Efficient Dealer-Less Threshold Sharing of Standard RSA

Maged Hamada Ibrahim
Department of Electronics, Communications and Computers,
Faculty of Engineering, Helwan University, 1, Sherif St., Helwan, Cairo, Egypt (Email: mhii72@hotmail.com)

(Received Jan. 11, 2007; revised Oct. 23, 2007; and accepted Dec. 30, 2007)

Abstract

In [15] an efficient two-party, two-prime RSA function sharing protocol was proposed. The protocol proves efficiency over previously proposed protocols. When the sharing of standard RSA is considered, the protocol is faster than ever. In this paper, under the assumption that the adversary has eavesdropping and halting capabilities, we propose an efficient extension to the protocol of [15]. Our protocol enjoys the following properties (some of which are inherit from [15]): The protocol is fully distributed (i.e. does not require an honest dealer). It is a t-private and t-resilient (t, n)-threshold structure against a stationary adversary and also t-proactive (t, n)-threshold structure against a mobile adversary, where the number of players $n > 3t$. The players jointly generate two-prime RSA modulus in a number of trials of $\mathcal{O}(\ell/\lg\ell)$ since, the protocol avoids the inefficient distributed biprimality test. An extension of the protocol allows the generation of a RSA modulus which is a composite of two safe primes. Distributed primality tests are performed over a public modulus not a shared secret one, which reduces complexity on a per trial basis. We must emphasize that robustness against malicious adversaries (adversaries that masquerade the corrupted player by altering, deleting, sending wrong values, etc.) are beyond the scope of this paper.

Keywords: Digital signatures, distributed trust, GCD algorithm, primality test, quadratic slowdown, secure distributed computations, standard RSA, threshold RSA

1 Introduction

Distributed threshold computations (threshold cryptology) performed by a set of n players allow a fraction of these players, that exceeds a specified threshold, $1 \leq t < n$, to carry out the computation of a function via some secret parameters shared among them. Yet, no fraction of the players that does not exceed the threshold value can perform the computations correctly. Such threshold structures have many desirable security properties. For example, we are not forced to give complete trust to a single player that can misuse the output of the computation (e.g. digital signature), instead, the trust is distributed among a number of players. Another benefit is that the computation of the function will be performed correctly even if at most t players are disconnected (it is assumed that an adversary that is able to disconnect a player is also able to eavesdrop him). In other words, threshold structures deplete the capabilities of the adversary. An adversary that eavesdrops and halts at most t of the n players cannot perform the computations correctly and cannot prevent the correct computation of the function by the rest of the alive players. With the help of some proofs of knowledge techniques, threshold structures are also able to tolerate malicious behavior of a minority of the players. However, in this paper, we will not consider malicious adversaries. We only consider an adversary that has eavesdropping and halting capabilities.

Along the lifetime of the shared secret parameters, the adversary has plenty of time to attack as many players as she can. Consequently, the assumption that the adversary will not exceed a specific threshold t along the lifetime of the secret (stationary (non-mobile) adversary) is impractical, and threshold structures are becoming useless. In this case, proactive security comes to play. In proactive security, the lifetime of the secret is divided into time periods such that the adversary cannot successfully attack more than t players within any time period. A structure is t-proactive if it remains secure even if up to t players are successfully attacked within every time period. The idea is that at the beginning of each time period, the players join to randomize the shares of the secret without changing the value of this secret and without disclosing its value. As a result, the information gained by an adversary within any time period becomes obsolete in the next time period.

Threshold signature schemes represent the most important application of threshold cryptology. Secure threshold schemes for discrete log based systems were given: El-Gamal [3, 10], and DSS [7]. The results were considered theoretically satisfactory. However, solutions for the RSA function are considered more challenging due to the number theoretic constraints associated with the generation of this function. The main reason is that, in El-Gamal and DSS algorithms, all computations are performed over prime fields with publicly known primes. On the other hand, RSA requires secret primes in order to securely generate the function, also, the computations are performed over a secret non-prime modulus.

The nature of the RSA modulus N as a composite of two secret primes increased the difficulties to share the generation of the RSA function without the help of a trusted dealer. Due to the way the modulus is generated –as a product of two ℓ-bit random numbers chosen simultaneously– the probability that such generated modulus is a product of exactly two primes is $(\ln 2.\ell)^{-2}$, according to the prime number theorem, requiring a number of trials in the order of $O(\ell^2)$. Since ℓ ranges from 512 to 1024 bits depending on the security level and policy, the running time to reach a suitable modulus is several days using average processing speed which is quite a burden. Another, less challenging problem is that the RSA Euler totient $\varphi(N)$ does not represent a field which increased the difficulties to share the private exponent using efficient threshold secret sharing schemes which require a prime field to perform (e.g. Shamir's secret sharing scheme). The later problem is solved by employing an extension to Shamir's secret sharing scheme to perform over the integers (i.e. modulo nothing). Yet, still the nature of the RSA modulus N represents the major obstacle facing the efficient sharing of the RSA function without the help of an honest dealer. The honest dealer may setup the RSA function for the players and distribute shares of the private exponent. However, the honest dealer assumption threatens the main merit of distributed threshold computations, since, we are still forced to completely trust a single entity which represents a single point of failure and which may keep a copy of the secret exponent and go sign messages on behalf of the players.

Paper organization. This paper is organized as follows: Section 2 represents the previous work in the field. The motivations and contributions are stated in Section 3. The model assumptions are given in Section 4. Section 5 describes the outlines of our protocol. The tools used in building our protocol is given in Section 6. The protocol for a stationary adversary is described in Section 7. In section 8 we describe the protocol for safe primes. The protocol for mobile adversaries is discussed in Section 9. In Section 10, we compare an evaluate our protocol. Finally, the conclusions are given in Section 11.

2 Previous Work

The work in [5] represents the first realistic and general solution to RSA function sharing, the protocol in [5] is a modification of the protocol in [2]. In [17] a new idea to share the RSA function was introduced, it uses multi-level sharing. The RSA secret exponent is shared additively among the n players and each of the n shares is re-shared using any (t, n) secret sharing scheme. The robustness of the protocol of [17] was introduced in [8]. However, the idea of using multi-level (backup) sharing is practically inefficient as discussed in [4], specially in the absence of the honest dealer. In all the above protocols, in the absence of an honest dealer, when full distribution of the RSA function is considered, due to the employment of the inefficient biprimality distributed test, the running time complexity – according to number theory – is $\mathcal{O}(\ell^2)$, requiring a long time (several days) to reach a suitable two ℓ-bit primes RSA modulus N. The time goes longer and longer when safe primes are considered. Such quadratic slowdown was always considered as a nightmare. The protocol in [6] is a dealer-less protocol for the threshold sharing of the RSA function, the protocol employs biprimality testing and hence, it is rather slow.

In [4], to escape the complexity of the joint generation of the RSA modulus, it is assumed that the RSA Euler totient is shared among the players using (t, n) secret sharing scheme and this sharing among the players is performed by an honest dealer in an isolated environment. Such assumption overcomes the difficulties associated with the nature of the RSA modulus, yet, the protocol is not fully distributed and the idea of isolating the players is not practical as well. However, [4] introduced an efficient algorithm (GCD algorithm) to compute inverses over the shared secret Euler totient in order to efficiently compute shares of the RSA private exponent. The GCD algorithm is very efficient and we will employ it in the final phase of our protocol.

Different from the above, in [1] the idea to perform distributed primality tests over a shared secret modulus was given in an attempt to reduce the quadratic slowdown in the shared generation of the RSA function. Yet, performing distributed primality tests over a shared secret modulus, not a public modulus, increases complexity on per trial basis.

For the seeking of speed, drifting from the standard RSA settings and employing multiprime RSA (which is theoretically acceptable) the protocol proposed in [13] is a one trial protocol that does not require any distributed primality tests to jointly generate the RSA modulus. Consequently, the protocol in [13] and its extension [12, 14] are faster than ever known. Yet, still the multiprime RSA represents a drift from standard. Also, the protocols [12, 14] have the restriction that the number of primes of the modulus N are governed by the threshold value t, namely, N is required to be a composite of $t + 2$ primes. Such requirement becomes impractical for large threshold values.

In [15], under standard RSA assumptions, a two-party protocol for the shared generation of the RSA function was given. The protocol is faster than previously proposed protocols in the sense that it completely avoids the inefficient distributed biprimality test. The idea is as follows: Alice picks a random ℓ-bit secret integer a_1 and a random ℓ-bit secret prime p_a, Bob picks a random ℓ-bit secret integer b_1 and a random ℓ-bit secret prime p_b, using private two-party computations they jointly compute $N_1 = (a_1 + b_1)p_a p_b$, with no information revealed about any of their secret parameters. Then, they perform distributed Fermat's test to test $a_1 + b_1$ for primality. The test is performed over a public modulus N_1, consequently, the computational complexity is farther improved (on per trial basis). Since p_a and p_b are already primes, the number of trials in this case is in the order of $\mathcal{O}(\ell)$. They repeat the computations to share another prime $a_2 + b_2$. Now, Alice holds a_1, a_2 and Bob holds b_1, b_2 such that $a_1 + b_1$ and $a_2 + b_2$ are both primes. The sharing of the two primes could be performed in parallel and consequently the number of trials is still $\mathcal{O}(\ell)$. If trivial division tests are performed on the picked random integers, the complexity will improve to $\mathcal{O}(\ell/\lg\ell)$. They proceed to compute the RSA modulus $N = (a_1 + b_1)(a_2 + b_2)$ with no information revealed about any of the secret parameters. Alice and Bob agree on a public exponent e and proceed to compute shares of the secret exponent d using a reduced version of the GCD algorithm from [4].

3 Motivations and Contributions

In this section we present our motivations behind the work introduced in this paper and our contributions.

3.1 Motivations

The recent two-party protocol in [15] efficiently eliminates quadratic slowdown in the two-prime RSA function sharing and allows two parties to jointly generate shares of the private exponent in a number of trials in the order of $\mathcal{O}(\ell/\lg\ell)$. Hence, considering standard RSA, the protocol of [15] is the fastest. Yet, the protocol is limited to the two-party case. Direct extension of the protocol using oblivious transfer of strings will result in a $(n-1, n)$ threshold structure requiring all the n players to be alive in order to jointly generate the signature. An adversary that successfully kills at least one player prevents the rest of the players from computing the signature and consequently losing one of the main merits of threshold structures. Also, such direct extension will be computationally very complex.

3.2 Contributions

The contributions of this paper is to propose an efficient (t, n) threshold RSA signature protocol under standard settings. The protocol is faster than previously proposed threshold RSA signature protocols (assuming standard settings). The protocol is considered an efficient extension to the two-party protocol proposed in [15]. Our protocol enjoys the following properties:

- It is fully distributed and does not require a trusted dealer.
- The players jointly generate two ℓ-bit primes RSA modulus in a number of trials of $\mathcal{O}(\ell/\lg\ell)$.
- It allows the generation of a RSA modulus which is a composite of two safe primes.
- It is an optimum t-private, t-resilient protocol against a stationary eavesdropping and halting adversary. It is t-proactive against a mobile eavesdropping and halting adversary.
- Distributed primality test is performed over a publicly known modulus.

4 The Model Assumptions

In this section we precisely state our assumptions of the adversary and the communication models.

4.1 The Adversary Model

We assume that the adversary can see and learn all information sent to or from the corrupted player without compromising the correct behavior of this player. The alive players follow the execution steps of the protocol word for word. *We must emphasize that, in this work, we do not consider malicious behavior of an adversary. The only active behavior assumed in this work is that the adversary can halt (disconnect) the player to prevent him from participating in the protocol.*

Under the assumption that the adversary has an eavesdropping and halting capabilities, we distinguish two types of this adversary according to her mobility. *A stationary one* which means that the players chosen by the adversary will not exceed the threshold t along the life-time of the secret. *A mobile one* which is able to jump among the players attacking as much players as she can, she has the whole life-time of the secret to do so.

We assume that the number of players $n > 3t$, $t \geq 1$. The protocol is t-private, an adversary that successfully eavesdrops no more than t players cannot factor N. We assume that the adversary has a halting capabilities, an adversary that is able to halt a player can also eavesdrop him. The protocol is t-resilient, i.e. an adversary that is able to disconnect at most t players cannot prevent the rest of the players from completing the computations correctly. The protocol is t-proactive, the life-time of the secret is divided into time periods, an adversary that successfully eavesdrops and halts no more than t players in every time period does not threaten the secrecy of the protocol.

4.2 The Communication Model

In the communication model, the n players are fully connected such that any player can communicate with any other player through a private and authenticated one-to-one channel. Also the players have access to a dedicated broadcast channel. By dedicated we mean that if a player broadcasts a message, this message will be received by all other players and will be recognized as coming from this particular player. Private and authenticated channels can be realized by standard cryptographic techniques to achieve privacy, commitments, non-repudiation and authentication.

5 The Protocol Outlines

For clarity, we prefer to describe the idea and outlines of our protocol assuming stationary adversary model and without considering safe primes at the moment. With a set $\mathcal{P}$ of n players that are fully connected and that have access to a dedicated broadcast channel, the protocol outlines are as follows: The players run `JRSS` over the integers to share a random ℓ-bit secret integer p_1 so that each player holds a share of p_1 over a polynomial of degree t. Let $\mathcal{P}^*$ be a subset of any available $t+1$ players. Each player in $\mathcal{P}^*$ picks a random ℓ-bit secret prime q_i, he shares his picked prime among the n players using (t,n)-Shamir's secret sharing over the integers. At this point, each of the n players holds a share of p_1 over a polynomial of degree t and a share of each selected prime over $t+1$ polynomials each of degree t. The set of $t+1$ primes q_i's are not part of the final RSA modulus N, they are here to preserve the privacy of p_1 and to help in simplifying the distributed primality tests. Using secure distributed computations (the completeness theorem) the players join to compute shares of $N_1 = p_1(\prod_{i=1}^{t+1} q_i)$ over a polynomial of degree t. During the sharing of N_1, the players perform successive polynomial degree reduction to avoid the increase in the degree of the polynomial over which they share N_1. Each player broadcasts his share of N_1 so that each player is able to securely compute N_1.

Once N_1 is computed, the players proceed to perform the distributed primality test. They assume for an instant that $\varphi(N_1) = (p_1 - 1)(\prod_{i=1}^{t+1}(q_i - 1))$. Of course, this assumption is not true unless p_1 is also a prime. In this test the players agree on an integer $g \in_R Z^*_{N_1}$ and using secure distributed computations they securely compute $G = g^{\varphi(N_1)} \bmod N_1$ with no information revealed about $\varphi(N_1)$. They check G for unity. The players repeat the above described computations until $G = 1$, in this case, p_1 is a prime. The expected number of trials is $\mathcal{O}(\ell)$. If trivial division test is performed by each player on his picked random integer during the sharing of p_1, the complexity is improved to $\mathcal{O}(\ell/\lg\ell)$.

Now, the players share a suitable prime p_1, they repeat the above computations to share another prime p_2. Notice that the computations for p_1 is completely independent of the computations for p_2 and hence, these computations maybe performed in parallel.

Each player now holds a share of a secret prime p_1 over a polynomial of degree t and a share of a secret prime p_2 over a polynomial of degree t. Next, using the completeness theorem, the players proceed to securely compute the RSA modulus $N = p_1 p_2$ with no information revealed about p_1 or p_2. Using the shares of p_1 and p_2, the players share the RSA Euler totient $\varphi(N) = (p_1 - 1)(p_2 - 1)$. They agree on the public RSA exponent e and employ the efficient GCD algorithm [4] to compute shares of the private exponent d over a polynomial of degree t. At the end, any $t+1$ players are able to perform the signature on a given message.

6 Existing Tools

In this section we review the basic tools used in our protocol. The reader must be familiar with these tools in order to smoothly follow the protocol.

6.1 Shamir's Secret Sharing over a Prime Field

Although we will not employ Shamir's secret sharing over a prime filed, we briefly review it here since it represents the original scheme. Let $s \in Z_p$ be a secret held by the dealer where Z_p is a prime field. In order to share this secret among n players, $\mathcal{P} = \{P_1, ..., P_n\}$, where $p > n > t \geq 1$ [18], the dealer defines a polynomial $f(x) = \sum_{i=0}^{t} a_i x^i \bmod p$, he sets $a_0 = s$ and each other coefficient $a_{j\neq 0} \in_R Z_p$. $\forall i = (1, ..., n)$, the dealer secretly delivers $f(i)$ to player P_i. In the reconstruction phase, each player P_i broadcasts $f(i)$, the players are able to compute s from any $t+1$ shares using Lagrange interpolation formula, $s = f(0) = \sum_{i\in\mathcal{B}} \lambda_i f(i) \bmod p$, where, λ_i is Lagrange coefficient of player i, $\mathcal{B} \subseteq \mathcal{P}$ and $|\mathcal{B}| = t+1$.

6.2 Shamir's Secret Sharing over the Integers

Let $s \in [0, ..., K]$ be a secret where K is an approximate upper bound on s. In order to share the secret s over the integers (i.e. not modulo anything) among $n > t$ players [4, 6], the dealer defines the polynomial $f(x) = \sum_{i=0}^{t} a_i x^i$. He sets $a_0 = Ls$ where $L = n!$ and each other coefficient $a_{j\neq 0} \in_R [-L^2K...L^2K]$. $\forall i = (1, ..., n)$, the dealer computes and secretly delivers $f(i)$ to player P_i. In the reconstruction phase, the players broadcast their shares, the secret s can be computed from any $t+1$ shares using Lagrange interpolation. The players must not forget to divide the interpolation result by L. The reason for incorporating L in the computations is to ensure that Lagrange coefficients are integer values and no fractions will arise, since we are working modulo nothing.

6.3 Joint Random-Secret Sharing (JRSS)

The purpose of this scheme is to allow a set of n players to jointly agree on a random secret value r with no information revealed to any of them about r, also no coalition of at most t players knows any information about r. We describe the scheme over a prime field Z_p. Each player P_i picks $t+1$ values $r_i = a_0^{(i)}, ..., a_t^{(i)} \in_R Z_p$, he constructs the polynomial $R_i(x) = \sum_{j=0}^{t} a_j^{(i)} x^j$. $\forall j = (1, ..., n)$, each player P_i computes and secretly delivers $R_i(j)$ to player

P_j. Each player P_i sums what he receives from the other players to compute $\sigma(i) = \sum_{j=1}^{n} R_j(i)$. Each $\sigma(i)$ is a point on a t-degree polynomial $\sigma(x)$, notice that the free coefficient of this polynomial is $r = \sum_{i=1}^{n} r_i$. A special case of this scheme is the *Joint zero-secret sharing* in which the players distribute shares of a zero secret, $r = 0$ among themselves. The only difference is that each player constructs a t-degree polynomial with its free coefficient equals zero.

6.4 Changing the Polynomial Degree

In this sub-section we show how a set $\mathcal{P}$ of n players switch from a (t, n) secret sharing structure to a (t', n) secret sharing structure with $t \neq t'$. This technique is very helpful when polynomial degree reduction is required. The idea is simple, let $A(x)$ be the t-degree polynomial over which the players share a secret a. Each player $P_i \in \mathcal{P}$ holds a share $A(i)$ of a. Let $\mathcal{B} \subset \mathcal{P}$ be a subset of $t + 1$ players, therefore, $a = \sum_{i \in \mathcal{B}} \lambda_i A(i)$ where λ_i is Lagrange coefficient. Each player $P_i \in \mathcal{B}$ simply shares the quantity $A'(i) = \lambda_i A(i)$ among the n players using a t'- degree polynomial, $p_i(x)$. Notice that, $a = \sum_{i \in \mathcal{B}} A'(i)$. Each player $P_i \in \mathcal{P}$ computes $\sum_{j \in \mathcal{B}} p_j(i)$ which is a share of the secret a over a polynomial of degree t'.

6.5 Secure Distributed Computations

Let $a, b \in Z_p$ be two secrets that are shared using the t-degree polynomials $A(x)$ and $B(x)$ respectively. To compute $a+b$, each player P_i holding $A(i)$ and $B(i)$ computes $C(i) = A(i)+B(i)$ which is a share (point) on the t-degree polynomial $C(x) = A(x)+B(x)$, notice that the free term of $C(x)$ is $a + b$. In order to compute ca where c is a public constant, each player P_i computes $C(i) = cA(i)$, now each $C(i)$ is a point on a t-degree polynomial $C(x)$ which has a free term ca.

In the multiplication scenario, the number of players $n > 2t$, notice that the free coefficient of the polynomial $M(x) = A(x)B(x)$ is ab. However, there are two problems to be considered: The first is that the degree of $M(x)$ is $2t$ requiring at least $2t + 1$ shares to interpolate the polynomial. The second is that $M(x)$ is not random. It is required to randomize the coefficients of $M(x)$ and then to reduce its degree back to t. The reduction step is very important when successive multiplications are to be performed in order to avoid a large increase in the degree of the resulting polynomial since in this case there may not be enough players (shares) to interpolate the polynomial. To randomize $M(x)$ each player P_i simply selects a random $2t$-degree polynomial $r_i(x)$ subject to the condition that its free coefficient is 0 and distributes shares of this polynomial among the players. Each player P_i sums what he has to compute $M(i) + \sum_{j=1}^{n} r_j(i)$ which represents a share of the $2t$-degree polynomial $M'(x) = M(x) + \sum_{i=1}^{n} r_i(x)$ satisfying $M'(0) = M(0) = ab$. The players finally redistribute their shares of $M'(x)$ in order to reduce the degree back to t [9].

6.6 The GCD Algorithm

The GCD algorithm [4] is an algorithm to compute inverses over a shared secret modulus. For simplicity, we describe the algorithm in an n-out-of-n structure. The algorithm can be easily modified to support a threshold structure. Assume that a multiple of the RSA Euler totient ϕ is shared additively among a set of players, that is, each player P_i holds a share α_i such that $\sum_i \alpha_i = \psi\phi$. each player P_i picks a randomizing integer r_i of order $O(N^3)$ and broadcasts $\gamma_i = \alpha_i + r_i e$. All the players are able to compute $\gamma = \sum_i \gamma_i = \sum_i (\alpha_i + r_i e) = \psi\phi + Re$ where $R = \sum_i r_i$. Assume that $\gcd(\gamma, e) = 1$, there exist a and b such that $a\gamma + be = 1$ and thus $d = aR + b = e^{-1} \bmod \phi$. Player P_1 sets $d_1 = ar_1 + b$ and each other player $P_{i \neq 1}$ sets $d_i = ar_i$. It is obvious that $d = \sum_i d_i$. A warning has been given in [4] not to use the same value of ψ for different values of e since this attempt reveals $\psi\phi$ via the Chinese remainder theorem.

7 Our Protocol: Stationary Adversary

Let $\mathcal{P}$=$\{P_1, ..., P_n\}$ be the set of n players. By the notation *stationary adversary*, we assume that the adversary eavesdrops and halts no more than t players. Since we will perform distributed multiplications side by side with polynomial degree reductions, we need $2t+1$ alive players. Consequently, it is required that the number of players $n \geq 3t + 1$. Notice that, if the adversary has no halting capabilities, the number of players required will drop to $n \geq 2t+1$. For clarity, we do not consider the safe primes RSA modulus in this section. The protocol to share the generation of the RSA function under stationary adversary assumptions is as follows:

7.1 Sharing two Random Primes

In this subsection we show how the players share and test two secret primes p_1 and p_2. Each prime is to be shared over a polynomial of degree t. The sharing and testing of p_1 is similar to the sharing and testing of p_2. The players share p_1 as follows:

1) **Sharing a random integer** p_1
The players run the JRSS" to share a random ℓ-bit integer p_1, each player $P_i \in \mathcal{P}$,
   - Picks a random ℓ'-bit integer r_i, where $\ell' \simeq \ell - \lg n$.
   - Defines $t + 1$ values, $a_0^{(i)}, ..., a_t^{(i)}$. He sets $a_0^{(i)} = Lr_i$, and sets each $a_{j \neq 0}^{(i)} \in_R [-L^2 2^{\ell'} ... L^2 2^{\ell'}]$ where $2^{\ell'}$ is an approximate upper bound on r_i.

- Constructs the t-degree polynomial, $f_i(x) = \sum_{j=0}^{t} a_j^{(i)} x^j$.
- $\forall j = (1, ..., n)$, computes $f_i(j)$ and secretely delivers $f_i(j)$ to player P_j.
- Collects n shares, $f_1(i), ..., f_n(i)$, from the other players.
- Computes his share of p_1 as $f_{p_1}(i) = \sum_{j=1}^{n} f_j(i)$.

As a result, each player $P_i \in \mathcal{P}$ holds a share $f_{p_1}(i)$, which represents a point on a t-degree polynomial $f_{p_1}(x)$, with $f_{p_1}(0) = Lp_1$.

2) **Sharing the temporary prime factors and their Euler totients**
Let $\mathcal{P}^* \subset \mathcal{P}$ be any available subset of $t+1$ players. For simplicity and wlog, let $\mathcal{P}^*=\{P_1, ..., P_{t+1}\}$. Each player $P_i \in \mathcal{P}^*$ picks a random ℓ-bit prime q_i, he shares q_i and its Euler totient, $\varphi(q_i) = q_i - 1$ as follows, each player $P_i \in \mathcal{P}^*$,

- Defines $t+1$ values, $a_0^{(i)}, ..., a_t^{(i)}$. He sets $a_0^{(i)} = Lq_i$, and sets each $a_{j\neq 0}^{(i)} \in_R [-L^2 2^\ell ... L^2 2^\ell]$.
- Defines $t+1$ values, $b_0^{(i)}, ..., b_t^{(i)}$. He sets $b_0^{(i)} = L(q_i - 1)$, and sets each $b_{j\neq 0}^{(i)} \in_R [-L^2 2^\ell ... L^2 2^\ell]$.
- Constructs the t-degree polynomials, $f_{q_i}(x) = \sum_{j=0}^{t} a_j^{(i)} x^j$ and $f_{q_i - 1}(x) = \sum_{j=0}^{t} b_j^{(i)} x^j$.
- $\forall j = (1, ..., n)$, computes and secretely delivers $f_{q_i}(j)$ and $f_{q_i-1}(j)$ to player P_j.

As a result, each player $P_i \in \mathcal{P}$ holds the shares, $f_{p_1}(i)$, $f_{q_1}(i), ..., f_{q_{t+1}}(i)$, $f_{q_1-1}(i), ..., f_{q_{t+1}-1}(i)$. Each share is a point on a t-degree polynomial.

3) **Computing the modulus N_1**
The players want to compute $N_1 = p_1(\prod_{i=1}^{t+1} q_i)$ with no information revealed about p_1. They employ the completeness theorem to perform successive multiplication with polynomial degree reduction. To share $p_1 q_1$ over a polynomial of degree t, each player $P_i \in \mathcal{P}$ computes $f_{p_1 q_1}(i) = f_{p_1}(i) f_{q_1}(i)$. The quantity, $f_{p_1 q_1}(i)$ represents a point on a $2t$-degree polynomial $f_{p_1 q_1}(x)$ with its free term, $f_{p_1 q_1}(0) = L^2 p_1 q_1$. Next, it is required to reduce the degree of $f_{p_1 q_1}(x)$ back to t before performing any farther multiplications. Let $\mathcal{B} \subset \mathcal{P}$ be a subset of $2t+1$ players, therefore, $f_{p_1 q_1}(0) = \sum_{i\in\mathcal{B}} \lambda_i f_{p_1 q_1}(i)$ where λ_i is Lagrange coefficient. Each player $P_i \in \mathcal{B}$ simply,

- Shares the quantity $f'_{p_1 q_1}(i) = \lambda_i f_{p_1 q_1}(i)$ among the n players using a t-degree polynomial, say $p_i(x)$. Notice that, $f_{p_1 q_1}(0) = \sum_{i\in\mathcal{B}} f'_{p_1 q_1}(i)$.
- Computes $F_{p_1 q_1}(i) = \sum_{j=1}^{n} p_j(i)$ which is a share of the secret $L^2 p_1 q_1$ over a polynomial of degree t.

The players repeat the multiplication and the polynomial degree reduction for the remaining values $q_2, ..., q_{t+1}$ in a pairwise fashion. At the end, each player $P_i \in \mathcal{P}$ holds a share $F_{N_1}(i)$ which is a point on a t-degree polynomial $F_{N_1}(x)$ with its free term, $F_{N_1}(0) = L^{t+2} p_1 \prod_{i=1}^{t+1} q_i$. The players are now ready to compute the modulus, N_1.

- Each player P_i broadcasts his share $F_{N_1}(i)$.
- Any player is able to compute N_1 using Lagrange interpolation. The players must not forget to divide the interpolation result by L^{t+2}.

Remark 1. To avoid the blowup in the size of the final shares as well as $F_{N_1}(0)$ due to the accumulation of the quantity L which may case technical problems for large t and n, it is possible to perform successive division by L without compromising the correctness of the protocol. As we keep the free term as well as the coefficients of any shared polynomial multiplied by L then one may write $f_{p_1}(x) = Lf^*_{p_1}(x)$, likewise, $f_{q_1}(x) = Lf^*_{q_1}(x)$ and consequently $f_{p_1}(x) f_{q_1}(x) = L^2 f^*_{p_1}(x) f^*_{q_1}(x)$ thus any share $f_{p_1}(i) f_{q_1}(i)$ could be written as $L^2 f^*_{p_1}(i) f^*_{q_1}(i)$. This allows each player to divide his share by L before performing the next polynomial multiplication by $f_{q_2}(x)$. Notice that $f_{q_2}(x)$ will contain a free term multiplied by L, therefore, the blowup will not exceed L^2 and we are still keeping the resulting Lagrange coefficients as integers. This technique can be employed along the whole protocol whenever successive multiplications and polynomial degree reductions are performed. We will continue the description of our protocol without performing such successive division by L assuming it is understood.

4) **Distributed primality test for p_1**
Each player $P_i \in \mathcal{P}$ holds the shares, $f_{p_1}(i)$, $f_{q_1-1}(i)$,..., $f_{q_{t+1}-1}(i)$, while N_1 is publicly known. They agree on a quantity $g \in_R Z^*_{N_1}$. Assuming for an instant that $\varphi(N_1) = (p_1 - 1)(\prod_{i=1}^{t+1}(q_i - 1))$. Of course, this assumption is not true unless p_1 is also a prime. The objective now is that the players jointly compute the quantity $g^{\varphi(N_1)} \bmod N_1$ and test this quantity for unity. Using multiplication side by side with polynomial degree reduction, the players compute shares of the quantity $\varphi(N_1) = A - B$ where $A = p_1 \prod_{i=1}^{t+1}(q_i - 1)$ and $B = \prod_{i=1}^{t+1}(q_i - 1)$. As a result, each player holds a share $F_{\varphi(N_1)}(i)$ which is a point on a t-degree polynomial $F_{\varphi(N_1)}(x)$ with $F_{\varphi(N_1)}(0) = L^{t+2}\varphi(N_1)$. Again, Let $\mathcal{B}$ be a subset of any $t+1$ alive players. This subset is not necessarily the same subset chosen before, however, for simplicity, wlog we assume that $\mathcal{B}=\{P_1, ..., P_{t+1}\}$. Let $\lambda_1, ..., \lambda_{t+1}$ be the Lagrange coefficients corresponding to the chosen $\mathcal{B}$ which are publicly known. Each player $P_i \in \mathcal{B}$,

- Computes and broadcasts $g^{F_{\varphi(N_1)}(i)} \bmod N_1$.

Each player in $\mathcal{P}$ locally computes,

$$G = g^{\sum_{i=1}^{t+1} \lambda_i F_{\varphi(N_1)}(i)} \bmod N_1 = g^{L^{t+2}\varphi(N_1)} \bmod N_1.$$

Distributed primality test for p_1 (alternative). In an attempt to reduce interaction among the players during the distributed primality test of p_1, this test may be performed in an alternative way, given that the subset of players or a fraction of them that originally picked the temporary prime factors are still alive. However, this attempt requires more recursive exponentiations and hence, loses speed. The alive players maybe determined by letting each player broadcasts "I'm alive" message. There must be at least one alive player in $\mathcal{P}^*$ since the adversary can halt no more than t players. The interaction among the players decreases as the number of alive players in $\mathcal{P}^*$ increases. Each player $P_i \in \mathcal{P}$ broadcasts $g^{f_{p_1}(i)} \bmod N_1$. Any player in $\mathcal{P}$ is able to compute $G_0 = g^{\sum_i \lambda_i f_{p_1}(i) - L} \bmod N_1$ which is $g^{L(p_1-1)} \bmod N_1$. Distributed primality test of p_1 can be performed using the following recursive technique:

- For $k = 1, ..., t+1$:
  - If player $P_k \in \mathcal{P}^*$ is alive, he computes and broadcasts (replaces G_{k-1} by) $G_k = G_{k-1}^{q_k - 1} \bmod N_1$. Else, the players set $G_k = G_{k-1}$ for a dead player P_k.

Obviously, if all the players in $\mathcal{P}^*$ are alive, it is easily verified that the final quantity, $G = G_{t+1} = g^{L\varphi(N_1)} \bmod N_1$.

Let $\mathcal{I} = \{i_1, ..., i_k\}$ be the set of indices of the dead players in $\mathcal{P}^*$, $k \leq t$. Define $\varphi' = \prod_{i_j \in \mathcal{I}} (q_{i_j} - 1)$. Each player in $\mathcal{P}$ already holds a share $f_{q_{i_j}-1}(i)$ of each $(q_{i_j} - 1)$ over a t-degree polynomial $f_{q_{i_j}-1}(x)$. Using multiplication, side by side with polynomial degree reduction (a maximum of $t-1$ multiplications and reductions) each player $P_i \in \mathcal{P}$ holds a share $\Phi(i)$ of φ' over a t-degree polynomial $\Phi'(x)$ with $\Phi(0) = L^k \varphi'$. Given a subset of any $t+1$ alive players $\mathcal{B}$ each player $P_i \in \mathcal{B}$ broadcasts $G_k^{\lambda_i \Phi(i)} \bmod N_1$. Finally, any player is able to compute $G = G_{t+1}^{\sum_i \lambda_i \Phi(i)} \bmod N_1$.

The players repeat steps 1,2,3 and 4 until the quantity $G = 1$. Since $q_1, ..., q_{t+1}$ are originally primes, the expected number of trials to reach a prime p_1 is $\mathcal{O}(\ell)$.

Sharing another prime p_2. Once the players establish shares of a suitable prime p_1 over a t-degree polynomial, they repeat the steps 1 through 4 in exactly the same way to establish shares of another suitable prime p_2. Notice that the sharing of p_1 is completely independent of the sharing of p_2, consequently, the computations may be performed in parallel, resulting in an overall complexity of $\mathcal{O}(\ell)$. If trivial division test is performed during the sharing of the random integer in step 1, the complexity is reduced to $\mathcal{O}(\ell/\lg\ell)$.

Lemma 1. *Under the assumption that, 1) Shamir's secret sharing over the integers is secure, 2) The completeness theorem for distributed computations holds, 3) The factorization of a composite of at least two ℓ-bit primes is hard and 4) solving the discrete log problem over a composite group (RSA assumption) is hard; an eavesdropping adversary that successfully eavesdrops no more than t players gains no information about the shared random primes p_1 or p_2.*

Proof. Considering the situation for p_1 (the situation for p_2 is similar), a wise adversary (capable of eavesdropping no more than t players) will try her attack on the set of players that picked the temporary prime factors, say, $\mathcal{P}^* - \{P_{t+1}\}$. In this case, the adversary learns $q_1, ..., q_t$, the shares $f_{p_1}(1), ..., f_{p_1}(t)$ of the integer p_1 and the shares $f_{q_{t+1}}(1), ..., f_{q_{t+1}}(t)$ of the prime q_{t+1}. Following assumption (1), the adversary gains no information about p_1 or q_{t+1} from the collected shares. The adversary knows the public modulus N_1 and is able to compute $p_1 q_{t+1} = N_1 / \prod_{i=1}^{t} q_i$. In case p_1 is a prime, following assumption (2) and (3), the adversary is unable to factorize $p_1 q_{t+1}$ and hence, unable to fully factorize N_1. During the primality test, assumption (4) emphasize the infeasibility of the adversary to deduce any information broadcasted in the exponent of g. Consequently, the secrecy of p_1 is preserved. Notice that the polynomials used to share $q_1, ...q_{t+1}$ are completely independent of the polynomials used to share $(q_1 - 1), ..., (q_{t+1} - 1)$, consequently (and following assumption (2)) the shares $f_{q_{t+1}-1}(1), ..., f_{q_{t+1}-1}(t)$ do not help the adversary to gain any information about the prime factor, q_{t+1}. □

7.2 Joint Computation of the RSA Modulus N

We reached the situation where each player $P_i \in \mathcal{P}$ holds a share $f_{p_1}(i)$ of a random prime p_1 over a t-degree polynomial $f_{p_1}(x)$ and a share $f_{p_2}(i)$ of a random prime p_2 over a t-degree polynomial $f_{p_2}(x)$. Except for the shares of p_1 and p_2, all other parameters are erased. The players compute the RSA modulus $N = p_1 p_2$ as follows: Each player $P_i \in \mathcal{P}$,

- Defines $2t+1$ values, $a_0^{(i)}, ..., a_{2t}^{(i)}$, each $a_{j \neq 0}^{(i)} \in_R [-L^2 2^{2\ell} ... L^2 2^{2\ell}]$ and $a_0^{(i)} = 0$.
- Constructs the $2t$-degree randomizing polynomial, $f_i(x) = \sum_{j=0}^{2t} a_j^{(i)} x^j$.
- Computes and secretely delivers $f_i(j)$ to player P_j, $\forall j = (1, ..., n)$.

Now, each player $P_i \in \mathcal{P}$ computes and broadcasts, $f_{p_1 p_2}(i) = f_{p_1}(i) f_{p_2}(i) + \sum_{j=1}^{n} f_j(i)$. The quantity, $f_{p_1 p_2}(i)$ represents a point on a $2t$-degree polynomial $f_{p_1 p_2}(x)$ with its free term, $f_{p_1 p_2}(0) = L^2 p_1 p_2$. Any player is able to compute $N = f_{p_1 p_2}(0)/L^2$.

Lemma 2. *Under assumptions 1,2 and 3 in Lemma 1, an adversary that eavesdrops no more than t players and knows N gains no information about p_1 or p_2.*

7.3 Sharing the RSA Private Exponent

The RSA totient, $\varphi(N) = (p_1-1)(p_2-1) = p_1p_2-p_1-p_2+1 = (N+1)-p_1-p_2$. Regarding $N+1$ as a leftover term, which is publicly known, the players already holds shares of p_1 and p_2 and consequently, $\varphi(N)$. One must not forget L, that is, interpolating $f_{p_1}(x)$ and $f_{p_2}(x)$ will result in Lp_1 and Lp_2 respectively. The players agree on a public exponent e. Next, they join to compute shares of the private exponent d. We recall the efficient GCD algorithm from [4] to compute inverses over a shared secret modulus. We carefully adapt the algorithm to match the situation on hand. The players proceed as follows:

1) **Shared computation of $\gamma = \psi\varphi(N) + Re$**
Each player $P_i \in \mathcal{P}$:

- Picks $\psi_i \in_R [0...N^2]$ and, $b_1^{(i)}, ..., b_t^{(i)} \in_R [-L^2N^3...L^2N^3]$.
- Picks $r_i \in_R [0...N^3]$ and, $c_1^{(i)}, ..., c_t^{(i)} \in_R [-L^2N^4...L^2N^4]$.
- Picks $v_1^{(i)}, ..., v_{2t}^{(i)} \in_R [-L^2N^5...L^2N^5]$.
- Constructs the polynomials $g_i(x) = L\psi_i + b_1^{(i)}x + ... + b_t^{(i)}x^t$, $h_i(x) = Lr_i + c_1^{(i)}x + ... + c_{t'}^{(i)}x^t$ and the randomizing polynomial $v_i(x) = 0 + v_1^{(i)}x + ... + v_{2t}^{(i)}x^{2t}$.
- $\forall j = (1, ..., n)$, sends $g_i(j), h_i(j)$ and $v_i(j)$ to player P_j.
- Computes $g(i) = \sum_{j=1}^{n} g_j(i)$, $h(i) = \sum_{j=1}^{n} h_j(i)$ and $v(i) = \sum_{j=1}^{n} v_j(i)$.

Each player $P_i \in \mathcal{P}$ holds a share $g(i)$ on a t-degree polynomial $g(x)$ with $g(0) = L\psi$, a share $h(i)$ on a t-degree polynomial $h(x)$ with $h(0) = LR$ and a share $v(i)$ on a $2t$-degree randomizing polynomial $v(x)$ with $v(0) = 0$.

- Each player $P_i \in \mathcal{P}$ computes and broadcasts $\Gamma(i) = [L(N+1) - f_{p_1}(i) - f_{p_2}(i)]g(i) + Leh(i) + v(i)$.

$\Gamma(i)$ represents a point on a $2t$-degree polynomial $\Gamma(x)$ with its free term $\Gamma(0) = L^2\gamma$. Any player can compute $\Gamma(0)$ by interpolating any $2t+1$ shares, $\Gamma(.)$ and finally computes $\gamma = \Gamma(0)/L^2$.

Remark 2. To avoid the growth in the size of the shares of the private exponent d, $\mathcal{O}(N^5)$, we have one of two options: (i) As notified in [4], such growth is due to the fact that $\lg N$ is used to define negligible anything less than $2^{-\lg N}$. One may choose a different security parameter k and define negligible anything smaller than 2^{-k} instead of $2^{-\lg N}$, then part of the growth in the size of the shares would be in multiplicative factors of 2^k rather than N, e.g. $k = 100$ instead of $\lg N = 1024$. In this case the growth is $\mathcal{O}(N^2 2^{3k})$ which is even much less than $\mathcal{O}(N^3)$. Notice that the essential size of ψ is in the order of $\mathcal{O}(\varphi(N)^2)$ not $\mathcal{O}(N^2)$. One may consider the growth is still in the order of $\mathcal{O}(N^3)$. Actually, the parameter k is used in [4] to allow their security definitions and analysis. (ii) Another suggestion is to agree on a public prime $p > N^2 2^k$ and to perform the shared computation of γ over the prime field Z_p. Notice that ψ is to be in the order of $\mathcal{O}(N2^k)$ and hence $\gamma < N^2 2^k$. Both options have their bros and cons. The first suffers from the growth in the size of the final shares of d and the technical difficulties associated with working over the integers while the second is expected to be slow. We will continue the description of the protocol assuming the first option is chosen.

2) **Computing shares of the secret exponent d**

- Using the Euclidean algorithm, find a, b such that $a\gamma + be = 1$. If such a, b do not exist (with very low probability), repeat from step 1.
- Each player $P_i \in \mathcal{P}$ computes his share of the secret key as $D(i) = ah(i) + b$.

Lemma 3. *The above GCD algorithm reveals no information about $\varphi(N)$ or the secret exponent d.*

Proof. The reader may refer to [4] for a complete proof of Lemma 3 □

8 Safe Primes RSA Modulus

It was always recommended by the inventors of the RSA algorithm and other experts in the field that the RSA modulus is to be a composite of two safe primes instead of ordinary primes. A safe prime is on the form $2p+1$ where p is also a prime spoken of as Sophie-Germain prime. Such setting has many theoretical and technical benefits. Theoretically, it allows several security proofs to take place. Technically, it facilitates the computation of the RSA function. For example, if $N = (2p_1+1)(2p_2+1)$ where p_1 and p_2 are also primes then $\varphi(N) = 4p_1p_2$. The selection and agreement on a public modulus e is more efficient since in order for the condition $e \in_R Z^*_{\varphi(N)}$ to be satisfied, it is enough to select e as an odd number less than p_1 and p_2 without any farther trials. Our protocol can be carefully extended to allow the sharing of safe primes. However the complexity will slightly increase accordingly.

Once the players complete steps 1,2,3 and 4, each player $P_i \in \mathcal{P}$ holds a share $f_{p_1}(i)$ of a prime p_1 over a t-degree polynomial $f_{p_1}(x)$. Assuming for an instant that $2p_1+1$ is also a prime. In this case, define N_1^* as $N_1^* = (2p_1+1)(\prod_{i=1}^{t+1} q_i)$ and consequently, $\varphi(N_1^*) =$

$2p_1 \prod_{i=1}^{t+1}(q_i - 1)$. In order to securely test $2p_1 + 1$ for primality, the players proceed as follows:

1) **Sharing the temporary prime factors and their Euler totients**
Let $\mathcal{P}^* \subset \mathcal{P}$ be any available subset of $t+1$ players. For simplicity and wlog, let $\mathcal{P}^*=\{P_1, ..., P_{t+1}\}$. Each player $P_i \in \mathcal{P}^*$ picks a random ℓ-bit prime q_i, he shares q_i and its Euler totient, $\varphi(q_i) = q_i - 1$. As a result, each player $P_i \in \mathcal{P}$ holds the shares, $f_{q_1}(i), ..., f_{q_{t+1}}(i)$, $f_{q_1-1}(i), ..., f_{q_{t+1}-1}(i)$. Each share is a point on a t-degree polynomial.

2) **Computing the modulus N_1^***
The players want to compute $N_1^* = (2p_1+1)(\prod_{i=1}^{t+1} q_i)$ with no information revealed about p_1 or any of the q_i's. They employ the completeness theorem to perform successive multiplication with polynomial degree reduction in a way very similar to the sharing of N_1. At the end, each player $P_i \in \mathcal{P}$ holds a share $F_{N_1^*}(i)$ which is a point on a t-degree polynomial $F_{N_1^*}(x)$ with its free term, $F_{N_1^*}(0) = L^{t+2}N_1^*$. The players are now ready to compute the modulus, N_1^*.

   - Each player P_i broadcasts his share $F_{N_1^*}(i)$.
   - Any player is able to compute N_1^* using Lagrange interpolation. The players must not forget to divide the interpolation result by L^{t+2}.

3) **Distributed primality test for $2p_1 + 1$**
Each player $P_i \in \mathcal{P}$ holds the shares, $f_{p_1}(i)$, $f_{q_1-1}(i), ..., f_{q_{t+1}-1}(i)$, while N_1^* is publicly known. They agree on a quantity $g \in_R Z_{N_1}^*$. Assuming for an instant that $\varphi(N_1^*) = 2p_1(\prod_{i=1}^{t+1}(q_i - 1))$. This assumption is not true unless $2p_1 + 1$ is also a prime. The objective now is that the players jointly compute the quantity $g^{\varphi(N_1^*)} \bmod N_1^*$ and test this quantity for unity. Using multiplication side by side with polynomial degree reduction, the players compute shares of the quantity $\varphi(N_1^*)$. As a result, each player holds a share $F_{\varphi(N_1^*)}(i)$ which is a point on a t-degree polynomial $F_{\varphi(N_1^*)}(x)$ with $F_{\varphi(N_1^*)}(0) = L^{t+2}\varphi(N_1^*)$. Again, Let $\mathcal{P}^*$ be a subset of any $t+1$ alive players. Wlog let $\mathcal{P}^*=\{P_1, ..., P_{t+1}\}$. Let $\lambda_1, ..., \lambda_{t+1}$ be the Lagrange coefficients corresponding to the chosen $\mathcal{P}^*$ which are publicly known. Each player $P_i \in \mathcal{P}^*$,

   - Computes and broadcasts $g^{F_{\varphi(N_1^*)}(i)} \bmod N_1^*$.

   Each player in $\mathcal{P}$ locally computes,
   $G = g^{\sum_{i=1}^{t+1} \lambda_i F_{\varphi(N_1^*)}(i)} \bmod N_1^* = g^{L^{t+2}\varphi(N_1^*)} \bmod N_1^*$.
   The players repeat for a fresh value p_1 and fresh primes q_i's until $G = 1$.

4) **Computing the RSA modulus N**
Once $p_1, p_2, 2p_1+1$ and $2p_2+1$ are found to be primes, the players proceed to compute the RSA modulus $N = 4p_1p_2 + 2p_1 + 2p_2 + 1$. Each player $P_i \in \mathcal{P}$

   - Defines $2t + 1$ values, $a_0^{(i)}, ..., a_{2t}^{(i)}$, each $a_{j\neq 0}^{(i)} \in_R [-L^2 2^{2\ell} ... L^2 2^{2\ell}]$ and $a_0^{(i)} = 0$.
   - Constructs the $2t$-degree randomizing polynomial, $f_i(x) = \sum_{j=0}^{2t} a_j^{(i)} x^j$.
   - Computes and secretely delivers $f_i(j)$ to player P_j, $\forall j = (1, ..., n)$.

   Now, each player $P_i \in \mathcal{P}$ computes and broadcasts, $f_{N-1}(i) = 4f_{p_1}(i)f_{p_2}(i) + 2Lf_{p_1}(i) + 2Lf_{p_2}(i) + \sum_{j=1}^{n} f_j(i)$. The quantity, $f_{N-1}(i)$ represents a point on a $2t$-degree polynomial $f_{N-1}(x)$ with its free term, $f_{N-1}(0) = L^2(N-1)$. Any player is able to compute $N = (f_{N-1}(0)/L^2) + 1$.

5) **Sharing the RSA Euler totient**
Due to the simple expression for $\varphi(N) = 4p_1p_2$, the players easily compute shares of $\varphi(N)$. Let $\mathcal{B} \subset \mathcal{P}$ and $|\mathcal{B}|=2t+1$. Each player $P_i \in \mathcal{B}$,

   - Computes $f_\varphi(i) = 4f_{p_1}(i)f_{p_2}(i)$.
   - Shares the quantity $f'_\varphi(i) = \lambda_i f_\varphi(i)$ among the n players using a t-degree polynomial, $p_i(x)$.
   - Computes $F_\varphi(i) = \sum_{j=1}^{n} p_j(i)$.

   Now each player $P_i \in \mathcal{P}$ holds a share $F_\varphi(i)$ which is a point on a t-degree polynomial $F_\varphi(x)$ with $F_\varphi(0) = 4L^2p_1p_2 = L^2\varphi(N)$.

The players proceed to compute shares of the secret exponent d using the GCD algorithm.

9 Proactive Security and Mobile Adversaries

Long-lived shared secrets are vulnerable to a mobile adversary, an adversary that is able to jump among the players attacking as much players as she can. She has the whole life-time of the secret to attack $t+1$ or more players to possess the secret key. Also, she can kill $n-t$ or more players to destroy the secret key, since in this case, at most t good players remain, which are not enough to compute the RSA function. In order to attain long-lived certified secret key, its shares held by the players must be periodically renewed so that the information gained by the adversary in a time period say τ is nonsense in time period $\tau+1$. The new shares must interpolate to the same secret key. A good survey on this issue can be found in [11].

9.1 Proactiveness Phases

What is actually required to protect the secrecy of shared information over the long run is to periodically refresh the shares without changing the secret information in such a way that any information learned by the adversary about individual shares becomes obsolete after the shares are renewed. Similarly to avoid the gradual destruction of the information by corruption of shares, it is necessary to periodically recover lost or corrupted shares without compromising the secrecy of the shares. In the proactive approach, the lifetime of the secret is divided into periods of time (e.g. a day, a week etc.). At the beginning of each time period the share holders engage in an interactive update protocol, after which they hold completely new shares of the same secret. If there exist some halted players, all good players join together to re-share their shares with the rebooted players.

A proactive signature scheme involves three main phases:

1) **The key distribution phase.** In this phase, the players join to compute shares of the secret key over a polynomial of degree t. We use in this setup phase our protocol assuming the adversary is stationary.

2) **Joint signature generation phase.** In this phase each player publishes his partial signature on the message piece using his share of the secret key. Any $t+1$ or more pieces combine to generate the final signature. Yet, no player by himself or any coalition of players not exceeding the threshold value can forge the signature.

3) **Shares refreshment phase.** Where the players are able to periodically renew their shares of the secret key. The information about the shares of the secret key that an adversary is able to collect in one time period becomes obsolete in the next time period after renewing the shares. This is the phase we consider in this section.

9.2 Shares Refreshment

Once the shares of the secret key d are established (this is identical to what we have already described in the stationary adversary model) each player $P_i \in \mathcal{P}$ holds a share $D^{(0)}(i)$ which is a point on a polynomial of degree t, $D^{(0)}(x)$ with $D^{(0)}(0) = d$. The superscript indicates that we are in time interval number 0 (i.e. the initial shares). Consider any intermediate time period (τ). Let $\mathcal{P}' \subseteq \mathcal{P}$ represents the set of all players with alive shares at the end of time interval τ (the beginning of time interval $\tau + 1$). Each player $P_i \in \mathcal{P}'$ holds a share $D^{(\tau)}(i)$. Once time interval (τ) has elapsed, at the beginning of time interval $(\tau+1)$, the players initialize a shares renewal phase. All halted players reboot their systems. We assume that the player loses his share after rebooting. There must be at least $t + 1$ players with correct shares. The players in $\mathcal{P}'$ must help the rebooted players to recover their lost shares by redistributing valid shares of the secret key d among them. Assume wlog that $|\mathcal{P}'| = t + 1$. Each player $P_i \in \mathcal{P}'$,

- Defines $t + 1$ values, $a_0^{(i)}, ..., a_t^{(i)}$, each $a_{j\neq 0}^{(i)} \in_R [-L^2N^3...L^2N^3]$ and $a_0^{(i)} = \lambda_i D^{(\tau)}(i)$.
- Constructs the t-degree polynomial, $f_i(x) = \sum_{j=0}^{t} a_j^{(i)} x^j$.
- Computes and secretely delivers, $f_i(j)$ to player P_j, $\forall j = (1, ..., n)$.

Now each player $P_i \in \mathcal{P}$ is able to compute his new/recovered share of the secret key as $D^{(\tau+1)}(i) = \sum_j f_j(i)$. Notice that, the described shares recovery protocol automatically refreshes the shares held by all the players. Another method to refresh the shares (this method could be performed if all the players are already alive i.e. none of the players is halted and no recovery is needed) is as follows: Each player simply run a joint zero-secret sharing, that is, each $P_i \in \mathcal{P}$,

- Defines $t + 1$ values, $a_0^{(i)}, ..., a_t^{(i)}$, each $a_{j\neq 0}^{(i)} \in_R [-L^2N^3...L^2N^3]$ and $a_0^{(i)} = 0$.
- Constructs the t-degree randomizing polynomial, $f_i(x) = \sum_{j=0}^{t} a_j^{(i)} x^j$.
- Computes and secretely delivers $f_i(j)$ to player P_j, $\forall j = (1, ..., n)$.
- Computes his new share as, $D^{(\tau+1)}(i) = D^{(\tau)}(i) + \sum_{j=1}^{n} f_j(i)$.

Notice that the shares $D^{(\tau)}(.)$ are completely inconsistent with the new shares $D^{(\tau+1)}(.)$. However, any $t+1$ shares, $D^{(\tau+1)}(.)$ will interpolate to the same secret d. consequently, the shares collected by the adversary (no more than t shares) in time period τ become obsolete in time period $\tau + 1$.

9.3 Updating Personal Public-Keys

We must mention that each player $P_i \in \mathcal{P}$ has his own private and public key pair which is used to attain all private and authenticated information transfers in the RSA function sharing protocol. These keys must be renewed as well or else, the discussed proactiveness will fail to preserve the privacy of the secret key, since, an adversary controlling a player in time period (τ) but not in time period $(\tau + 1)$ can know all private information sent by this player in time period $(\tau + 1)$. Hence, before initializing a share renewal phase, the first thing to be done is that every player must update his own private and public key pair. Also, any player after reboot must negotiate for a new private/public key pair including registration and certification of the new pair.

10 Comparison and Evaluation

In this section we compare our protocol to the protocol of [15]. In secure multiparty computations (SMPC), the two-party computations have been always considered as a special case that require special treatment due to several reasons. (i) In two-party computations there are no fraction of the parties that may represent a majority, consequently, both parties must be alive in order to share the computations. (ii) Threshold structures tools used in this paper fail to fit the two-party case whenever secure distributed computations (e.g. shared multiplication of two secrets) are performed since in this case at least three parties are required ($n > 2t, t \geq 1$) in the absence of a halting adversary and at least four parties ($n > 3t, t \geq 1$) for an eavesdropping and a halting adversary. (iii) In case of a malicious adversary (corruption of players), in the two-party case, it is theoretically impossible to correct errors (to filter out bad players), only error detection is possible, after which, the protocol aborts, whereas, when there are plenty of players, minority corrupted players may be detected and filtered out from disturbing the correct behavior. Therefore, the comparison to the protocol of [15] in the strict sense would be unfair. However, to overcome the above mentioned problems in the two-party structure, the protocol in [15] relies heavily on the oblivious transfer protocols that require extensive amount of modulo exponentiations which are computationally expensive. In our protocol, no exponentiations are performed except when performing the distributed primality test. This test is also performed in [15]. Yet, the protocol proposed in this paper as a threshold structure fails to fit the two-party situation.

11 Conclusions

In this paper, we introduced a fully distributed proactive and threshold RSA function sharing protocol under standard RSA assumptions. The protocol represents an efficient extension to the two-party protocol of [15] and proves efficiency over previously proposed threshold-structure protocols. Our protocol avoids many of the existing inefficient techniques such as backup sharing, distributed biprimality test and distributed primality test over a shared secret modulus. It is an optimum t-private, t-resilient protocol against a stationary eavesdropping and halting adversary. It is t-proactive against a mobile eavesdropping and halting adversary. Although we proposed our protocol assuming an eavesdropping and halting adversary, the protocol can be extended to face a malicious adversary using verifiable secret sharing and techniques from [14, 16]. However, an efficient extension is not expected to be straightforward.

Acknowledgments

The author would like to thank the anonymous reviewers of the IJNS for their valuable comments that reflect their sharp reading of this paper.

References

[1] J. Algesheimer, J. Camenisch, and V. Shoup, "Efficient computation modulo a shared secret with application to the generation of shared safe-prime products," *Crypto '02*, pp. 417-432, 2002.

[2] D. Boneh, and M. Franklin, "Efficient generation of shared RSA keys," *Crypto '97*, pp. 425-439, 1997.

[3] M. Cerecedo, T. Matsumoto, and H. Imai, "Efficient and secure multiparty generation of digital signatures based on discrete logarithms," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E76–A, no. 4, pp. 532-545, 1993.

[4] D. Catalano, R. Gennaro, and S. Halevi, "Computing inverses over a shared secretmodulus," *Eurocrypt '00*, pp. 190-206, 2000.

[5] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung, "Optimal-resilience proactive public-key cryptosystems," *38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pp. 384-393, 1997.

[6] Y. Frankel, P. D. MacKenzie, and M. Yung, "Robust efficient distributed RSA-key generation," *Proceedings of STOC '98*, pp. 663-672, 1998.

[7] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," *Eurocrypt '96*, pp. 354-371, 1996.

[8] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust and efficient sharing of RSA function," *Crypto '96*, pp. 157-172, 1996.

[9] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," *Proceedings of the 19th STOC*, pp. 218-229, 1987.

[10] L. Harn, "Group-oriented (t, n) threshold digital dignature scheme and multisignature," *IEE Proceedings - Computers and Digital Techniques*, vol. 141, no. 5, pp. 307-313, 1994.

[11] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," *Crypto '95*, pp. 339-352, Springer Verlag, 1995.

[12] M. H. Ibrahim, I. A. Ali, I. I. Ibrahim, and A. H. E. Sawy, "Fast fully distributed and threshold RSA function sharing," *Proceedings of the Information Systems: New Generations (ISNG-2004), special Session on Smart Cards*, pp. 11-16, Las Vegas, USA, 2004.

[13] M. H. Ibrahim, I. I. Ibrahim, and A. H. E. Sawy, "Fast three-party shared generation of RSA keys without distributed primality tests," *Proceedings of the Informatin Systems: New Generations Conference (ISNG 2004), special Session on Smart Cards*, pp. 5-10, Las Vegas, USA, 2004.

[14] M. H. Ibrahim, "Verifiable threshold sharing of a large secret safe prime," *Proceedings of the International Conference on Information Technology Coding and Computing, ITCC 2005*, pp. 608-613, USA, 2005.

[15] M. H. Ibrahim, "Eliminating quadratic slowdown in two-prime RSA function sharing," *International Journal of Network Security (IJNS)*, vol. 7, no. 1, pp. 107-114, 2008.

[16] T. V. Le, K. Q. Nguyen, and V. Varadharajan, "How to prove that a committed number is prime," *Asiacrypt '99*, pp. 208-218, Springer-Verlag, 1999.

[17] T. Rabin, "A Simplified approach to threshold and proactive RSA," *Crypto '98*, LNCS 1462, pp. 89-104, Springer-Verlag, 1998.

[18] A. Shamir, "How to share a secret," *Communication of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

Maged Hamada Ibrahim Received his BSc in communications and computers engineering from Helwan University, Cairo; Egypt. Received his MSc and PhD in Cryptography and Network security systems from Helwan University in 2001 and 2005 respectively. Currently, working as a lecturer, post doctor researcher and also joining several network security projects in Egypt. His main interest is Cryptography and network security. More specifically, working on the design of efficient and secure cryptographic algorithms, in particular, secure distributed multiparty computations. Other things that interest him are number theory and the investigation of mathematics for designing secure and efficient cryptographic schemes.