Proactive RSA

Yair Frankel[*] Peter Gemmell[†] Philip D. MacKenzie[‡] Moti Yung[§]

August 4, 1996

Abstract

The notion of "proactive security" of basic primitives and cryptosystems that are distributed amongst various servers, was introduced in order to 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 unable to corrupt too many participants during any short time period [OstrovskyYung]. The notion 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; RSA is perhaps the most important trapdoor function in use. Employing new combinatorial and elementary number theoretic techniques, our protocol enables the dynamic updating of the servers (which hold the RSA key distributively); it is secure even when a linear number of the servers are corrupted during any time period (linear redundancy); it efficiently "self-maintains" the security of the function and its messages (ciphertexts or signatures); and it enables continuous availability, namely, correct function application using the shared key is possible at any time.

We present an efficient way in which l servers can share an RSA private function so that, given $0 < \sigma < \tau < 1$:

- *Proactive (Dynamic) Robustness*: A *gateway* G can combine information from any set of $l\tau$ (honest) servers to deduce the RSA signature for any authorized message at any period.
- *Proactive Security (against mobile adversary):* Our protocol is secure against a polynomial time adversary who controls the gateway G and time-variant sets of up to $\min\{l(1-\tau), l\sigma\}$ servers, and can obtain the shares of up to $l\sigma$ servers (including those that it corrupts).
- *Uniform Boundedness:* The share-size is always bounded by the size of an RSA private key (i.e., logarithmically in N).

We also present special practical instances based on designs; some of these instances were recently implemented as part of a highly secure application testbed at Sandia National Laboratories.

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).

[*]Sandia National Labs, P.O Box 5800, Albuquerque, NM 87185-1110,yair@cs.sandia.gov
[†]Sandia National Labs, P.O Box 5800, Albuquerque, NM 87185-1110,psgemme@cs.sandia.gov
[‡]Sandia National Labs, P.O Box 5800, Albuquerque, NM 87185-1110,philmac@cs.sandia.gov.
[§]IBM T. J. Watson Research Center, Yorktown Heights, NY, moti@watson.ibm.com, moti@cs.columbia.edu

1 Introduction

In this work we deal with algorithmic mechanisms to secure the private memory of the RSA public key system via distribution and active communication. The notion of "proactive security" [OY91] assumes a very strong "mobile adversary" who may corrupt **all** participants (servers, each with private memory) throughout the lifetime of the system in a non-monotonic fashion (i.e. recoveries are possible) but the adversary is not able to corrupt too many participants during any short period of time. The servers engage in a "proactive maintenance" that self-secures them against a mobile adversary who tries to learn the secret or disrupt their operation. 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 easily replaced (e.g., basic cryptographic infrastructure functions). In addition to protection, the proactive approach provides a flexible (dynamic) key management method that implements server changes: As companies change (mergers, firing of executives, etc.) and governments change (common in the diplomatic arena), trust relations and thus server ownership must change.

A number of very useful cryptographic mechanisms have been efficiently "proactivized", such as pseudorandomness and secret sharing [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). The techniques needed to achieve the protection in these works allow the dynamic maintenance of the global representation of the cryptographic tool and the key, while allowing local re-randomization of the distributed representation of the key. They are all efficient and direct; i.e., they do not use secure distributed computing compilers which embed a cryptographic circuit in the communication protocol, but rather they employ a few communication rounds for each operation and update protocol. The distinction of efficient (communication independent of computation/circuit complexity) vs. inefficient (computation embedded in communication) within the realm of cryptographic protocols (that are all polynomial-time) was put forth in [FY93].

The previous proactivization techniques do not seem to be sufficient to proactively maintain the security of an RSA public-key system [RSA78] (perhaps the most popular public key system today). 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 ([OY91]) since this is inherently inefficient. Previous proactivization techniques which do not embed a circuit 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 proactivization techniques would increase the size of the shares each time servers perform a share re-randomization to self secure themselves (since the integers are not a finite group and the adversary can force shares to grow in size). Our result has small uniformly bounded ($O(\log N)$) share sizes.

In Figure 1, a perspective of the development of the distribution of the RSA function memory (to enhance the secure memory requirement of the private portion of public keys). It is presented in a strictly increased security and availability order. Note that our "proactive" solution is, in particular, robust. In Figure 2, a perspective of "proactivizing" of various cryptographic primitives is given.

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. The protocol combines a number of new ideas and employs technologies that have been developed for other purposes. We capitalize on the fact that these diverse techniques (combinatorial, algebraic, cryptographic, program-checking and protocol techniques) are combined and augmented in a new way to achieve an efficient cryptographic protocol.

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

Figure 1: History of Increased Security of Distributed RSA

Ostrovsky-Yung[OY91]	mobile adv. "proactive protocol" introduced
CannettiHerzberg[CH94]	proactive pseudorandomness
[HJKY95]	proactive secret sharing
[HJJKY96]	proactive (Disc. Log based) public key with publicly known key domain
Our result	proactivized RSA

Figure 2: Basic results on "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 (r strictly less than l). This construction allows for the verifiable distribution of shares of an RSA key by a key 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 exponentiation cipher, and having certain algebraic structure) that complete the design.

We prove the security of the combined system throughout its life time, and thus show that RSA is efficiently "proactivizable". We show that, in fact, a minority of processors (a linear fraction of them which is bounded away from half, e.g. 1/3) can be corrupted at any period, yet the function is available and can be robustly computed at any time and its security is maintained throughout.

In practice, for a small (constant) numbers of servers, we can replace the probabilistic assignment of shares to servers in [AGY95] by a specifically designed assignment. This case was suggested and investigated first in [JJKY95] (where a solution with logarithmically growing shares was given). In Appendix E, we discuss these practical assignments. We note that Sandia National Laboratories has implemented this proactive RSA implementation to demonstrate proof of concept of its practicality.

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(correctness) whereas Section 5 presents the proof of security.

2 Model and Definitions

We now discuss the proactive model and several important definitions used in this paper.

Definition 2.1 *Let h be the security parameter. Let key generator GE define a family of RSA functions to be $(e, d, N) \leftarrow GE(1^h)$ where N is a composite number $N = P * Q$, where P, Q are prime numbers of $h/2$ bits each. The exponent e and N are made public, while $d \equiv e^{-1} \bmod \lambda(N)$ is kept private.*[1]

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 this function is formally defined in Definition A.1*

Our system model is similar to that of the proactive secret sharing model of [HJKY95, HJJKY96] (motivated by the initial modeling in [OY91]), but modified to incorporate the function sharing model (for RSA) of [DDFY92] (motivated by the idea of threshold cryptography [DF89]). The 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. The intent of the system is reflected in two properties:

Function sharing: When k uncorrupted servers are active, the shared function f_x can be reconstructed to compute $f_x(\alpha)$ (availability), yet nothing about f_x is revealed other than $f_x(\alpha)$ (security). (Extends secret sharing [B79, S79] to functions which can be applied many times.).

Mobile adversary property: We assume that our adversary is computationally bounded and therefore can not break any of the underlying cryptographic primitives used. An adversary $\mathcal{A}$ can have control of any server at any time period and we consider a server that is corrupted during an update phase as being corrupted during both its adjacent periods. A corrupt server gives the adversary access to its memory and acts arbitrarily. A server that recovers from the adversary restarts from a fresh state and we assume it can authenticate itself (modeling hardware protected smart-card authentication at reboot). We assume the adversary 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.

Hence, unlike secret sharing (which is a one-time reveal operation), security is maintained and the secrecy of the function holds throughout, even though the mobile adversary may have access to all of the servers throughout the lifetime of the system (albeit, no more than k' simultaneously at a round).

In what follows we discuss informally some of the issues and assumptions in our model.

The communication model: The communication model 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 (efficient combining function which produces the correct final result) that can be assumed to be an insecure gateway. 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.

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).

[1] $\lambda(N) = \text{lcm}(P-1, Q-1)$ 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.

We now give a more formal model of correctness and security for proactive RSA (which can easily be generalized to other one-way or trapdoor functions).

For a polynomial time adversary $\mathcal{A}$ and a polynomial-size *history* H consisting of (1) message/signature pairs, L, obtained, and (2) a sequence of corruptions and signature requests (from messages in L) that the adversary will perform, let $\text{view}^S_{\mathcal{A},H}$ be the view that $\mathcal{A}$ discovered during the process of attacking the system S. The $\text{view}^S_{\mathcal{A},H}$ is constructed as follows.

1. When a message is broadcast in the system, that message is appended to $\text{view}^S_{\mathcal{A},H}$.
2. When $\mathcal{A}$ takes control of a server s, the memory of s is appended to $\text{view}^S_{\mathcal{A},H}$. When $\mathcal{A}$ leaves server s this is recorded in the view.
3. When $\mathcal{A}$ reads the memory of (but doesn't control) server s, the memory of s is appended to $\text{view}^S_{\mathcal{A},H}$.
4. The adversary computes and is allowed to output changes to the memory of some server s while $\mathcal{A}$ controls s (of course, the change is recorded in $\text{view}^S_{\mathcal{A}(L)}$).

Below we define the robustness function which assures the availability of the function. Informally, it states that the function can be produced at any time, as there is a correct representation of the secret available at good servers (correctness), and misbehavior is caught or prevented from influencing the outcome (verifiability). The protocol is to be polynomial time throughout and the information available at servers should be kept small (rather than allowed to grow exponentially, say). In fact the protocol presented in this paper assures boundedness throughout.

Definition 2.2 (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 (or correct) proactive RSA system if it contains probabilistic polynomial-time protocols for the following tasks: initial 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 above) and for any polynomial poly$(\cdot)$,*

- *with probability greater than $1 - \frac{1}{poly(h)}$, 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.*

The definitions and explanations of the subprotocols in the definitions above will be given in Section 3.

Next we define what does it mean for the system to be secure. The adversary also collects everything from the public channel and stores all information gained in its view during its attack on the system. The adversary has a *history* H consisting of (1) message/signature pairs, L, obtained before the system is run, and (2) a sequence of corruptions and signature requests (from messages in L) that the adversary will perform. The adversary operates dynamically, changing the memory of servers it controls, or forcing servers it controls to send messages, depending on the current $\text{view}_{\mathcal{A}}$. It, however, is non-adaptive in its choice of when and which servers to corrupt, and when and which signature requests to perform. Also note that when the adversary no longer controls a server, it is "removed" by an underlying system management (that server is "rebooted" by the other servers). We assume at that point that the server is authenticated as a rejoining server to the system (as any cryptographic system must rely on initial minimal authentication of parties).

Definition 2.3 (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 $poly(\cdot)$:*

- $\Pr[u^e \equiv w \bmod N : (e,d,N) \leftarrow GE(1^h); w \in_R \{0,1\}^h; u \leftarrow A(1^h, w, view_{\mathcal{A},H}^{S(e,d,N)})] < \frac{1}{poly(h)}$.

3 The Protocol

3.1 Outline

Our protocol is a combination of combinatorial, number theoretic, cryptographic, program checking and protocol techniques.

As in [F89], we use an r-out-of-r scheme but employ a reduction to a method that allows a linear fraction of faults at any time. During the *initialization phase* (Section 3.2) a trusted dealer computes a private RSA key d and public key (e,N) [RSA78]. It duplicates the private key m times, and the ith duplicate is divided into r shares $\{a_{i,j}^0\}_{j\in\{1,\ldots,r\}}$ such that for all i, $d = \sum_{j=1}^r a_{i,j}^0$. In general, $a_{i,j}^t$ will denote the jth share of the ith duplicate key during round t, and for all i and t, $d = \sum_{j=1}^r a_{i,j}^t$. Each share will be associated with a subset of servers such that our system will perform as required. In particular, no subset of servers of size $k' \leq l\sigma$ will be associated with all the shares of any duplicate of the private key, and every subset of servers of size $k \geq l\tau$ will be associated with all the shares of some duplicate of the private key. The assignment (initially given in [AGY95]) into subsets can be chosen at random by the dealer (or by the servers efficiently). To achieve function sharing on an input message M, a gateway G obtains from the servers possessing $a_{i,j}^t$ the value $M^{a_{i,j}^t}$, and hence G can compute $f_{d,N}(M) \equiv M^d \equiv \prod_{j=1}^r M^{a_{i,j}^t} \bmod N$ during the *signing phase* (Section 3.3) as in [F89].

The shares will be unchanged during intervals of time that we call operational periods, and between these operation periods there will be *update periods* (Section 3.4) where (a) the good shares are renewed and (b) corrupted shares are recovered. The basic technique to renew and recover shares is by creating shares of shares and distributing the share of shares appropriately.

To provide for robustness throughout the protocol, we give a witness (similar to [FGY96]) to check intermediate function results that provide the proactivization of the protocol. For instance we must assure that everyone gets valid shares of d during update periods.

Remark: In all of these protocols messages are put on authenticated bulletin board. One implementation is having every message signed by the sender using a secure signature scheme, and any message with an invalid signature is ignored. The first field in any message is simply a tag to indicate the type of message. Individual signature keys are renewed throughout, though, as was mentioned, we need an initial authentication token to be available for recovering servers. We can assume that all the signed messages originate at the correct server; otherwise, the assumption about the security of the underlying signature scheme is violated.

3.2 Initialization Protocol

3.2.1 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. Let $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, r\}$ be 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 3.1 *A randomly chosen assignment is (σ, τ)-terrific with overwhelming probability.*

The lemma is similar to Proposition 3 in [AGY95]. We can control the probability of obtaining a non-(σ, τ)-terrific assignment to be smaller than that of breaking the RSA function given the security parameter. Note that once we have terrific assignment, any choice of "bad servers" is allowed– which is important in the mobile adversary case.

The assignment communication protocol runs as follows:

1. The servers generate public/private key pairs, and broadcast their public keys (authenticated with their renewable authenticated token– modeling their trusted hardware based mechanism or trusted channels).
2. The server with the smallest ID in each committee generates a public/private key pair, broadcasts the public key, and broadcasts encrypted copies of the private key using the public keys of the other servers in the committee.

The keys distributed are used in a secure (probabilistic) encryption ([GM84, L96]) so that they generate private channels between the sender of messages and the holder(s) of the private key.

As a result we can immediately see that:

Lemma 3.2 ([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 is assumed that it does not have the capability to break these underlying primitives, i.e. it cannot get more than a negligible advantage in computing any function of α by seeing the encryption of it.

3.2.2 Distributing the secret

The dealer generates an RSA instance (e, d, N) and broadcasts a witness (g, g^d) of the function to the servers, where g has maximal order, $\lambda(N)$. This witness is used later on to determine if servers have correct shares (witness-based checking [FGY96] is based on the fact that having some random function values does not help in breaking the cryptographic function). The dealer then broadcasts a message which contains, for each committee $C_{i,j}$, the $g^{a^0_{i,j}}$ (used as a public witness to the share $a^0_{i,j}$) and an encrypted $a^0_{i,j}$ (using $C_{i,j}$'s public key). The correctness of the shares is verified (with respect to the function witness g^d).

We can assume that the initial distribution is correct as we trust the dealer. Then, although we assume that the dealer is trusted, we could design a protocol in which the servers can actually guarantee the robustness of the system (i.e., that they can sign messages properly), even if the dealer acts incorrectly. (We cannot, however, guarantee the security of the system, since the dealer could simply send the value d to the adversary.)

The correctness of the shares is verified (with respect to the function witness g^d) using, e.g., a procedure similar to one in Feldman [F87]. Let us review the dealer's protocol.

1. The dealer generates p, q, e, d, as in RSA. Thus $N = pq$ and $ed \equiv 1 \bmod \lambda(N)$.
2. The dealer generates[2] $g \in_R [2, N-2]$ and broadcasts $[\text{DISTRIBUTE.1}, N, e, g, g^d \bmod N]$.
3. For each $(i,j) \in I \times J \setminus \{r\}$, the dealer generates $a^0_{i,j} \in_R [-N^2, N^2]$. 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 $[\text{DISTRIBUTE.2}, \{\epsilon_{i,j}\}_{i \in I, j \in J}, \{\text{ENC}_{i,j}(a^0_{i,j})\}_{i \in I, j \in J})]$.
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 $b^0_{i,j} = \epsilon_{i,j}$.

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

---

[2] 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.

[3] 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.

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

So far the solution is for a static adversary, now we need to update and recover. 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.

1. Assume server s has public/private key pair (u, v). It chooses a new public/private key pair (u', v') and broadcasts [UPDATE.SERVER.KEY, s, u']. (Note that the signature of this message uses the old private key.)
2. If any server detects two messages from any server s, it informs the system management.

The public/private key pairs of each committee are renewed as follows (after the public/private keys of the servers have been renewed).

1. Assume committee $C_{i,j}$ has public/private key pair (x, y). Also assume server s is the lowest numbered server on $C_{i,j}$ (that has not been declared "bad" by a majority of the servers) with (new) public/private key pair (u', v').
2. Server s creates a new public/private key pair (x', y') and broadcasts [UPDATE.C.KEY.1, s, i, j, x'].
3. Server s also broadcasts [UPDATE.C.KEY.2, $s, \{\mathrm{ENC}_{s'}(y')\}_{s' \in C_{i,j}}$].
4. Each server $s' \in C_{i,j}$ verifies that (x', y') is a valid public/private key pair. If it is not, s' broadcasts the challenge [UPDATE.C.KEY.CHALLENGE, s', i, j], to which s responds with [UPDATE.C.KEY.DEFEND, s, i, j, y'],
5. All servers verify all challenges (by checking if (x', y') is not a valid public/private key pair) and inform the system management of any bad servers (i.e., those that sent out a defend message with an invalid y'). Note that committees with a bad server distributing the new committee key must run the committee key renewal protocol again.

Lost Share Detection:

1. Every server s sends out [LOSS.DETECT, $s, \{b_{i,j}^{2t}\}_{i \in I, j \in J}$].
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. In [HJKY95], the properties of secret-sharing polynomials are used to recover a lost share using other servers' shares. By "blinding" the secret-sharing polynomials with polynomials that evaluate to zero at the appropriate point, the lost share can be recovered while no information is revealed. The closest analogue to this approach in our system would be to recover a bad family's shares by having another family blind its own shares (by shares that add to zero), and sending these "blinded shares" to the bad family. However, it can be shown that this type of approach is insecure. In [AGY95], each committee is required to share its share in the same fashion as the original secret, so its share can be recovered if it is lost. This requires each server to have a very large secure memory to hold all the shares of shares for the duration of a round. In our protocol, only the original shares need to be stored at the servers.

Now we describe the protocol

1. For all $(i,j,i') \in I \times J \times I$, each server s in $C_{i,j}$ does the following: s chooses $w_{s,i,j,i',j'} \in_R [-N^2, N^2]$ 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
$$[\text{RECOVER.1}, s, i, j, i', \{\epsilon_{s,i,j,i',j'}\}_{(i',j') \in I \times J}, \{e_{s,i,j,i',j'}\}_{(i',j') \in I \times J}].$$
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 $|c^{2t}_{s',i,j,i',j'}| \leq N^2$ for all s'.
5. If server s finds that verification fails for a message from server s', s broadcasts $[\text{RECOVER.ACCUSE}, s, i, j, i', j', s']$, to which s' responds by broadcasting $[\text{RECOVER.DEFEND}, s', i, j, i', j', c^{2t}_{s',i,j,i',j'}]$.
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 will then show that the sizes of the shares are bounded throughout.

Theorem 4.1 *The proactive RSA system above is robust against any (k', k, l)-restricted adversary $\mathcal{A}$.*

Proof: See Appendix B. □

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 $[-rN^2, rN^2]$, and thus are of size at most $2h + \log r$. (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 4.1 *For any $t > 0$, and any good committee $C_{i',j'}$ with $j' \in J \setminus \{r\}$, $-rN^2 \le a^{2t}_{i',j'} \le rN^2$.*

Proof: See the verification in step 4 of the Share Renewal/Lost Share Recovery protocol. $\Box$

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 has broken the RSA function.

Lemma 4.2 *For any $t > 0$, and any good family $F_{i'}$, if $\sum_{j' \in J} a^{2t}_{i',j'} \ne d$, then the adversary can break the underlying RSA function of the system.*

Proof: Let $2t$ be the first time that for some good family $F_{i'}$, $\sum_{j' \in J} a^{2t}_{i',j'} \ne d$. From Theorem 4.1, $\sum_{j' \in J} a^{2t}_{i',j'} \equiv d \bmod \lambda(N)$.

Let F_i be the family used by $F_{i'}$ in update round $2t-1$ to construct the shares $\{a^{2t}_{i',j'}\}_{j' \in J}$. It is verified in the Share Renewal/Lost Share Recovery protocol that for all $j \in J$, $b^{2t-2}_{i,j} \equiv \prod_{j' \in J} \epsilon^{2t-2}_{s,i,j,i',j'} \bmod N$, and that for all $j, j' \in J$, $\epsilon^{2t-2}_{s,i,j,i',j'} \equiv g^{c^{2t-2}_{s,i,j,i',j'}} \bmod N$. Since $g^{a^{2t-2}_{i,j}} \equiv b^{2t-2}_{i,j} \bmod N$, $a^{2t-2}_{i,j} \equiv \sum_{j' \in J} c^{2t-2}_{s,i,j,i',j'} \bmod \lambda(N)$. However,

$$\sum_{j' \in J} a^{2t}_{i',j'} = \sum_{j,j' \in J} c^{2t-2}_{s,i,j,i',j'} \ne d = \sum_{j \in J} a^{2t}_{i,j}.$$

Therefore, for some $j \in J$, $a^{2t-2}_{i,j} \ne \sum_{j' \in J} c^{2t-2}_{s,i,j,i',j'}$, i.e. $\sum_{j' \in J} c^{2t-2}_{s,i,j,i',j'} = a^{2t-2}_{i,j} + k\lambda(N)$, for some $k \ne 0$. This implies the adversary knows a multiple of $\lambda(N)$, which further implies that it can factor and break the underlying RSA function [RSA78]. $\Box$

Corollary 4.1 *Assuming the system has not been broken (to be shown next), the size of shares is bounded by $2h + 2\log r$.*

Proof: If the system has not been broken, then the underlying RSA function has not been broken. Thus, by Lemma 4.2, for every good family F_i, $\sum_{j \in J} a^{2t}_{i,j} = d$. From Lemma 4.1, for every $j \in J \setminus \{r\}$, $-rN^2 \le a^{2t}_{i,j} \le rN^2$. Then $d - (r-1)rN^2 \le a^{2t}_{i,r} \le d + (r-1)rN^2$. Since $d \le N$, $-r^2N^2 \le a^{2t}_{i,r} \le r^2N^2$, and thus the share size is bounded by $2h + 2\log r$. $\Box$

5 Proof of Security

The proof of security is a unique combination of combinatorial, number theoretic, and cryptographic proof techniques. Note that RSA, as a trapdoor function, is assumed secure, as explained formally in Appendix A.

We claim that:

Theorem 5.1 *The proactive RSA system above is secure against any (k', k, l)-restricted adversary $\mathcal{A}$.*

We prove the security of our system by constructing a simulator which reduces the security of the RSA function to the security of our scheme against the mobile adversary. The main security lemma and its proof are in Section D.1. It states that if there is a simulator, the system is secure (as explained formally in Definition 2.3). In Section C we construct a simulator, and in Section D we present the involved technical proof of the properties of a simulator, which completes the above theorem. The proofs are very important part of this work, but are given in the Appendix due to space limitations.

Acknowledgements

We would like to thank Markus Jakobsson, Stas Jarecki, Hugo Krawczyk for helpful discussions on proactive public key and on proactive RSA. We thank Kevin McCurley for many helpful discussions on number theory and Tal Rabin for her comments about "good guys being honest but not stupid" which led to the further protocol refinement discussed in Footnote 3 and Section F. 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*, ESA 95.

[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*, Cryptography and Coding, Claredon Press, 241–246, (Eds. H. Baker and F. Piper).

[CH94] R. Cannetti and A. Herzberg, *Maintaining Security in the presence of transient faults*, In Y. Desmedt, editor, *Advances in Cryptology, Proc. of Crypto '94 (Lecture Notes in Computer Science 839)*, pages 425–438. Springer-Verlag, 1994.

[CF85] J. (Benaloh) Cohen and M. Fischer, *A robust and verifiable cryptographically secure election scheme*, Proc. 26th Annual Symposium on the Foundations of Computer Science, 1985, pp. 372–382.

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

[DF89] Y. Desmedt and Y. Frankel, *Threshold cryptosystems*, In G. Brassard, editor, *Advances in Cryptology, Proc. of Crypto '89 (Lecture Notes in Computer Science 435)*, pp. 307–315. Springer-Verlag, 1990.

[DF91] Y. Desmedt and Y. Frankel, *Shared generation of authenticators and signatures*, Advances in Cryptology–Proc. of 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 Symposium on the Foundations of Computer Science, pp. 427-437, 1987

[F89] Y. Frankel, *A practical protocol for large group oriented networks*, In J. J. Quisquater and J. Vandewalle, editor, *Advances in Cryptology, Proc. of Eurocrypt '89, (Lecture Notes in Computer Science 773)*, Springer-Verlarg, pp. 56-61.

[FGY96] Yair Frankel, Peter Gemmell, Moti Yung, *Witness-based Cryptographic Program Checking and Robust Function Sharing* Proc. of Symposium on the Theory of Computation, 1996, pp. 499–508.

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

[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).

[GJKR96] R. Genaro, 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*, draft.

[JJKY95] M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, *Proactive RSA distribution with constant shareholders*, manuscript.

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

[OY91] R. Ostrovsky and M Yung, *How to withstand mobile virus attacks*, Proc. of the 10th ACM Symposium on the Principles in Distributed Computing, 1991, pp. 51-61.

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

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

A RSA security

Definition A.1 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 $poly(\cdot)$:

- $\Pr[u^e \equiv w \bmod N : (e,d,N) \leftarrow GE(1^h); w \in_R \{0,1\}^h; u \leftarrow A(1^h, w, H)] < \frac{1}{poly(h)}$.

Remark: The definition above assumed security with respect to a history H, namely with respect to known cleartext-ciphertext pairs. When H is empty, this is the traditional definition of security of RSA. We have chosen the known ciphertext case to simplify the proof by reduction. In secure applications (secure signatures and encryption and in secure protocols) H consists of cleartext-ciphertext pairs where ciphertexts (signature values) are from a polynomial-time samplable distribution, which is equivalent to the empty H case, due to the public availability of the encryption function. We chose this definition, since the application (protocol or cryptosystem) should define the content of the history with respect to a single signer; whereas the goal of our security proof is to show that when the signer is replaced by the servers, the security is preserved.

B Robustness Proofs

Proof of Theorem 4.1:

Proof: We say the system is *correct at time 2t* when $d \equiv \sum_{j\in J} a_{i,j}^{2t} \bmod \lambda(N)$ for all good families F_i. (We note that the majority agreement on $b_{i,j}^{2t}$ 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.)

We say that the system is *verifiable at time 2t* if given the outputs of all the committees in all the families, the opened shares at committees that were challenged by having contradictory outputs: $\{a_{i,j}^{2t}\}$ $\text{challenged}_{C_{i,j}}$, and the public witnesses $\{b_{i,j}^{2t}\}_{(i,j)\in I\times J}$; the gateway G can pick the correct shares of all good committees. Verifiability implies that the gateway G can identify good family, and compute $M^d \equiv \prod_{j\in J} M^{a_{i,j}^{2t}} \bmod N$ for a good family F_i.

Together, correctness and verifiability imply robustness.

With overwhelming probability we have an assignment that for any choice of $k \geq l\tau$ (at any time unit) we have 90 percent of the families being good. The existence of a good family is guaranteed by the assignment of servers to committees due to Lemma 3.2.

Assume correctness at time $2t$, then for each committee $C_{i,j}$ with bad servers, our protocol allows a good server on that committee to prove its function-evaluation share, $M^{a^{2t}_{i,j}}$, to be correct (by verifiability, since if there are differences, the share itself is opened). Thus, G can collect the right exponents and indeed produce a (correct) signature.

Therefore, we need only prove (by induction on t) that $\forall t$, the system is correct and verifiable at round $2t$.

Since we assume the dealer is trusted (i.e., acting correctly), then the initial distribution of shares is defined in such a way that for any (good) family F_i, the shares sent to the committees of F_i will have the property that $d \equiv \sum_{j\in J} a^0_{i,j} \bmod \lambda(N)$, and there will be a good server on each committee $C_{i,j}$ to store the value $a^0_{i,j}$. Thus, the system is correct at round 0. Also, the public values $\{b^0_{i,j}\}_{(i,j)\in I\times J}$, are correct (i.e., for all committees $C_{i,j}$, $b^0_{i,j} \equiv g^{a^0_{i,j}} \bmod N$), and g is a maximal order element $\bmod N$. Then, using checking arguments as in [FGY96], the verifiability is assured. (Note that $\lambda(N)$ is the maximal order.)

For the inductive step – the shares of a family $F_{i'}$ can change only during the Share Renewal/Lost Share Recovery Protocol. Let F_i be the family used by $F_{i'}$ in update round $2t-1$ to construct the shares $\{a^{2t}_{i',j'}\}_{j'\in J}$. It is verified in the Share Renewal/Lost Share Recovery protocol that for all $j \in J$, $b^{2t-2}_{i,j} \equiv \prod_{j'\in J} \epsilon^{2t-2}_{s,i,j,i',j'} \bmod N$, and that for all $j, j' \in J$, $\epsilon^{2t-2}_{s,i,j,i',j'} \equiv g^{c^{2t-2}_{s,i,j,i',j'}} \bmod N$. By induction, it is guaranteed that $\prod_{j\in J} b^{2t-2}_{i,j} \equiv g^d \bmod N$. This maintains verifiability. Then, since g is of maximal order $(\lambda(N))$, $\sum_{j'\in J} a^{2t}_{i',j'} \equiv \sum_{j,j'\in J} c^{2t-2}_{s,i,j,i',j'} \equiv d \bmod \lambda(N)$. This assures maintenance of correctness. (Note that if g is of order $v < \lambda(N)$, then we can only prove that $\sum_{j'\in J} a^{2t}_{i',j'} \equiv d \bmod v$, and it might not be the case that $\sum_{j'\in J} a^{2t}_{i',j'} \equiv d \bmod \lambda(N)$.) $\square$

C 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^t_i such that C_{i,j^t_i} is a committee which contains no servers with memory viewed by $\mathcal{A}$ during round t. (See property 1 from Section 3.2.1).

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 $j \in J \setminus j^0_i$, SIM generates $a^{0,sim}_{i,j} \in_R [-N^2, N^2]$ and computes $b^{0,sim}_{i,j} \equiv \epsilon_{i,j} \equiv g^{a^{0,sim}_{i,j}} \bmod N$. For any C_{i,j^{2t}_i}, SIM can not compute a valid value for $a^{0,sim}_{i,j^{2t}_i}$ (i.e., one that will make $\sum_{j\in J} a^{0,sim}_{i,j} = d$) or else it could compute d on its own. However, SIM needs to compute a valid value for $b^{0,sim}_{i,j^0_i}$, to pass the verification step. The simulator can do this as follows. For all $i \in I$, SIM generates $b^{0,sim}_{i,j^0_i} \equiv \epsilon_{i,j^0_i} = g^d / \prod_{j\in J, j\neq j^0_i} g_{a^{0,sim}_{i,j}}$. 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^{2t}_i}. For each $i \in I$, for each $j \in J \setminus j_i$, each $s \in C_{i,j}$ computes $r_{i,j} \equiv M_v^{a^{2t,sim}_{i,j}} \bmod N$. Now, For each $i \in I$, each $s \in C_{i,j^{2t}_i}$ computes (with the help of SIM)

$r_{i,j_i^{2t}} \equiv M_v^d / \prod_{j \in J, j \neq j_i^{2t}} r_{i,j} \bmod N$. 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.

C.0.1 Share Distribution Simulation

The Share Distribution Simulation is similar to the Share Distributions follows, except that the pair (g, g^d) is produced by choosing $\rho \in_R [0, N]$ and letting $g = \rho^e$, and that the shares are produced as follows.

- For each $i \in I$, for each $j \in J \setminus \{r\}$, SIM generates $a_{i,j}^{0,sim} \in_R [-N^2, N^2]$. 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 $b_{i,j}^{0,sim} \equiv \epsilon_{i,j} \equiv g^{a_{i,j}^{0,sim}} \bmod N$. For all $i \in I$, SIM generates $b_{i,j_i^0}^{0,sim} \equiv \epsilon_{i,j_i^0} \equiv g^d / \prod_{j \in J, j \neq j_i^0} g^{a_{i,j}^{0,sim}} \bmod N$.

C.0.2 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) $r_{i,j_i} \equiv M^d / \prod_{j \in J, j \neq j_i} r_{i,j} \bmod N$.

C.0.3 Lost Share Detection Simulation

The servers perform lost share detection exactly as in the real protocol.

C.0.4 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 $\epsilon_{s,i,j_i,i',j_{i'}'} \equiv b_{i,j_i}^{2t,sim} / \prod_{j' \in J \setminus \{j_{i'}'\}} g^{c_{s,i,j_i,i',j'}^{2t,sim}} \bmod N$.
- 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 $s' \in C_{i,j_i}$.

D Security proofs

In this appendix we give the proofs of security for our protocol. First, we reduce the security to a simulator of certain properties, then we prove that the simulator constructed above indeed possesses these properties.

D.1 Main Security Lemma

Lemma D.1 *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* $\mathrm{view}_{\mathcal{A},H}^{\mathrm{simu}(e,N,\mathcal{A},H)}$ *is indistinguishable from* $\mathrm{view}_{\mathcal{A},H}^{\mathrm{real}(e,d,N)}$ *then $S(e,d,N)$ is a (k',k,l)-secure robust proactive RSA system.*

Proof: Assume, 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 (computed non-adaptively), there exists a probabilistic polynomial time simulator $\mathrm{simu}(e,N,\mathcal{A},H)$ such that $\mathrm{view}_{\mathcal{A},H}^{\mathrm{simu}(e,N,\mathcal{A},H)}$ is indistinguishable from $\mathrm{view}_{\mathcal{A},H}^{\mathrm{real}(e,d,N)}$. Any polynomial-time attacker A could not sign a random message $w \in_R \{0,1\}^h$ with probability more than $1/\mathrm{poly}(h)$ given only $\mathrm{view}_{\mathcal{A},H}^{\mathrm{simu}(e,N,\mathcal{A},H)}$ since the combined simulation could be done in polynomial time, and this would contradict the RSA security assumption. However, if the attacker A can sign a random message $w \in_R \{0,1\}^h$ with probability greater than $1/\mathrm{poly}(h)$ when given $\mathrm{view}_{\mathcal{A},H}^{\mathrm{real}(e,d,N)}$, then it can distinguish between $\mathrm{view}_{\mathcal{A},H}^{\mathrm{simu}(e,N,\mathcal{A},H)}$ and $\mathrm{view}_{\mathcal{A},H}^{\mathrm{real}(e,d,N)}$, which contradicts the indistinguishability of the two views. □

D.2 The simulator and indistiguisability 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].

Lemma D.2 *If the adversary* **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: First we note that 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 including encryptions of zero in place of the true encryptions are also statistically indistinguishable.

Now the true simulated protocol using semantically secure encryption is polynomial-time indistinguishable from the zero-encryption simulated secure channel protocol, by the semantically secure encryption assumption, and using a standard hybrid (walking) method. By the same reasoning, the true real protocol is polynomial-time indistinguishable from the zero-encryption real secure channel protocol.

Now assume there is a polynomial-time distinguisher $\mathcal{T}$ between the real and simulated protocols. Let P_r, P_s, $P_{r,z}$, and $P_{s,z}$ be the probability $\mathcal{T}$ outputs one given the view from a real protocol, a simulated protocol, a real protocol with zero-encryption, and a simulated protocol with zero-encryptions. Recall that $\mathrm{poly}(h)$ denotes any polynomial in h. From above, $|P_r - P_{r,z}| < 1/\mathrm{poly}(h)$, $|P_{r,z} - P_{s,z}| < 1/\mathrm{poly}(h)$, and $|P_s - P_{s,z}| < 1/\mathrm{poly}(h)$. Then $|P_r - P_s| < 1/\mathrm{poly}(h)$, and thus the real and simulated protocol are polynomial-time indistinguishable. □

Lemma D.3 *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 $a^{2t}_{i,j_i^{2t}}$. 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 $c^{2t}_{s,i,j,i',j'}$ except for $\{c^{2t}_{s,i,j_i^{2t},i',j_{i'}^{2t+2}}\}$

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

$$diff(Y,Y') = \sum_{v\in\mathcal{D}\cup\mathcal{D}'} |\Pr[Y=v] - \Pr[Y'=v]|.$$

To prove lemma D.3, we need show only that $diff(\mathrm{view}_{\mathcal{A}(L)}^{\mathrm{simu}(e,N,\mathcal{A},L)}, \mathrm{view}_{\mathcal{A}(L)}^{\mathrm{real}(e,d,N,L)})$ is small.

$$\text{Let } X = \prod_{i,j\neq j_i^0} a^0_{i,j} \times \prod_{t,i,j,s\in C_{i,j},i',j':(j,j')\neq(j_i^{2t},j_{i'}^{2t+2})} c^{2t}_{s,i,j,i',j'}$$

$$\text{and let } X^{sim} = \prod_{i,j\neq j_i^0} a^{0,sim}_{i,j} \times \prod_{t,i,j,s\in C_{i,j},i',j':(j,j')\neq(j_i^{2t},j_{i'}^{2t+2})} c^{2t,sim}_{s,i,j,i',j'}.$$

Here multiplication denotes cross products.

Lemma D.4 *$diff(view_{\mathcal{A}(L)}^{\mathrm{simu}(e,N,\mathcal{A},L)}, view_{\mathcal{A}(L)}^{\mathrm{real}(e,d,N,L)}) \leq diff(X, X^{sim})$.*

Proof:

The following parts of the *view* are exactly determined by X (or X^{sim} in the simulation) g, g^d, the history tape, and $\mathcal{A}$'s random bits. They therefore make no contribution to the statistical difference of the views. (Note that g, g^d are random in both the real protocol and the simulation and therefore make no contribution to the statistical difference. The history tape is fixed beforehand; so it makes no contribution either.)

- All values $\{a^{2t}_{i,j}\}_{t>0}$ are determined by $\{c^{2t'}_{s,i,j,i',j'}\}_{2t'<2t:(j,j')\neq(j_i^{2t},j_{i'}^{2t+2})}$ and $\{a^0_{i,j}\}_{j\neq j_i^0}$.

  This is so because:

  (1) $a^0_{i,j_i^0} = d - \sum_{j\neq j_i^0} a^0_j$

  (2) For all t', i, i', $c^{2t'}_{s,i,j_i^{2t},i',j_{i'}^{2t+2}} = a^{2t'}_{i,j_i^{2t}} - \sum_{j'} c^{2t'}_{s,i,j_i^{2t},i',j'}$

  (3) For all t, i', j', $a^{2t+2}_{i',j'} = \sum_j c^{2t}_{s^{2t}_{i,j},i,j,i',j'}$

  where $s^{2t}_{i,j}$ is the server such that $C_{i',j'}$ accepts $s^{2t}_{i,j}$'s c-value as $C_{i',j'}$'s share-of-a-share from the jth share at time $2t$.

  All values $\{a^{2t,sim}_{i,j}\}_{t>0}$ are determined similarly by $\{c^{2t',sim}_{s,i,j,i',j'}\}_{2t'<2t:(j,j')\neq(j_i^{2t},j_{i'}^{2t+2})}$ and $\{a^{0,sim}_{i,j}\}_{j\neq j_i^0}$.

- $\{b_{i,j}^{2t}\}_{t,i,j}$.
  (In the real protocol $b_{i,j}^{2t} \equiv g^{a_{i,j}^{2t}} \bmod N$ for $j \neq j_i^{2t}$, and $b_{i,j_i^{2t}}^{2t} \equiv g^d / \prod_{j \neq j_i^{2t}} g^{a_{i,j}^{2t}} \bmod N$. The same is true for the values $\{b_{i,j}^{2t,sim}\}$.)
- [DISTRIBUTE.1, $N, g, g^d \bmod N$], [DISTRIBUTE.2, $\{\epsilon_{i,j}\}_{i \in I, j \in J}$]
  The first is all given fixed values. The second is computable from X (X^{sim}).
- [SIGN.1, M], [SIGN.2, $s, i, j, r_{i,j}$], [SIGN.CHALLENGE, $s', i, j, a_{i,j}^{2t,sim}$].
  The last one is determined by misbehavior by $\mathcal{A}$ (a function of the preceding part of $\mathcal{A}$'s view and $\mathcal{A}$'s random bits).
- [LOSS.DETECT, $s, \{b_{i,j}^{2t,sim}\}_{i \in I, j \in J}$].
  Obvious from above.
- [RECOVER.ACCUSE, s, i, j, j', s'], [RECOVER.DEFEND, $s', i, j, j', u_{s',i,j,j'}^{2t}$]. The accusations and defenses sent by bad servers are functions of the previous messages and $\mathcal{A}$'s random bits. The accusations and defenses sent by good servers are also functions of the previous messages and $\mathcal{A}$'s random bits.

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

Lemma D.5

$$\mathit{diff}(X, X^{sim}) \leq \frac{2dt_{max}m^3c}{N^2}$$

Proof:
We will need the following claim:

Claim D.1 *For random variables Z, Z', Y, Y' drawn from distributions C, C', D, D', we have*

$$\mathit{diff}((Z,Y),(Z',Y')) \leq \max_{v \in C' \cap D'} \{\mathit{diff}(Z|Y=v, Z'|Y'=v)\} + \mathit{diff}(Y, Y')$$

The proof is by standard methods.

In what follows, we will use the notation $\overline{v}$ to represent an arbitrary integer vector of the appropriate length. These vectors will contain possible previous real and simulated renewal and initial share values.

X and X^{sim} can each be expressed as a cross-product of subviews, one subview of each (family, time step) pair.

Now define:

$$\begin{aligned}
\overline{c^{2t}} &= \prod_{i,j,s \in C_{i,j}, i',j' : (j,j') \neq (j_i^{2t}, j_{i'}^{2t+2})} c_{s,i,j,i',j'}^{2t}, \\
\overline{c^{2t,prev}} &= \left(\prod_{t'<t, i,j,s \in C_{i,j}, i',j' : (j,j') \neq (j_i^{2t'}, j_{i'}^{2t'+2})} c_{s,i,j,i',j'}^{2t'} \times \prod_{i,j \neq j_i^0} a_{i,j}^0 \right), \\
\overline{c^{2t,sim}} &= \prod_{i,j,s \in C_{i,j}, i',j' : (j,j') \neq (j_i^{2t}, j_{i'}^{2t+2})} c_{s,i,j,i',j'}^{2t,sim}, \text{ and} \\
\overline{c^{2t,prev,sim}} &= \left(\prod_{t'<t, i,j,s \in C_{i,j}, i',j' : (j,j') \neq (j_i^{2t'}, j_{i'}^{2t'+2})} c_{s,i,j,i',j'}^{2t',sim} \times \prod_{i,j \neq j_i^0} a_{i,j}^{0,sim} \right).
\end{aligned}$$

Lemma D.6

$$\mathit{diff}(X, X^{sim}) \leq \sum_t \max_{\overline{v^{2t}}}\{\mathit{diff}(\overline{c^{2t}}|\overline{c^{2t,prev}} = \overline{v^{2t}}, \overline{c^{2t,sim}}|\overline{c^{2t,prev,sim}} = \overline{v^{2t}})\} + \mathit{diff}(\prod_{i,j \neq j_i^0} a_{i,j}^0, \prod_{i,j \neq j_i^0} a_{i,j}^{0,sim}).$$

Proof: Follows from Claim D.1. □

Lemma D.7

$$\mathit{diff}(\prod_{i,j \neq j_i^0} a_{i,j}^0, \prod_{i,j \neq j_i^0} a_{i,j}^{0,sim}) \leq \frac{dm}{N^2}$$

Proof:

For all i, $\mathit{diff}(\{a_{i,j}^0\}_{j \neq r, j_i^0}, \{a_{i,j}^{0,sim}\}_{j \neq r, j_i^0}) = 0$ because these variables are chosen independently and uniformly from $\{-N^2, N^2\}$.

If $j_i^0 = r$, we are done. Otherwise we only need to bound (for all i),

$$\mathit{diff}(a_{i,r}^0 | \{a_{i,j}^0\}_{j \neq r, j_i^0} = \overline{v}, a_{i,r}^{0,sim} | \{a_{i,j}^{0,sim}\}_{j \neq r, j_i^0} = \overline{v}).$$

For all i,

$$\begin{aligned} a_{i,r}^0 &= d - \sum_{j \neq j_i^0, r} a_{i,j}^0 - a_{i,j_i^0}^0 = \mathcal{C}_0 + d - a_{i,j_i^0}^0, \text{ and} \\ a_{i,r}^{0,sim} &= -\sum_{j \neq j_i^0, r} a_{i,j}^{0,sim} - a_{i,j_i^0}^{0,sim} = \mathcal{C}_0 - a_{i,j_i^0}^{0,sim}, \end{aligned}$$

where $a_{i,j_i^0}^0, a_{i,j_i^0}^{0,sim}$ are chosen independently, randomly in $\{-N^2, N^2\}$. Thus

$$\mathit{diff}(a_{i,r}^0 | \{a_{i,j}^0\}_{j \neq r, j_i^0} = \overline{v}, a_{i,r}^{0,sim} | \{a_{i,j}^{0,sim}\}_{j \neq r, j_i^0} = \overline{v}) \leq \frac{d}{N^2}$$

Furthermore, the families of initial shares are independent and therefore, the differences between them can be summed.

□

Lemma D.8 *For all t,*

$$\mathit{diff}(\overline{c^{2t}}|\overline{c^{2t,prev}} = \overline{v}, \overline{c^{2t,sim}}|\overline{c^{2t,prev,sim}} = \overline{v}) \leq \frac{2dm^2c}{N^2}$$

Proof:

First, we define an ordering on the indices: we say that $(t, s, i, j, i', j') > (t_1, s_1, i_1, j_1, i_2, j_2)$ if (t, i, j, i', j') is lexicographically greater than $(t_1, i_1, j_1, i_2, j_2)$.

Now define:

$$\overline{c_{s,i,j,i',j'}^{2t,prev}} = \left(\prod_{(t',s',i_1,j_1,i_2,j_2)<(t,s,i,j,i',j') s' \in C_{i_1,j_1}, (j_1,j_2) \neq (j_{i_1}^{2t'}, j_{i_2}^{2t'+2})} c_{s,i,j,i',j'}^{2t'} \right) \times \prod_{i,j \neq j_i^0} a_{i,j}^0$$

$$\overline{c_{s,i,j,i',j'}^{2t,prev,sim}} = \left(\prod_{(t',s',i_1,j_1,i_2,j_2)<(t,s,i,j,i',j') s' \in C_{i_1,j_1}, (j_1,j_2) \neq (j_{i_1}^{2t'}, j_{i_2}^{2t'+2})} c_{s,i,j,i',j'}^{2t',sim} \right) \times \prod_{i,j \neq j_i^0} a_{i,j}^{0,sim}$$

- For bad servers s, we have: $\forall i, j, i', j' : s \in C^{2t}_{i,j}$,

$$diff\Big(\Big[c^{2t}_{s,i,j,i',j'} | \overline{c^{2t,prev}_{s,i,j,i',j'}} = \overline{v}\Big], \Big[c^{2t,sim}_{s,i,j,i',j'} | \overline{c^{2t,prev,sim}_{s,i,j,i',j'}} = \overline{v}\Big]\Big) = 0$$

  This is because the c^{2t} ($c^{2t,sim}$) values are determined by the rest of the real (simulated) view and $\mathcal{A}$'s random bits.

- For good servers s such that $s \in C^{2t}_{i,j} : j \in Bad^{2t}_i$, we have:

$$diff\Big(\Big[c^{2t}_{s,i,j,i',j'} | \overline{c^{2t,prev}_{s,i,j,i',j'}} = \overline{v}\Big], \Big[c^{2t,sim}_{s,i,j,i',j'} | \overline{c^{2t,prev,sim}_{s,i,j,i',j'}} = \overline{v}\Big]\Big) = 0$$

  This is so because $\mathcal{A}$ knows $a^{2t}_{i,j}$.

- For the kth good server $s_k \in C^{2t}_{i,j^{2t}_i}$, we have:

$$diff\left(\left[c^{2t}_{s_k,i,j^{2t}_i,i',j':j'\neq r,j'\neq j^{2t}_{i'}} | \overline{c^{2t,prev}_{s,i,j,i',j'}} = \overline{v}\right], \left[c^{2t,sim}_{s_k,i,j^{2t}_i,i',j':j'\neq r,,j'\neq j^{2t}_{i'}} | \overline{c^{2t,prev,sim}_{s,i,j,i',j'}} = \overline{v}\right]\right) = 0$$

  because values $c^{2t}_{s,i,j,i',j'}$, $c^{2t,sim}_{s,i,j,i',j'}$, $j' \neq r$ are chosen uniformly and independently in $\{-N^2, N^2\}$.

- For the kth good server $s_k \in C^{2t}_{i,j^{2t}_i}$, such that $j^{2t+2}_{i'} \neq r$, we have:

$$diff\left(\left[c^{2t}_{s_k,i,j^{2t}_i,i',r} | \overline{c^{2t,prev}_{s_k,i,j^{2t}_i,i',r}} = \overline{v}\right], \left[c^{2t,sim}_{s_k,i,j^{2t}_i,i',r} | \overline{c^{2t,prev,sim}_{s_k,i,j^{2t}_i,i',r}} = \overline{v}\right]\right) \leq \frac{d}{N^2}$$

  because

$$\begin{aligned}
c^{2t}_{s_k,i,j^{2t}_i,i',r} &= a^{2t}_{i,j^{2t}_i} - \sum_{j'\neq r} c^{2t}_{s_k,i,j,i',j'} \\
&= a^{2t}_{i,j^{2t}_i} - \sum_{j'\notin\{r,j^{2t+2}_{i'}\}} c^{2t}_{s_k,i,j,i',j'} - c^{2t}_{s_k,i,j,i',j^{2t+2}_{i'}} \\
&= \mathcal{C}_0 - c^{2t}_{s_k,i,j,i',j^{2t+2}_{i'}}, \text{ and} \\
c^{2t,sim}_{s_k,i,j^{2t}_i,i',r} &= a^{2t,sim}_{i,j^{2t}_i} - \sum_{j'\neq r} c^{2t,sim}_{s_k,i,j,i',j'} \\
&= a^{2t}_{i,j^{2t}_i} - d - \sum_{j'\notin\{r,j^{2t+2}_{i'}\}} c^{2t,sim}_{s_k,i,j,i',j'} - c^{2t,sim}_{s_k,i,j,i',j^{2t+2}_{i'}} \\
&= \mathcal{C}_0 - d - c^{2t,sim}_{s_k,i,j,i',j^{2t+2}_{i'}},
\end{aligned}$$

  where $\mathcal{C}_0 = a^{2t}_{i,j^{2t}_i} - \sum_{j'\notin\{r,j^{2t+2}_{i'}\}} c^{2t}_{s_k,i,j,i',j'} = a^{2t,sim}_{i,j^{2t}_i} - \sum_{j'\notin\{r,j^{2t+2}_{i'}\}} c^{2t,sim}_{s_k,i,j,i',j'}$, and $c^{2t,sim}_{s^{2t}_i,i,j,i',j'}$ and $c^{2t}_{s^{2t}_i,i,j,i',j'}$ are independently and uniformly distributed in $\{-N^2, N^2\}$.

□

Lemma D.6, Lemma D.7, and Lemma D.8 combined prove Lemma D.5. □ □

E Practical schemes

The complexity of the scheme is the complexity of RSA times the size of the assignment. This size is a small degree polynomial overhead (a function of k'– the density of bad servers). In practice however, for a small (constant) numbers of servers, we can even do much better. We can first replace, in a modular fashion, the probabilistic assignment by a specifically designed assignment. What we observed is that there will often be many possible choices of assignments, and they will generally exhibit tradeoffs in factors such as memory requirements (how many shares each server must hold), communication requirements (what is the number of shares of shares that must be transmitted in the update protocol), and computation requirements (how many multiplications must be done by the gateway in the signature protocol). Note that there may be optimizations possible in the update protocol to reduce the number of messages. In particular, it is enough to have one good family (rather than majority) since it will be easy to detect (using RSA public encryption) which is the correct result among the constant number of computed ones.

We show some tradeoffs in the table below.

Assignment	Security	Communication	Computation	Storage
		(Shares trnsmt./update)	(Mults/signature)	(Shares/server)
A	(1,2,3)-secure	36	2	2
B	(1,2,3)-secure	18	3	2
C	(1,3,4)-secure	8	2	1
D	(2,3,4)-secure	48	3	2
E	(2,3,5)-secure	200	5	4
F	(2,4,6)-secure	72	3	2

The explicit assignments are given below, with rows corresponding to families, and columns corresponding to committees. It should be relatively straightforward to verify that they satisfy the desired properties that (1) for every family, at least one committee in that family contains all good servers, and (2) for at least one family, every committee contains a good server.

A		B			C		D			E					F		
1	2	1,2	1,3	2,3	1,2	3,4	1,2	3	4	1,2	1,3	2,4	3,5	4,5	1,2	3,4	5,6
1	3						1	2	3,4	1,4	1,5	3,4	2,5	2,3	2,3	4,5	6,1
2	3																

Note that to achieve an $(x, x+1, l)$-secure system, one can simply make $\binom{l}{x+1}$ families, each with one of the possible subsets of $x+1$ out of l servers. Scheme A is an example of this. Schemes like these provide very efficient signature operations by the combiner. The practical combiner can check each family sequentially using $x+1$ modular multiplications and applying the RSA public encryption function. Only upon failure does it continue to consider the next family. All our schemes above have a small constant factor slow-down in the overall RSA operation. Such a slow down is justified in sensitive applications that require proactive security.

F Discussion of Further Issues

In general, we assume that good servers in our system are not "curious", meaning that they would not use information obtained from bad servers. For instance, in a $(2,3,5)$-secure system above, if 2 bad servers revealed their shares, then one curious good server could compute the secret key. Note that this is unavoidable in any distributed function-sharing system. Our system does not hinder or promote the revelation of shares known by bad servers. If a bad server attempts to break the system (e.g., by sending an invalid partial signature), the only outcome is that it is discovered and its share is revealed. The bad server could effect this outcome anyway.

More generally, if bad servers in *any* distributed function-sharing system reveal their shares, then the *policy* of the system will be destroyed, since, even if no single "curious" good server can compute the secret key, fewer good servers are required to compute the function. Fortunately, in our system, we can allay this problem by simply performing an update operation whenever a key is revealed (after the appropriate bad servers are rebooted). This actually makes our system more robust against this attack (which we call a *policy attack*).

Also note that given a (k', k, l)-robust-secure proactive RSA system, we automatically have a $(k' - 1, k, l)$-robust-secure proactive RSA system which is resilient to "curious" good servers computing the secret key.