}(s = x) = \frac{1}{b}$  for all  $x \in [1, b]$ . Let  $(s_1, \dots, s_t) \in_R [-B, B]^t$ , and  $\delta = \sum_{i=1}^t s_i - s$ . For any coalition  $C \subset [1, t]$ , let  $p_{x,C} = \text{prob}(s = x | \delta, \{s_i\}_{i \in C})$ . Then, for every coalition  $C$  and every  $\epsilon > 0$ , the distributions  $\{p_x\}_x$  and  $\{p_{x,C}\}_x$  are statistically indistinguishable when  $B > tb^{2+\epsilon}$ .*

Due to space limitation we give the proof in the full version of the paper.

## 8 Summary and open problems

We presented techniques that allow two or more parties to generate an RSA modulus  $N = pq$  such that all parties are convinced that  $N$  is indeed a product of two primes; however none of them can factor  $N$ . When only two parties are involved, interaction with a third helper party is needed to complete some steps of the protocol. Finally we show how the parties can generate shares of a private decryption exponent to allow threshold decryption.

Our protocols are practical, though there is some slowdown in comparison to single user generation of an RSA key. The main reason is that both primes  $p, q$  are generated at once. This increases the number of tries until a suitable  $N$  is found, as was discussed in Section 2. A possible approach for solving this

<sup>5</sup> For instance we show how to efficiently implement 2-out-of- $k$  sharing from 2-out-of-2 sharing. Let  $d$  be a secret and  $r = \lceil \log k \rceil$ . Let  $d = d_{1,0} + d_{1,1} = d_{2,0} + d_{2,1} = \dots = d_{r,0} + d_{r,1}$  be  $r$  independent 2-out-of-2 sharings of the secret  $d$ . For an  $i \in [0, k]$  let  $i = i_r i_{r-1} \dots i_0$  be the binary digits of  $i$ . Party  $i$ 's share of the secret  $d$  is the set  $\{d_{r,i_r}, d_{r-1,i_{r-1}}, \dots, d_{0,i_0}\}$ . Given two parties  $i = i_r i_{r-1} \dots i_0$  and  $j = j_r j_{r-1} \dots j_0$  there exists an  $s$  such that  $i_s \neq j_s$ . Then  $d = d_{s,i_s} + d_{s,j_s}$ , enabling the two parties to reconstruct the secret. Hence, we achieved 2-out-of- $k$  sharing using only  $\log k$  independent 2-out-of-2 sharings (as opposed to  $\binom{k}{2}$  required by the naive solution).

is to generate  $N$  as  $N = p_a p_b (q_a + q_b)$  where  $p_a, p_b$  are primes known to Alice, Bob respectively and  $q_a, q_b$  are random  $n$  bit integers. The number of probes until  $q_a + q_b$  is found to be prime is just as in single user generation of  $N$ . Unfortunately, this approach doesn't scale well. To support  $k$  parties,  $N$  must be a product of  $k + 1$  primes. Also, one has to design a protocol for testing that such an  $N$  is indeed a product of *three* primes.

In the two party case our protocols require the use of a third helper party. The helper party is needed for the private computation of  $N = (p_a + p_b)(q_a + q_b)$ . Therefore, it is of some interest to develop efficient two party protocols for this specific function which do not make use of a third party. General two party computation protocols (e.g. [26]) are too inefficient.

Our protocols generate an RSA modulus which is the product of two large random primes. It would be useful to be able to generate moduli of some special form. For example, a modulus which is a product of "safe primes" (i.e., where both  $\frac{p-1}{2}$  and  $\frac{q-1}{2}$  are prime) has been considered for security purposes [18] as well as for technical reasons related to threshold cryptography [10, 16].

Throughout the paper we use a model in which parties honestly follow the protocol. The case of active adversaries that cheat during the protocol is of great interest as well. Since the RSA function is verifiable (the parties can simply check that they correctly decrypt encrypted messages) active adversaries are limited in the amount of damage they can cause. However, it may still be possible that one party can cheat during the protocol and consequently be able to factor the resulting  $N$ . Our techniques can be made to withstand some number of active adversaries though more parties must participate in the protocol. We leave the details for the full version of the paper.

## Acknowledgments

We thank Yair Frankel and Don Beaver for several stimulating discussions on our results.

## References

1. N. Alon, Z. Galil and M. Yung, "Dynamic-resharing verifiable secret sharing," ESA 1995.
2. D. Beaver, "Security, fault tolerance, and communication complexity in distributed systems," Ph.D. thesis, Harvard University, May 1990.
3. M. Ben-Or, S. Goldwasser, A. Wigderson, "Completeness theorems for non-cryptographic fault tolerant distributed computation", STOC 1988, pp. 1-10.
4. J. Benaloh (Cohen), "Secret sharing homomorphisms: keeping shares of a secret secret," Crypto '86, 251-260.
5. J. Carter and M. Wegman, "Universal classes of hash functions", *J. Comput. Syst. Sci.* 18 (1979), 143-154.
6. D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," ACM STOC 1988, 11-19.

7. N. De Bruijn, "On the number of uncanceled elements in the sieve of Eratosthenes", *Proc. Neder. Akad. Wetensch*, vol. 53, 1950, pp. 803–812. Reviewed in *LeVeque Reviews in Number Theory*, Vol. 4, Section N-28, p. 221.
8. A. DeSantis, Y. Desmedt, Y. Frankel, M. Yung, "How to share a function securely", *STOC 1994*, pp. 522–533.
9. Y. Desmedt, "Threshold cryptography," *European Transactions on Telecommunications and Related Technologies*, Vol. 5, No. 4, July-August 1994, pp. 35–43.
10. Y. Desmedt and Y. Frankel, "Shared generation of authenticators and signatures", *Crypto '91*, 457–469.
11. U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology* 1 (1988), 77–94.
12. A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," *Crypto '86*, 186–194.
13. Y. Frankel, "A practical protocol for large group oriented networks", *Eurocrypt 89*, pp. 56–61.
14. M. Franklin and S. Haber, "Joint encryption and message-efficient secure computation," *Journal of Cryptology*, 9 (1996), 217–232.
15. R. Fagin, M. Naor, P. Winkler, "Comparing information without leaking it", *CACM*, Vol 39, No. 5, May 1996, pp. 77–85.
16. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust and efficient sharing of RSA functions", *Crypto 96*, pp. 157–172.
17. O. Goldreich, S. Micali, A. Wigderson, "How to play any mental game", *STOC 1987*, pp. 218–229.
18. J. Gordon, "Strong primes are easy to find", *Eurocrypt 84*, pp. 216–223.
19. L. Guillou and J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," *Eurocrypt '88*, 123–128.
20. K. Ohta and T. Okamoto, "A modification of the Fiat-Shamir scheme," *Crypto '88*, 232–243.
21. H. Ong and C. Schnorr, "Fast signature generation with a Fiat Shamir-like scheme," *Eurocrypt '90*, 432–440.
22. T. Pederson, "A threshold cryptosystem without a trusted party," *Proceedings of Eurocrypt 91*, pp. 522–526.
23. M. Rabin, "Probabilistic algorithm for testing primality", *J. of Number Theory*, vol. 12, pp. 128–138, 1980.
24. R. Solovay, V. Strassen, "A fast monte carlo test for primality", *SIAM journal of computing*, vol. 6, pp. 84–85, 1977.
25. M. Wegman and J. Carter, "New hash functions and their use in authentication and set equality", *J. Comput. Syst. Sci.* 22 (1981), 265–279.
26. A. Yao, "How to generate and exchange secrets", *FOCS 1986*, pp. 162–167.