Proactive RSA

Yair Frankel[1] * Peter Gemmell[2] * Philip D. MacKenzie[3] * Moti Yung[4]

[1] CertCo, N.Y. NY, frankely@certco.com; on leave from Sandia National Labs
[2] Sandia National Labs, Albuquerque NM, psgemme@sandia.gov
[3] Boise State University, Boise ID, philmac@cs.idbsu.edu
[4] CertCo, N.Y. NY, moti@certco.com, moti@cs.columbia.edu.

Abstract. Distributed threshold protocols that incorporate proactive maintenance can tolerate a very strong "mobile adversary." This adversary may corrupt all participants throughout the lifetime of the system in a non-monotonic fashion (i.e., recoveries are possible) but the adversary is limited to the number of participants it can corrupt during any short time period. The proactive maintenance assures increased security and availability of the cryptographic primitive. We present a proactive RSA system in which a threshold of servers applies the RSA signature (or decryption) function in a distributed manner. Our protocol enables servers which hold the RSA key distributively to dynamically and co-operatively self-update; it is secure even when a linear number of the servers are corrupted during any time period; it efficiently maintains the security of the function; and it enables continuous function availability (correct efficient function application using the shared key is possible at any time). A major technical difficulty in "proactivizing" RSA was the fact that the servers have to update the "distributed representation" of an RSA key, while not learning the order of the group from which keys are drawn (in order not to compromise the RSA security). We give a distributed threshold RSA method which permits "proactivization".

1 Introduction

This work concerns algorithmic mechanisms to provide an increased level of security and availability to an RSA public-key system via distribution of the private key and active communication between shareholders. This improved level of security and availability counters a very strong "mobile adversary" who may corrupt **all** participants (servers, each with private memory) throughout the lifetime of the system but is not able to corrupt too many participants during any short period of time. The servers engage in a "proactive maintenance" of key shares that protects them against this mobile adversary who tries to learn the secret or disrupt their operation. *Proactive security* refers to security and availability

* This work was performed under U.S. Department of Energy contract number DE-AC04-94AL85000.

in the presence of a this mobile adversary which was first suggested by Ostrovsky and Yung [OY91]. Proactive security is vital for dealing with the increasing number of threats (including viruses and hackers) to local and international network domains, and for securing long-lived cryptographic keys that cannot be replaced easily (e.g., basic cryptographic infrastructure functions). In addition to protection, proactive maintenance techniques provide flexible and dynamic key management. As companies change (through mergers, firing of executives, etc.) and governments change (through elections, appointments, etc.), trust relations, and thus shareholders, must change. Proactive maintenance techniques allow for easy enrollment and disenrollment of shareholders.

A number of very useful cryptographic mechanisms have been efficiently "proactivized", such as pseudorandomness and secret sharing [OY91, CH94, HJKY95]. More recently, [HJJKY96] developed proactive public-key schemes for keys with publicly known key-domain (essentially, those based on the Discrete Logarithm problem over groups of known order [DF89, GJKR]). Our result and [HJJKY96] both extend the notion of *threshold cryptosystems* by incorporating mechanisms to protect against a mobile adversary.

The previous proactivization techniques do not seem to be sufficient to construct an efficient proactively-secure RSA public-key system [RSA78]. One of the problems with distributing power to perform a keyed RSA has been how to distribute the shares without revealing $\phi(N)$ (knowledge of which implies breaking the key). We cannot store the secret distributively inside a distributed circuit state (inefficient techniques of circuit evaluation as was done proactively in [OY91]) since this is inherently inefficient (not allowing circuit computation embedding in the communication is a major distinction between inefficient and efficient protocols [FY93]). Previous efficient proactivization techniques require providing the order of the share domain to the shareholders, hence these results are not useful for RSA. To resolve this problem our result generates shares over the integers. Even then using previous techniques would increase the size of the shares each time servers perform a share re-randomization to self-secure themselves, whereas we use small uniformly bounded ($O(\log N)$) share sizes.

Figure 1 depicts the development of the distribution of the RSA private key to enhance its security. It is presented in strict order of increasing security and availability. Note that our "proactive" solution is robust. That is, the RSA primitive is available and efficiently computable in the presence of adversarial shareholders. Figure 2 depicts the "proactivizing" of various cryptographic primitives.

Our contribution is a new way to distribute and maintain the RSA function (and its relatives) so that robust computation is possible at any point assuming a mobile adversary (it is also a new "robust threshold RSA" if one assumes a stationary adversary, but it permits "proactivization").

The primary techniques used in the result: We first employ a combinatorial reduction of r-out-of-r (verifiable) secret sharing (additive threshold scheme) to r-out-of-l (verifiable) secret sharing (for $r < l/(2+\epsilon)$). This probabilistic construction allows for the verifiable distribution of shares of an RSA key by a key

[F89]	(l,l)- (additive) shared RSA
[DF91]	heuristic (t,l)-shared RSA scheme
[FD92, DDFY92]	provable (t,l)-shared RSA
[FGY96, GJKR96]	Efficiently Robust (i.e. verifiable) shared RSA
Our result	First "proactivized" (vs. Mobile Adversary) shared RSA

Fig. 1. History of Increased Security of Distributed RSA

[OY91]	mobile adv. and "proactive protocol" model introduced
[CH94]	proactive pseudorandomness
[HJKY95]	proactive secret sharing
[HJJKY96]	proactive public key with public key domain (Disc. Log based)
Our result	proactivized RSA

Fig. 2. Basic results on "proactivization"

generator and also allows the re-randomization of these shares by the servers; it also simplifies the domain over which sharing is done (when compared with [DDFY92]). This construction was originally designed for a specific verifiable secret sharing scheme which is based on the quadratic residue problem modulo Blum integers [AGY95]. (We extend the construction in [AGY95], by observing that their results will hold for more general sets of good and bad servers.) We use a simulatability argument (similar to one that was put forth in the static distribution of RSA [DDFY92]) to show that the distribution of shares is secure. We then employ the idea of witness-based cryptographic program checking [FGY96] which extends Blum's methodology of program result checking [Bl88] to a system where the checker itself is not trusted by the program. We then develop specific techniques that use the RSA properties (being an exponentiation cipher, and having certain algebraic structure) that complete the design.

We prove the security and robustness of the combined system throughout its lifetime, and thus show that RSA is efficiently "proactivizable". We design efficient protocols for arbitrary numbers $k' < k$ that are each a constant fraction of the number of shareholders. We show that k uncorrupted shareholders may compute RSA signatures efficiently in the presence of corrupted shareholders. We show that k' shareholders can not learn any information about the RSA key.

The protocol is geared towards practical adaptations where a small (constant) numbers of servers is expected. In this case the probabilistic construction of server assignment in [AGY95] we can replaced by a specific assignment based on combinatorial designs.

Organization: In Section 2 we present the model and our definitions of robustness (correctness) and security in the proactive RSA model. Section 3 describes the system and its protocols. Section 4 presents the proof of robustness, and Section 5 presents the proof of security.

2 Model and Definitions

Definition 1. Let h be the security parameter. Let key generator GE define a family of RSA functions to be $(e, d, N) \leftarrow GE(1^h)$ such that N is a composite number $N = P*Q$ where P, Q are prime numbers of $h/2$ bits each. The exponent e and modulus N are made public while $d \equiv e^{-1} \bmod \lambda(N)$ is kept private.[5]

The **RSA encryption function** is public, defined for each message $M \in Z_N$ as: $C = C(M) \equiv M^e \bmod N$. The **RSA decryption function** (also called signature function) is the inverse: $M = C^d \bmod N$. It can be performed by the owner of the private key d. The security of RSA is given in Definition 11.

Our system consists of l servers $\{s_1, \ldots, s_l\}$ and a function f_x for some key x. The system is synchronized and there are two types of time periods repeated in sequence: an *update* period and an *operational* period.

We say an adversary is **(k', k, l)-restricted** if it can corrupt at most $\min\{l - k, k'\}$ servers, and view the memory of at most k' servers (including those that it corrupts) during any time period. Our system is designed such that the following properties hold:

- During an operational period, any k uncorrupted servers can efficiently compute $f_x(\alpha)$, for any α, without revealing anything about f_x other than $f_x(\alpha)$.
- The function f_x is secure against any mobile adversary that is (k', k, l)-restricted. (We assume that the adversary is computationally bounded and therefore cannot break any of the underlying cryptographic primitives used.)

Hence our system is a robust function sharing system, in which the security of the function is maintained over many function applications (as opposed to secret sharing, which is a one-time reveal operation). Moreover, our system provides security against a mobile adversary that may have access to all of the servers throughout the lifetime of the system (albeit, no more than k' simultaneously in any period). Next, we discuss informally some of the issues and assumptions in our model.

The communication model: The communication model presented here is similar to [HJKY95]. The l servers communicate via an authenticated bulletin board [CF85] in a synchronized manner. The board is accessible by a Gateway (an efficient combining function which produces the correct final result) that can be assumed to be insecure. We assume that the adversary cannot jam communication. The board assumption models an underlying basic communication protocol (authenticated broadcasts) and allows us to disregard the low-level technical details.

[5] $\lambda(N) = \mathrm{lcm}(P-1, Q-1)$ is the smallest integer such that any element in Z_N^* raised by $\lambda(N)$ is the identity element. RSA is typically defined using $\phi(N)$, the number of elements in Z_N^*, but $\lambda(N)$ can be used instead. Knowing a value which is a multiple of $\lambda(N)$ implies breaking the system.

Time periods: Time is divided into *time periods* which are determined by the common global clock (e.g., a day, a week, etc.). There are two types of time periods repeated in sequence: an **update period** (odd times) and an **operational period** (even times). During the update period the servers engage in an interactive *update protocol.* At the end of an update period the servers hold new shares (which are used during the following operational period). We consider a server that is corrupted during an update phase as being corrupted during both its adjacent periods.

Definitions of the subprotocols discussed above will be given in Section 3. We next define what it means for the system to be robust.

Definition 2. **(Robustness)** Let h be the security parameter. Let key generator GE define a family of RSA functions (i.e., $(e,d,N) \leftarrow GE(1^h)$ be an RSA instance with security parameter h). A system $S(e,d,N)$ is a (k',k,l)-robust proactive RSA system if it contains probabilistic polynomial-time protocols for the following tasks: initial (typically centralized) *share distribution* (to l servers), *RSA function application*, *share renewal* (odd steps), *lost share detection* and *lost share recovery* (even steps); such that, for any probabilistic polynomial-time (k',k,l)-restricted adversary $\mathcal{A}$, for any polynomial-size history H (described below) and for any polynomial poly$(\cdot)$,

- (correctness) with probability greater than $1 - \frac{1}{\text{poly}(h)}$, for any operational round $2t$ (where, by definition, there are at least k uncorrupted servers in each), and for any $\alpha \in [0,N]$ (to be added to L), S can compute $\alpha^d \bmod N$ using the RSA function application protocol.
- (efficiency) the computational effort is polynomial.

Next we define what does it mean for the system to be secure.

An adversary can be either "stationary" or "mobile" and we consider the mobile case here. It can be "passive" or "arbitrary" (malicious) and we consider the later. Finally it can be oblivious (with a fixed corruption strategy) adaptive (where corruptions are based solely on previous function outputs, e.g. message/signature pairs) or communication-sensitive (i.e. fully adaptive, where corruptions may be dependent on ciphertexts exchanged, check shares, and partial function evaluations). We consider the adaptive adversary. We assume that the adversary collects everything from the public channel and stores all information gained in its view during its attack on the system. It is (k',k,l)-restricted, meaning that it can corrupt at most $\min\{l-k,k'\}$ servers, and view the memory of at most k' servers (including those that it corrupts) during any time period. The adversary collects *history* H consisting of (1) a list L of message/signature/time-period tuples ordered according to time, and (2) a sequence of corruptions and releases which remembers which server is under control at which time interval based on events in L. When the adversary no longer controls a server, it is "removed" by an underlying system management (that server is started from scratch and "rebooted" by the other servers). Given this adversarial behavior we define:

Definition 3. (Security) Let h be the security parameter. Let key generator GE define a family of RSA functions (i.e., $(e,d,N) \leftarrow GE(1^h)$ be an RSA instance with security parameter h). A system $S(e,d,N)$ is a (k',k,l)-robust secure proactive RSA system if for any probabilistic polynomial-time (k',k,l)-restricted adversary $\mathcal{A}$, for any polynomial size history H (described above) and for any polynomial poly$(\cdot)$: $\Pr[u^e \equiv w \bmod N : (e,d,N) \leftarrow GE(1^h); w \in_R \{0,1\}^h; u \leftarrow A(1^h, w, \text{view}_{\mathcal{A},H}^{S(e,d,N)})] < \frac{1}{\text{poly}(h)}$.

3 The Protocol

3.1 Initialization Protocol

Family and Committee Assignments. We first distribute shares in multiple r-out-of-r secret sharing protocols. This technique is essentially from [AGY95]. The assignment of families and committees can be done by the dealer (but can also be done by the servers). Let $S = \{s_1, \ldots, s_l\}$ be the set of servers and $\mathcal{F} = \{F_1, \ldots, F_m\}$ be the set of families, where each $F_i = \{C_{i,1}, \ldots, C_{i,r}\}$ is a set of committees of servers. Each committee is of size c. Denote $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, r\}$ the indices of families and committees, respectively. The parameters m, r, and c are chosen such that the result will be a (σ, τ)-terrific assignment, that is, one that obeys the following properties **for any** set of "bad" servers $B \subseteq S$ with $|B| \leq k' \leq l\sigma$ and any set of "good" servers $E \subseteq S$ with $|E| \geq k \geq l\tau$:

1. For all $i \in I$, there exists a $j \in J$ such that $B \cap C_{i,j} = \emptyset$. (For each family there is one committee with no bad servers which we call an *excellent* committee.)
2. For at least 90 percent of $i \in I$, for all j, $E \cap C_{i,j} \neq \emptyset$. (In 90 percent of the families, all committees have at least one good server. We call a family F_i with this property a *good* family.)

Given l, q, p, and security parameter $h \geq \max\{2l+2, 100\}$, we will set $c = \lceil \{2 \log h / \log(\frac{1-\sigma}{1-\tau})\} \rceil$, $r = (1-\tau)^{-c}/h$, and $m = 10h$.

Lemma 4. *A randomly chosen assignment is (σ, τ)-terrific with overwhelming probability.*

We can set the probability of obtaining a non-(σ, τ)-terrific assignment to be smaller than that of breaking the RSA function given the security parameter.

Once the assignment is set, the servers run a protocol to generate and distribute public/private key pairs for secure communication amongst themselves. That is, these keys are used for secure (probabilistic) encryption ([GM84, L96]) which emulates a private channel between the sender of a message and the holder(s) of the private key. The protocol to perform the generation and distribution of keys is similar to the one in Section 3.3, except that the messages are authenticated with a renewable token (which is, e.g., given to the server by a trusted system administrator on booting up). As a result we can see that:

Lemma 5 [AGY95]. *The preceding protocol gives the servers in a (σ, τ)-terrific assignment a public/private key pair for each committee, and further: excellent committees have secure keys.*

Notation: For each $(i,j) \in I \times J$, $\mathrm{ENC}_{i,j}(\alpha)$ will denote an encryption of α using the public key of $C_{i,j}$. For all $s \in S$, $\mathrm{ENC}_s(\alpha)$ will denote a probabilistic encryption of α using the public key of server s. Remember, in our model the adversary is computationally bounded and thus it cannot get more than a negligible advantage in computing any function of α by seeing its encryption.

Distributing the secret Let us review the set-up protocol.

1. The dealer generates p, q, e, d, as in RSA: $N = pq$ and $ed \equiv 1 \bmod \lambda(N)$.
2. The dealer generates[6] $g \in_R [2, N-2]$ and broadcasts $[\mathrm{DISTRIBUTE.1}, N, e, g, g^d \bmod N]$.
3. For each $(i,j) \in I \times J \setminus \{r\}$, the dealer generates $a^0_{i,j} \in_R [-2Nm^3cft^2, 2Nm^3cft^2]$. Then it sets $a^0_{i,r} = d - \sum_{j \in J \setminus \{r\}} a^0_{i,j}$
4. For each $i \in I$ and $j \in J$, the dealer sets $\epsilon_{i,j} \equiv g^{a^0_{i,j}} \bmod N$
5. The dealer broadcasts $[\mathrm{DISTRIBUTE.2}, \{\epsilon_{i,j}\}_{i \in I, j \in J}, \{\mathrm{ENC}_{i,j}(a^0_{i,j})\}_{i \in I, j \in J})]$.
6. Every server checks for all $i \in I$ that $\prod_{j \in J} g^{a^0_{i,j}} \equiv g^d \bmod N$ and each server in $C_{i,j}$ checks that $\epsilon_{i,j} \equiv g^{a^0_{i,j}} \bmod N$.
7. For each $(i,j) \in I \times J$, every server sets $b^0_{i,j} = \epsilon_{i,j}$.

Rather than relying on a centralized dealer, we can employ a new result by [BF97]. It enables the generation of a distributed RSA key (N, e, d) (step 1). Given the distributed d we can distributively perform steps 2-7 (with many random generators).

3.2 Operational period (for round $2t$)

This is the protocol to be followed when the gateway obtains a message M to be signed in round $2t$. This protocol follows the one in [FGY96]. We use the fact that since $d = \sum_{j \in J} a^t_{i,j}$ then $M^d \equiv \prod_{j \in J} M^{a^t_{i,j}} \bmod N$. We also need to verify correctness of the results using a witness.

1. The gateway broadcasts $[\mathrm{SIGN.1}, M]$.
2. For all $(i,j) \in I \times J$, each server $s \in C_{i,j}$ computes $r_{i,j} \equiv M^{a^{2t}_{i,j}} \bmod N$ and broadcasts the message $[\mathrm{SIGN.2}, s, i, j, M, r_{i,j}]$.
3. For all $(i,j) \in I \times J$, each server $s' \in C_{i,j}$ checks each message $[\mathrm{SIGN.2}, s, i, j, M, r_{i,j}]$. If M is not the same message broadcast by the gateway, then s'

[6] We assume here that the order of g is maximal (i.e., $\lambda(N)$). In practice, a trusted dealer will know the factorization of $P-1$ and $Q-1$ and then be able to generate such a g with overwhelming probability.

disregards the message, else if $r_{i,j} \not\equiv M^{a^{2t}_{i,j}} \bmod N$, then s' broadcasts the challenge[7] [SIGN.CHALLENGE, $s', i, j, a^{2t}_{i,j}$].

4. All servers verify all challenges (by checking if $b^{2t}_{i,j} \equiv g^{a^{2t}_{i,j}} \bmod N$) and inform the system management of any bad servers (i.e., those servers that sent a message with $r_{i,j} \not\equiv M^{a^{2t}_{i,j}} \bmod N$).
5. For some good family i, the gateway computes $\prod_{j\in J} r_{i,j} \equiv M^d \bmod N$. There is a vast majority of good committees that will give this value.

3.3 Update period (for round $2t+1$)

So far the solution is for a stationary adversary; now we need to update and recover (needed against a mobile adversary). In the update period the public and private keys of the servers are updated, lost shares are detected and shares are updated. This is the self-maintenance portion of the proactive protocol.

Key Renewal: The public/private key pairs of each server are simply renewed as follows. Each shareholder chooses a new public/private key pair and broadcasts the new public key signed with the old private key. For each committee, the least lexicographically ordered member chooses a new committee public/private key pair and broadcasts the new public key. The server also sends the new private key, encrypted, to each other member of the committee.

Lost Share Detection:

1. Every server s sends out [LOSS.DETECT, $s, \{b^{2t}_{i,j}\}_{i\in I, j\in J}$].
2. Each server decides the correct shares by majority, and informs the system management.

Share Renewal/Lost Share Recovery: We have one protocol that handles share renewal and lost share recovery. (For efficiency, one could streamline this protocol, or separate the protocols.) Note that possibly ten percent of the families have committees that contained all bad servers who erased those committees' shares. Those families would not be able to reconstruct the secret d, and thus all the shares in those families are useless. In our protocol, these useless shares will be replaced by shares of shares from a good family. Actually, each family's shares will be replaced by shares of shares from a good family, and thus all shares will be renewed. To create the new shares for a family $F_{i'}$, every family sends shares of its shares to the committees in $F_{i'}$. $F_{i'}$ takes the shares of shares of some family (which it verifies to be valid) and creates its new shares by summing these shares of shares in each committee. This type of share recovery is unlike the share recovery protocols in previous proactive schemes. Now we give the protocol:

[7] If the server is uncomfortable providing $a^{2t}_{i,j}$ in this message, [FGY96] could be used to prove knowledge of $a^{2t}_{i,j}$ from $g^{a^{2t}_{i,j}}$ and $M^{a^{2t}_{i,j}}$. However, in our model, revealing $a^{2t}_{i,j}$ does not lessen the security.

1. Let f be a function that is super-polynomial in h. The shareholders will randomize their shares over intervals proportional to f, N, and other terms so as to maintain statistical uncertainty of the value of d. For all $(i, j, i') \in I \times J \times I$, each server s in $C_{i,j}$ does the following: s chooses $w_{s,i,j,i',j'} \in_R [-2Nm^3cft^2, 2Nm^3cft^2]$ for $j' \in J \setminus \{r\}$ and sets $c^{2t}_{s,i,j,i',j'} = w_{s,i,j,i',j'}$ for $j' \in J \setminus \{r\}$. Then s sets $c^{2t}_{s,i,j,i',r} = a^{2t}_{i,j} - \sum_{j' \in J \setminus \{r\}} c^{2t}_{s,i,j,i',j'}$. Then for all $j' \in J$, s computes $e_{s,i,j,i',j'} = \mathrm{ENC}_{i',j'}[c^{2t}_{s,i,j,i',j'}]$ and $\epsilon_{s,i,j,i',j'} = g^{c^{2t}_{s,i,j,i',j'}} \bmod N$.
2. For all $(i, j) \in I \times J$, each server s in $C_{i,j}$ broadcasts
$$[\mathrm{RECOVER.1}, s, i, j, i', \{\epsilon_{s,i,j,i',j'}\}_{(i',j') \in I \times J}, \{e_{s,i,j,i',j'}\}_{(i',j') \in I \times J}].$$
3. Every server verifies, for all $(i, j, i') \in I \times J \times I$ and all $s \in C_{i,j}$, that $\prod_{j' \in J} \epsilon_{s,i,j,i',j'} = b^{2t}_{i,j} \bmod N$, and informs the system management if it doesn't hold for some s. From this point on, we only deal with messages from those s where it does hold.
4. For all $(i', j') \in I \times J$, if $s \in C_{i',j'}$, decrypt shares to $C_{i',j'}$ and verify $g^{c^{2t}_{s',i,j,i',j'}} \equiv \epsilon_{s',i,j,i',j'} \bmod N$ for all s'. For all $(i', j') \in I \times J \setminus \{r\}$, if $s \in C_{i',j'}$, also verify $|c^{2t}_{s',i,j,i',j'}| \le 2Nm^3cft^2$ for all s'.
5. If server s finds that verification fails for a message from server s', s broadcasts
$[\mathrm{RECOVER.ACCUSE}, s, i, j, i', j', s']$, to which s' responds by broadcasting $[\mathrm{RECOVER.DEFEND}, s', i, j, i', j', c^{2t}_{s',i,j,i',j'}]$.
6. All servers check all accusations and inform the system management of any bad servers (i.e., those that defended with an invalid value of $c^{2t}_{s',i,j,i',j'}$). Again, from this point on, we only deal with messages from the good servers.
7. If $s \in C_{i',j'}$, using the shares of the lexicographically first family F_i with shares that passed verification, using the lexicographically first servers in each committee in that family with shares that passed verification (call them $s_{i,j}$), compute $a^{2t+2}_{i',j'} = \sum_{j \in J} c^{2t}_{s_{i,j},i,j,i',j'}$, and $b^{2t+2}_{i',j'} \equiv \prod_{j \in J} \epsilon_{s_{i,j},i,j,i',j'} \bmod N$.
8. Everything is erased except $a^{2t+2}_{i,j}$ and $b^{2t+2}_{i,j}$ for all $(i, j) \in I \times J$.

4 Proof of Robustness

We will show that the proactive RSA system from Section 3 is robust as defined. It will be implied by two conditions: correctness (of the function representation), and verifiability (of correctness of evaluations), throughout. We then will show that the shares remain small throughout the protocol.

Theorem 6. *The proactive RSA system above is robust against any (k', k, l)-restricted adversary $\mathcal{A}$.*

Proof. We say the system is *correct at time* $2t$ when $d \equiv \sum_{j \in J} a^{2t}_{i,j} \bmod \lambda(N)$ for all good families F_i. (We note that the majority agreement on $b^{2t}_{i,j}$ implies that all good servers in a committee $C_{i,j}$ will either agree on one share $a_{i,j}^{2t}$ or agree they have none.)

At the time of function evaluation, the gateway G is given the outputs of all the committees in all the families, the opened shares at the committees that were challenged by having contradictory outputs and the public witnesses $\{b_{i,j}^{2t}\}_{(i,j)\in I\times J}$. We say that the system is *verifiable at time* $2t$ if G can pick the correct shares of all good committees. Verifiability implies that the gateway G can identify a good family and compute $M^d \equiv \prod_{j\in J} M^{a_{i,j}^{2t}} \bmod N$ for a good family F_i; this implies efficiency. Thus, correctness and verifiability imply robustness. We omit the inductive proof that the system maintains correctness and verifiability throughout.

Next we deal with boundedness of the shares throughout the protocol. The following lemmas show that the sizes of shares are bounded (by a polynomial in h) at good committees. The initial shares (sent by the trusted dealer) are in the range $[-r2Nm^3cf, r2Nm^3cf]$, and thus are of size at most $2h+\log(m^3cf)+\log r$. (Note that this could be verified by the servers in the distribution protocol. Also note that r is bounded by a polynomial in h.)

Lemma 7. *For any $t > 0$, and any good committee $C_{i',j'}$ with $j' \in J \setminus \{r\}$, $-2Nrm^3cft^2 \leq a_{i',j'}^{2t} \leq 2Nrm^3cft^2$.*

The proof follows from the verification in step 4 of the Share Renewal/Lost Share Recovery protocol. Robustness assures that modulo the (maximal) order of g, we maintain a correct representation of the function. Next we show that if the adversary can violate a certain bound on the representation size at some step, then that adversary knows a multiple of $\lambda(N)$, and thus has broken the RSA function:

Lemma 8. *For any $t > 0$, and any good family $F_{i'}$, if $\sum_{j'\in J} a_{i',j'}^{2t} \neq d$, then the adversary can break the underlying RSA function of the system.*

Thus we conclude that:

Corollary 9. *Assuming the system has not been broken (to be shown next), the size of shares is bounded by $h + \log(r^2m^3cft^2)$.*

The complexity of the scheme is mr times that of a single RSA.

Small scale implementations are much more efficient. For example 2-out-of-3 system can be done by 2-out-of-2 sharings for the pairs: (1,2), (1,3) and (2,3). A 3-out-of-4 can be done by two 3-out-of-3 sharings for the multisets: (1 and 2,3,4) and (1,2, 3 and 4). These designs can be tuned to achieve a desirable number of messages during update and signature generation based on performance. Also in practice, the combiner can first compute a result based on a single family and check for its correctness (by applying the inverse public function). In the typical case, this first result will be correct, thus saving a great deal of computation.

5 Proof of Security

We claim that:

Theorem 10. *The proactive RSA system above is secure against any (k', k, l)-restricted adversary $\mathcal{A}$.*

We prove the security of our system (in Subsection 5.2) by constructing a simulator (in Subsection 5.1) which reduces the security of the RSA function to the security of our scheme against the mobile adversary.

Definition 11. The RSA security assumption (with respect to a history): Let h be the security parameter. Let key generator GE define a family of RSA functions (i.e., $(e, d, N) \leftarrow GE(1^h)$ be an RSA instance with security parameter h). For any probabilistic polynomial-time adversary $\mathcal{A}$, given a history H containing polynomial-size list L of messages and their signatures, for any polynomial poly$(\cdot)$: $\Pr[u^e \equiv w \bmod N : (e, d, N) \leftarrow GE(1^h); w \in_R \{0,1\}^h; u \leftarrow \mathcal{A}(1^h, w, H)] < \frac{1}{\text{poly}(h)}$.

Remark: The definition above assumed history H, namely with respect to known cleartext-ciphertext pairs. When H is empty, this is the traditional definition of security. For secure uses of RSA (probabilistic encryption or signature schemes) the definition applies for properly chosen H's.

5.1 The Simulator

Here, for the sake of proof of security, we construct SIM to simulate the view of $\mathcal{A}$ with history H in the system we construct in Section 3. For simulation purposes, SIM will assume that all servers are controlled by $\mathcal{A}$ for the maximum time that the security requirements assume they are corrupted, i.e., if $\mathcal{A}$ controls server s sometime during round $2t+1$, then it also controls s all during rounds $2t, 2t+1, 2t+2$, and if $\mathcal{A}$ controls server s sometime during round $2t$, then it also controls s all during round $2t$.

Important Notation: For each $i \in I$ and each round t, SIM sets j_i^t such that C_{i,j_i^t} is a committee which contains no servers with memory viewed by $\mathcal{A}$ during round t. (See property 1 from Section 3.1).

It is easiest to describe the simulation as follows. First assume $L = \langle (M_1, M_1^d), \ldots, (M_v, M_v^d) \rangle$ is the list of (message,signature) pairs obtained by $\mathcal{A}$ from a previous "execution of RSA". Now SIM creates the initial shares, of course, independent of secret key d. The difficulty is that consistency among all broadcasts is needed to satisfy the various test performed throughout the protocol. For all $i \in I$, for all $j \in J \setminus j_i^0$, SIM generates $a_{i,j}^{0,sim} \in_R [-2Nm^3cft^2, 2Nm^3cft^2]$ and computes $b_{i,j}^{0,sim} \equiv \epsilon_{i,j} \equiv g^{a_{i,j}^{0,sim}} \bmod N$. For any $C_{i,j_i^{2t}}$, SIM can not compute a valid value for $a_{i,j_i^{2t}}^{0,sim}$ (i.e., one that will make $\sum_{j \in J} a_{i,j}^{0,sim} = d$) or else it could compute d on its own. However, SIM needs to compute a valid value for $b_{i,j_i^0}^{0,sim}$, to pass the verification step. The simulator can do this as follows. For all $i \in I$, SIM generates $b_{i,j_i^0}^{0,sim} \equiv \epsilon_{i,j_i^0} = g^d / \prod_{j \in J, j \neq j_i^0} g_{a_{i,j}^{0,sim}}$. The rest of the protocol continues as specified.

Simulating the signing phase uses a similar approach. Again, we have the problem that SIM does not have the shares for $C_{i,j_i^{2t}}$. For each $i \in I$, for each $j \in J \setminus j_i$, each $s \in C_{i,j}$ computes $r_{i,j} \equiv M_v^{a_{i,j}^{2t,sim}} \mod N$. Now, For each $i \in I$, each $s \in C_{i,j_i^{2t}}$ computes (with the help of SIM) $r_{i,j_i^{2t}} \equiv M_v^d / \prod_{j \in J, j \neq j_i^{2t}} r_{i,j} \mod N$. The rest of the protocol continues as specified.

Share renewal and lost share recovery are more complex, but are based on the same basic idea. Key renewal is performed exactly as in the real protocol and is not discussed further.

Share Distribution Simulation The Share Distribution Simulation is similar to the Share Distributions, except that firstly the pair (g, g^d) is produced by choosing $\rho \in_R [0, N]$ and letting $g = \rho^e$ (we can repeat the simulation with poly-log number of ρ's, one of which will surely give a maximal order element), and, secondly, that the shares are produced as follows.

- For each $i \in I$, for each $j \in J \setminus \{r\}$, SIM generates $a_{i,j}^{0,sim} \in_R [-2Nm^3cft^2, 2Nm^3cft^2]$. Then it sets $a_{i,r}^{0,sim} = -\sum_{j \in J \setminus \{r\}} a_{i,j}^{0,sim}$.
- For all $i \in I$, for all $j \in J \setminus j_i^0$, SIM computes $b_{i,j}^{0,sim} \equiv \epsilon_{i,j} \equiv g^{a_{i,j}^{0,sim}} \mod N$. For all $i \in I$, SIM generates $b_{i,j_i^0}^{0,sim} \equiv \epsilon_{i,j_i^0} \equiv g^d / \prod_{j \in J, j \neq j_i^0} g^{a_{i,j}^{0,sim}} \mod N$.

Signature Simulation Simulating a signature of M during round $2t$ is done as in the Signature protocol with the following exception: Let $j_i = j_i^{2t}$.

- In step 2, for each $i \in I$, each $s \in C_{i,j_i}$ computes (with SIM's help) $r_{i,j_i} \equiv M^d / \prod_{j \in J, j \neq j_i} r_{i,j} \mod N$.

Lost Share Detection Simulation The servers perform lost share detection exactly as in the real protocol.

Share Renewal/Lost Share Recovery Simulation Simulating the Share Renewal/Lost Share Recover Protocol is done as in the original protocol except as follows. Assume it is the start of round $2t + 1$. Say $F_{i'}$ is the family whose shares must be recovered. For all $i \in I$, let $j_i = j_i^{2t}$ and $j_i' = j_i^{2t+2}$. (The following protocol assumes for each $i \in I \setminus \{i'\}$, $j_i, j_{i'} \neq r$. The other cases are similar.)

- In step 1, for all $(i, i') \in I \times I$, every $s \in C_{i,j_i}$ computes $\epsilon_{s,i,j_i,i',j_{i'}'} \equiv b_{i,j_i}^{2t,sim} / \prod_{j' \in J \setminus \{j_{i'}'\}} g^{c_{s,i,j_i,i',j'}^{2t,sim}} \mod N$.
- In step 5, for all $(i, i') \in I \times I$, no server $s \in C_{i',j_{i'}'}$ broadcasts an accusation of any server $s' \in C_{i,j_i}$.

5.2 Security proofs

First, we reduce the security to a simulator of certain properties, then we prove that the simulator constructed above indeed possesses these properties.

Main Security Lemma. The following is shown by reduction.

Lemma 12. *Let h be the security parameter. Let G be a family of RSA functions with security parameter h. Let $S(e,d,N)$ be a system that satisfies the robustness property of a proactive RSA system. If, for any probabilistic polynomial-time (k',k,l)-restricted adversary $\mathcal{A}$, and for any history H containing a polynomial-size list L of (message,signature) pairs and a polynomial-size sequence of corruptions and signature requests, there exists a probabilistic polynomial-time simulator* $\mathrm{simu}(e,N,\mathcal{A},H)$ *such that $view_{\mathcal{A},H}^{\mathrm{simu}(e,N,\mathcal{A},H)}$ is indistinguishable from $view_{\mathcal{A},H}^{\mathrm{real}(e,d,N)}$ then $S(e,d,N)$ is a (k',k,l)-secure robust proactive RSA system.*

The simulator and indistinguishability proofs. We reduce the problem to proving that the unencrypted parts of the real and simulated views are statistically indistinguishable. We use semantically secure probabilistic encryption [GM84] which the proof of the following relies on:

Lemma 13. *If the adversary $\mathcal{A}$ is restricted to probabilistic polynomial time and if the servers are using semantically secure (probabilistic) encryption (in which distinguishing between encryptions of two given messages is difficult), then:*

If views from executions of the real and simulated protocols assuming secure communication channels are statistically indistinguishable, then views from executions of the real and simulated protocols using semantically secure encryption are polynomial-time indistinguishable.

Proof omitted due to space limitation.

Lemma 14. *Assuming secure channels and $0 < \sigma < \tau < 1$, for any probabilistic $(l\sigma, l\tau, l)$-restricted adversary $\mathcal{A}$, $view_{\mathcal{A}}^{\mathrm{simu}(e,N,\mathcal{A},C)}$ is statistically indistinguishable from $view_{\mathcal{A}}^{\mathrm{real}(e,d,N,C)}$.*

Proof. To simplify the proof we make the following assumptions (without loss of overall correctness):

1. we assume that $\mathcal{A}$'s random bits are fixed. We will show that, for every assignment of $\mathcal{A}$'s random bits, the two views are indistinguishable.
2. we assume, that for all i and even times $2t$, $\mathcal{A}$ sees all shares except $a_{i,j_i^{2t}}^{2t}$. j_i^{2t} is discussed in the simulation. We let $Bad_i^{2t} = J \setminus \{j_i^{2t}\}$ be the set of indices of family F_i's shares the adversary knows at round $2t$.
3. for all i, t, i', and j', we assume $\mathcal{A}$ sees all values of $c_{s,i,j,i',j'}^{2t}$ except for $\{c_{s,i,j_i^{2t},i',j_{i'}^{2t+2}}^{2t}\}$

The view of $\mathcal{A}$ also consists of all other messages that are broadcast in each round. However, these will not include the encryptions, since we are assuming secure channels for those messages.

For random variables Y and Y' drawn from distributions $\mathcal{D}$ and $\mathcal{D}'$, respectively, we define $\mathit{diff}(Y,Y') = \sum_{v \in \mathcal{D} \cup \mathcal{D}'} |\Pr[Y = v] - \Pr[Y' = v]|$.

To prove lemma 14, we need only show that

$$\mathit{diff}(\mathrm{view}_{\mathcal{A}(L)}^{\mathrm{simu}(e,N,\mathcal{A},L)}, \mathrm{view}_{\mathcal{A}(L)}^{\mathrm{real}(e,d,N,L)})$$

is small.

Let $X = \prod_{i,j \neq j_i^0} a_{i,j}^0 \times \prod_{t,i,j,s \in C_{i,j},i',j':(j,j') \neq (j_i^{2t}, j_{i'}^{2t+2})} c_{s,i,j,i',j'}^{2t}$ and let $X^{sim} = \prod_{i,j \neq j_i^0} a_{i,j}^{0,sim} \times \prod_{t,i,j,s \in C_{i,j},i',j':(j,j') \neq (j_i^{2t}, j_{i'}^{2t+2})} c_{s,i,j,i',j'}^{2t,sim}$. Here multiplication denotes cross products.

Lemma 15. $\mathit{diff}(\mathit{view}_{\mathcal{A}(L)}^{\mathrm{simu}(e,N,\mathcal{A},L)}, \mathit{view}_{\mathcal{A}(L)}^{\mathrm{real}(e,d,N,L)}) \leq \mathit{diff}(X, X^{sim})$.

Proof omitted due to space limitation.

The following lemma completes the proof by showing that $\mathit{diff}(X, X^{sim})$ is small (again the proof is omitted in this abstract).

Lemma 16.

$$\mathit{diff}(X, X^{sim}) \leq \frac{dm^3c}{Nm^3cft^2}$$

This completes the sketch of the main steps of the proof of security of the system.

Acknowledgments We would like to thank Markus Jakobsson, Stas Jarecki, Hugo Krawczyk, and Tal Rabin for helpful discussions on proactive public key and on proactive RSA. We thank Kevin McCurley for many helpful discussions on number theory. Finally, thanks to Nancy Irwin for her implementation which demonstrated the workings of the protocols presented in this paper.

References

[AGY95] N. Alon, Z. Galil and M. Yung, *Dynamic-resharing Verifiable Secret Sharing*, European Symposium on Algorithms (ESA) 95, Springer-Verlag LNCS.

[B79] G.R. Blakley, *Safeguarding Cryptographic Keys*, AFIPS Con. Proc (v. 48), 1979, pp 313–317.

[Bl88] M. Blum, *Designing programs to check their work*, ICSI Technical report TR-88-009.

[B88] C. Boyd, *Digital Multisignatures*, IMA Conference on Cryptography and Coding, Claredon Press, 241–246, (Eds. H. Baker and F. Piper), 1986.

[BF97] D. Boneh and M. Franklin, *Efficient Generation of Shared RSA Keys*, Crypto 97 (these proceedings).

[CH94] R. Canetti and A. Herzberg, *Maintaining Security in the presence of transient faults*, Crypto '94, Springer-Verlag, 1994.

[CF85] J. Cohen and M. Fischer, *A robust and verifiable cryptographically secure election scheme*, FOCS 85.

[DDFY92] A. De Santis, Y. Desmedt, Y. Frankel and M. Yung, *How to Share a Function Securely*, ACM STOC 94. (First version May 92).

[DF89] Y. Desmedt and Y. Frankel, *Threshold cryptosystems*, Crypto '89 Springer-Verlag, 1990.

[DF91] Y. Desmedt and Y. Frankel, *Shared generation of authenticators and signatures*, Crypto 91, Springer-Verlag LNCS 576, 1992, pp. 307–315.

[F87] P. Feldman, *A Practical Scheme for Non-Interactive Verifiable Secret Sharing*, Proc. of the 28th IEEE FOCS pp. 427-437, 1987

[F89] Y. Frankel, *A practical protocol for large group oriented networks*, Eurocrypt '89, Springer-Verlarg LNCS 773, pp. 56-61.

[FD92] Y. Frankel and Y. Desmedt. *Distributed reliable threshold multisignatures*, Tech. Report version TR–92–04–02, Dept. of EE & CS, Univ. of Wisconsin-Milwaukee, April 1992.

[FGY96] Y. Frankel, P. Gemmell and M. Yung, *Witness-based Cryptographic Program Checking and Robust Function Sharing* Proc. of STOC 1996, pp. 499–508.

[FY93] M. Franklin and M. Yung, *Secure and Efficient Digital Coin*, ICALP 93, Springer Verlag LNCS.

[GHY] Z. Galil, S. Haber and M. Yung, *Minimum-Knowledge Interactive Proofs for Decision Problems*, SIAM Journal on Computing, vol. 18, n.4, pp. 711–739. (Previous version in FOCS 85).

[GJKR] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, *Robust Threshold DSS Signatures*, Eurocrypt 96.

[GJKR96] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, *Robust and Efficient Sharing of RSA*, Crypto 96.

[GM84] S. Goldwasser and S. Micali, *Probabilistic Encryption*, J. Comp. Sys. Sci. 28, 1984, pp. 270-299.

[HJKY95] A. Herzberg, S. Jarecki, H. Krawczyk and M. Yung, *How to Cope with Perpetual Leakage, or: Proactive Secret Sharing*, Crypto 95.

[HJJKY96] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, *Proactive public key and signature systems*, The 4-th ACM Symp. on Comp. and Comm. Security. April 1997.

[L96] M. Luby, *Pseudorandomness and its Cryptographic Applications*, Princeton University Press, 1996.

[OY91] R. Ostrovsky and M. Yung, *How to withstand mobile virus attacks*, ACM Symposium on Principles of Distributed Computing (PODC), 1991, pp. 51-61.

[RSA78] R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signature and Public Key Cryptosystems*, Comm. of the ACM, 21 (1978), pp. 120-126.

[S79] A. Shamir. *How to share a secret*, Comm. of the ACM, 22 (1979), pp. 612-613.