Adaptive Security for the Additive-Sharing Based Proactive RSA

Yair Frankel[1], Philip D. MacKenzie[2], and Moti Yung[3]

[1] Ecash Technologies Inc., USA. yfrankel@cs.columbia.edu
[2] Bell Laboratories, Murray Hill, NJ 07974, USA. philmac@research.bell-labs.com
[3] CertCo Inc., New York, NY, USA. moti@cs.columbia.edu

Abstract. Adaptive security has recently been a very active area of research. In this paper we consider how to achieve adaptive security in the additive-sharing based proactive RSA protocol (from Crypto97). This protocol is the most efficient proactive RSA protocol for a constant number of shareholders, yet it is scalable, i.e., it provides reasonable asymptotic efficiency given certain constraints on the corruption threshold. It is based on organizing the shareholders in a certain design (randomly generated, in the asymptotic case) of families of committees and establishing communications based on this organization. This structure is very different than polynomial-based proactive RSA protocols, and the techniques for achieving adaptive security for those protocols do not apply. Therefore, we develop new techniques for achieving adaptive security in the additive-sharing based proactive RSA protocol, and we present complete proofs of security.

1 Introduction

Distributed cryptosystems provide for security and availability of a private key through distribution of shares of the key, coupled with a mechanism in which some number (a threshold) of shareholders are able to jointly compute the cryptographic function using their shares of the key, while a smaller number of possibly misbehaving parties are prevented from disrupting this computation, or computing the cryptographic function on their own. A distributed cryptosystem achieves *proactive security* if at least a threshold of shareholders must misbehave (or be compromised) *in a specified time period* in order to compromise the security or availability of the system. A distributed cryptosystem achieves *adaptive security* if it is secure against an adaptive adversary, that is, an adversary that may take actions based on the entire available history, including ciphertexts sent during the protocol. (We say such distributed cryptosystems are *adaptively-secure*.) More complicated protocols are typically required to achieve adaptive security, and proving that a protocol is adaptively secure is typically more difficult than proving that a protocol is secure against a non-adaptive adversary.

In this paper we consider the additive-sharing based proactive RSA protocol [FGMY97]. This is the most efficient proactive RSA protocol for a constant number of shareholders, and is also scalable. Unlike some solutions, it guarantees

K. Kim (Ed.): PKC 2001, LNCS 1992, pp. 240–263, 2001.

that the threshold value is maintained throughout, even when some participants are unavailable. Therefore it is an attractive solution for practical situations. It is based on organizing the shareholders in a "design" consisting of families of committees of shareholders. We modify this protocol to achieve adaptive security while maintaining the same organization and the basic protocol structure.

History of Threshold Proactive and Adaptive Systems:

Threshold schemes for discrete log and RSA based cryptosystems were presented in [DF89,B88,F89,DF91,DDFY92,FD92]. Robust schemes that assure the correctness and timeliness of the cryptographic operation were presented in [GJKR,FGY96,GJKR96,Sh00]. Some of these protocols make additional system requirements or additional constraints on certain cryptographic parameters beyond what is required for the corresponding non-distributed cryptosystems. This is not a criticism, since such constraints sometimes allow a cleaner or more efficient distributed solution. For example, to allow an efficient non-interactive robustness check in [GJKR96], secure keys are required at the combiner. Without this, every participant must have keys to verify the operation of other participant which makes the scheme much more expensive (i.e., quadratic in the number of participants). The solution of [GJKR96], and the algebraically-pleasing solution of [Sh00], also place constraints on the RSA keys which can be used. (We note that RSA keys constrained in such a manner cannot be generated using the distributed key generation methods in [BF97,FMY98].) Similar constraints are made in the early heuristic protocol of [DF91] (which [FMY00] builds on).

The notion of proactive security [OY91,CH94,HJKY95,HJJKY96] was developed to cope with a mobile adversary, i.e., an adversary that may compromise all servers over the lifetime of a system, but fewer than a given threshold at any particular time. An RSA based proactive scheme (which does not achieve optimal resilience in the asymptotic case) was given in [FGMY97]. It is based on additive sharing and is the one we concentrate on herein. A scheme based on polynomial sharing that achieves optimal resilience was given in [FGMY97b]. A scheme that combined additive and polynomial sharing was offered by [R98], but it has the sometimes undesirable property that the threshold value may be dynamically reduced due to occasional unavailability of possibly honest shareholders.

The importance of developing protocols that are provably secure against a fully-adaptive adversary has been discussed in [BeHa92,CFGN96,B97] in the context of general distributed computation. Adaptively secure (polynomial sharing) threshold schemes were given in [FMY99,FMY99b], and in [CGJKR], which is extended in [JL00]. The RSA based scheme in [CGJKR] seems to be an n-out-of-n system, since all parties need to compute together after each signature.

Our Contribution:

We present an adaptively secure version of the original additive-sharing based proactive RSA. We present a complete careful proof of its security and robustness without making any additional system requirements or any constraints on the cryptographic parameters. Our version exploits the organization and protocol structure of the original protocol. It is only slightly less efficient, yet it is provably secure against a fully-adaptive adversary. The scheme is very efficient for small

size system, yet it is scalable asymptotically to tolerate any fraction of malicious shareholders bounded away from half (e.g. 1/3). It can be used with general RSA keys and can be initiated distributedly.

Discussion of Techniques Used:

We distribute the RSA private key using many r-out-of-r additive threshold schemes as in [F89]. The additive shares are distributed to certain subsets of servers where the construction of subsets is taken from [AGY95]. By using a simple additive threshold scheme [F89] as the basis of our system, we simplify the domain over which sharing is done (when compared with [DDFY92]), and enable the verification and re-randomization of shares by the servers. Also, this method is what makes the system suitable for settings with a constant number of servers. To provide robustness, we 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. Finally to show that the distribution of shares is secure, we use a simulatability argument similar to one that was put forth in the static distribution of RSA [DDFY92].

For proving security against a fully-adaptive adversary, we must show that the protocol is simulatable. A problem arises if the simulator must simulate the actions of a servers whose shares are unknown, and those actions may commit to the share value. Then if the adversary decides to corrupt that server, it can distinguish between the real protocol and the simulation. In our proactive protocol we use the erasure primitive to maintain security, and we assume that uncorrupted shareholders will erase information when specified by the protocol. In fact, this assumption is required for the system to be secure against a mobile adversary. (In other words, in this security setting, the erasure primitive is as important as any other computational primitive.) Thus, we can substitute non-committing encryption [BeHa92] (which uses erasure) for probabilistic encryption to remove the commitments from the encryptions.

Allowing erasures may seem to be enough to prove security. However, it is not sufficient, since there are other share commitments in the protocol, namely, the partial signatures combined for signing and the witness-based cryptographic program checking used for *robustness*. This is all publicly available information that must be made accessible to any arbitrary party (a gateway) in order for that party to be able to produce the correct function result from the partial results. We thus require new ideas which are developed here. We use information-theoretically secure witnesses for robustness maintenance, which allows a simulator to "spread the fraud" over all servers, instead of concentrating it on a given shareholder with a single "bogus commitment." Then we show how to remove the share commitments from signatures through blinding. In this case, we still have bogus commitments, but we show that the probability of the simulator successfully fooling the adversary is high enough that (limited) backtracking can be employed to generate realistic-looking views for the adversary.

Other then the above changes the adaptively-secure protocol is compatible with the original Crypto97 protocol. (This enables a system designer to imple-

ment both and decide at configuration/run time which variant is applicable to his setting).

2 Model and Definitions

2.1 The RSA Function

First we define the RSA function.

Definition 1. *Let η be the security parameter. Let key generator GE define a family of RSA functions to be $(e, d, N) \leftarrow GE(1^\eta)$ such that N is a composite number $N = P * Q$ where P, Q are prime numbers of $\eta/2$ bits each. The exponent e and modulus 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 RSA is given in Definition 6.*

For naming convenience, we will assume our system is used for direct RSA signing of messages; however, the same protocol could be used for decryption. Our results simply concern the application of the RSA function in its assumed intractable direction as a one-way function (as assumed in protocols with formal security proofs which employ RSA, e.g. [ACGS]).

2.2 The Model

Now we define our Proactive RSA system.

The Participants: The participants in our system are (1) l servers $\{s_1, \ldots, s_l\}$, (2) a trusted dealer D, and (3) a gateway G. The dealer D generates an RSA instance (e, d, N), distributes shares of d among the l servers during an initialization phase, and then does not participate any further. The gateway G is used to broadcast messages to be signed to the servers and to combine their partial signatures into the final signature. (Note that G is not trusted, and in fact is not really necessary except for convenience.)

As the protocol is running, the servers will be classified as follows: A server is *compromised* if its secret is known to the adversary. A server is *corrupted* if it is controlled by the adversary. (We assume all corrupted servers are compromised.) We refer to a server as *good* if it is uncorrupted, and actively participates in a protocol. We refer to a server as *bad* if it is compromised. (Note that some servers may be neither good nor bad.) We assume

[1] $\lambda(N) = \mathrm{lcm}(P-1, Q-1)$ 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 it is easy to see that $\lambda(N)$ can be used instead. We use it because it gives an explicit way to describe an element of maximal order in Z_N^*. Note that $\phi(N)$ is a multiple of $\lambda(N)$, and that knowing any value which is a multiple of $\lambda(N)$ implies breaking the system.

that all uncorrupted servers receive all messages that are broadcast, and retrieve the information from messages encrypted with a public key, if they know the corresponding private key. We also assume that if they are active in a protocol, they participate honestly. We will show that our protocols perform correctly even if only the good servers and corrupted servers are active.

The Communication Model: The communication model presented here is similar to [HJKY95]. All participants communicate via an authenticated bulletin board [CF85] in a synchronized manner. 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 an operational period, the servers can cooperatively sign messages using the current key shares. 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. (Technical note: we consider a server that is corrupted during an update phase as being corrupted during both its adjacent periods. This is because the adversary could learn the shares used in both the adjacent operational periods.)

System Management: We assume that a server that is determined to be corrupted by a majority of active servers can be *refreshed* (i.e., erased and rebooted, or perhaps replaced by a new server with a blank memory) by some underlying system management. This is a necessary assumption for dealing with corrupted servers in any proactive system.

The Adversarial Model: We next define some orthogonal properties of the adversary.

Definition 2. *A* **stationary** *adversary may corrupt servers at different times, but a server that is corrupted remains corrupted for the duration of the protocol. A* **mobile** *adversary (we employ) may corrupt servers at different times, but a server that is corrupted may be refreshed by the underlying system management at some later time, and become "uncorrupted."*

Note that when we deal with a mobile adversary, the set of corrupted servers does not necessarily monotonically increase over time. Also note that the above definitions also apply to compromising or preventing honest participation by servers.

Definition 3. *A* (k', k, l)**-restricted** *adversary is a mobile adversary that can, during any time period, (a) prevent at most* $l - k$ *servers from honestly participating in a protocol (i.e., there will always be at least* k *good servers), (b) corrupt (i.e., force to behave in an arbitrary manner) at most* $\min\{l - k, k'\}$ *servers, and*

(c) compromise (i.e., view the memory of) at most k' servers (including those that it corrupts).

Threshold schemes are often described in terms of withstanding coalitions of "k-out-of-l" corrupted servers. In our notation, that would correspond to withstanding a $(k, k+1, l)$-restricted adversary.

The actions of an adversary at any time may include submitting messages to the system to be signed, corrupting or compromising servers, and broadcasting arbitrary information on the communication channel. The following definition describes different criteria on which an adversary may base its actions.

Definition 4. *A* **static** *adversary must decide on its actions before the execution of the protocol. A* **non-adaptive** *adversary (considered earlier) is allowed to base its actions on previous function outputs (i.e., previous message/signature pairs) obtained during the execution of the protocol, but no other information gained during the execution of the protocol. A* **fully adaptive** *adversary (which we employ) is allowed to base its actions not only on previous function outputs, but on all the information that it has previously obtained during the execution of the protocol, including all messages broadcast on the public channel (ciphertexts exchanged between servers, partial function evaluations, etc.) and the contents of all memories of servers that it has viewed.*

We assume that the adversary stores all messages that were broadcast on the public channel and the contents of all memories of servers that it can view. (Naturally, all this information is time-tagged.) Assuming S is the system being attacked, and $\mathcal{A}$ is the adversary, we call this stored information $view^S_{\mathcal{A},L}$, where L is the list of messages that are submitted to S to be signed. (Note that the signatures will be implicit in the view.)

system, etc.)

2.3 Properties

We say that a function $f(h)$ is *negligible* if for any polynomial $\text{poly}(\cdot)$, there is an h' such that for $h > h'$, $f(h) < 1/\text{poly}(h)$.

Now we define what it means for a system to be robust. In these definitions we assume the security parameter η is large enough so that the analysis holds.

Definition 5. (Robustness) *Let η be the security parameter. A system S is a (k', k, l)-robust proactive RSA system if it contains a polynomial-time protocol SIG (run by the servers) such that for any probabilistic polynomial-time (k', k, l)-restricted adversary $\mathcal{A}$, with all but negligible probability, assuming (e, d, N) is the RSA instance generated by the dealer, for every $\alpha \in [0, N]$ that is submitted to be signed (during an operational period), the servers can compute $\alpha^d \bmod N$ using SIG.*

Next we define what it means for the RSA system to be secure (RSA is treated as a one-way function).

Definition 6. (Security) *Let η be the security parameter. Let key generator GE define a family of RSA functions (i.e., $(e,d,N) \leftarrow GE(1^\eta)$ is an RSA instance with security parameter η). Let $S(e,d,N)$ denote a system S in which the RSA instance (e,d,N) was chosen by the trusted dealer D. Then S is a (k',k,l)-secure proactive RSA system if it generates instances of RSA using $GE(1^\eta)$, and if for any probabilistic polynomial-time (k',k,l)-restricted adversary $\mathcal{A}$, for any list L of randomly chosen messages submitted to S to be signed, for any probabilistic polynomial-time function A: $\Pr[u^e \equiv w \bmod N : (e,d,N) \leftarrow GE(1^\eta); w \in_R \{0,1\}^\eta; u \leftarrow A(1^\eta, w, view_{\mathcal{A},L}^{S(e,d,N)})]$ is negligible.*

In the next section we describe a proactive RSA system which is (k',k,l)-secure and (k',k,l)-robust, assuming a fully adaptive adversary.

3 The Adaptively-Secure Protocol

3.1 Outline

At the start an assignment of servers to committees and families is chosen (as in [AGY95]). Specifically, for given values m, r, and c, servers are randomly assigned to mr committees that are evenly split between m families, such that each committee has c servers assigned to it. (Note that each server may be assigned to multiple committees.) This random assignment may be performed by the trusted dealer, or by the servers themselves. The values m, r, and c are chosen such that with high probability each set of k servers can obtain all the shares from some family, and no set of k' servers can obtain all the shares from any family.

After the assignment of servers to committees, the dealer D generates an RSA instance (e,d,N), and distributes additive shares of d to the committees in each family as in [F89]. Thus each family obtains an r-out-of-r sharing of the key. In the original protocol $a_{i,j}^0$ is the share given to the jth committee in family i by the dealer, and more generally, $a_{i,j}^{2t}$ is the share corresponding to the jth committee in family i during operational period $2t$. During an operational period (say period $2t$), when a message M needs to be signed, the servers possessing $a_{i,j}^{2t}$ can produce the value (the partial signature) $M^{a_{i,j}^{2t}} \bmod N$. Because of the assignment choice above, at least one of the m families will produce correct values for all committees, and therefore, for at least one i, the gateway G can compute the RSA signature of M using the formula $M^d \equiv \prod_{j=1}^{r} M^{a_{i,j}^{2t}} \bmod N$.

During an update period (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 shares of shares appropriately.

To provide for robustness throughout the protocol, we use a witnesses (similar to [FGY96,GJKR96]) for each share. These witness are used to check partial signatures and to check that the renewal of shares in the update period is performed correctly.

To allow adaptive adversary we take a few extra steps, to assure simulatability of the protocol. We use information theoretic commitments and witness

process. We carefully evaluate results. But, the protocol organization and structure remains as in the original protocol, which is our main achievement. (We note that sometimes, adapting a protocol may be harder than having the freedom to design it afresh. Also, we do not change the basic model assumptions: threshold requirements, and the notion of threshold as some earlier protocols did).

Remark: In all of these subprotocols, messages are put on an authenticated bulletin board. One implementation is having every message signed by the sender using a secure signature scheme, and ignoring any message with an invalid signature. We include the renewal of individual signature keys in the description of our protocol. Note that we still need an initial authentication token to be available for recovering servers. We will henceforth assume that all the signed messages originate at the correct server; otherwise, the assumption about the security of the underlying signature scheme is violated.

Notation: In our protocol description, messages are presented inside brackets. Messages include a "message-tag" which is a self-explanatory (mnemonic) description of the role of the message, the tag is followed by the message content.

3.2 Initialization Phase

Families and Committees: The assignment we describe is a slight generalization of [AGY95]. This assignment can be performed by the trusted dealer, or by the servers themselves. 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 (k', k, l)-terrific assignment, that is, one that obeys the following properties **for any** set of "bad" servers $B \subseteq S$ with $|B| \leq k'$ and any set of "good" servers $E \subseteq S$ with $|E| \geq k$:

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.)

Let $\sigma = k'/l$ and let $\tau = k/l$. Given l, q, p, and security parameter $\eta \geq \max\{2l + 2, 100\}$, we will set $c = \lceil \{2 \log \eta / \log(\frac{1-\sigma}{1-\tau})\} \rceil$, $r = (1-\tau)^{-c}/\eta$, and $m = 10\eta$. Note that all of these values are polynomial in η as long as $\tau - \sigma$ is greater than some constant (i.e., as long as the gap between the number of good servers and bad servers is at least a constant fraction of l).

Lemma 7 ([AGY95]). *A randomly chosen assignment is ϵ-terrific with overwhelming probability.*

The proof is given in the appendix. We can control the probability of obtaining a non-ϵ-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.

In practical adaptations where a small (constant) numbers of servers is expected the randomized construction can be replaced be a deterministic design. For example:

- a (1,2,3)-terrific assignment can be made with 1 family with 3 committees: ({1,2},{1,3},{2,3});
- a (2,3,4)-terrific assignment can be made with two families, each with 3 committees:
  ({1,2},{3},{4}) and ({1},{2},{3,4});
- a (2,3,5)-terrific assignment can be made with three families, each with 3 committees:
  ({1,2},{3,4},{5}), ({1,3},{2,5},{4}) and ({2,4},{3,5},{1}); and
- a (2,4,6)-terrific assignment can be made with two families, each with 3 committees:
  ({1,2},{3,4},{5,6}) and ({2,3},{4,5},{6,1}).

3.3 Initialization Phase

In this phase, only the Share Distribution protocol is changed from the Basic Protocol. The change is simply the inclusion of companion shares.
Share Distribution 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 $d' \in_R [1, N^2]$.
3. The dealer generates[2] $g, h \in_R [2, N-2]$, sets $y = g^d h^{d'} \bmod N$ (Pedersen commitment to d), and broadcasts [DISTRIBUTE.1, N, e, g, h, y].
4. The dealer next generates the shares of the secret: For each $(i,j) \in I \times J \backslash \{r\}$, the dealer generates $a^0_{i,j} \in_R [-\rho, \rho]$ and $w^0_{i,j} \in_R [-N\rho, N\rho]$. Then it sets $a^0_{i,r} = d - \sum_{j \in J \backslash \{r\}} a^0_{i,j}$ and $w^0_{i,r} = d' - \sum_{j \in J \backslash \{r\}} w^0_{i,j}$.
5. The dealer generates the Pedersen commitment to the shares: For each $i \in I$ and $j \in J$, the dealer sets $\epsilon_{i,j} \equiv g^{a^0_{i,j}} h^{w^0_{i,j}} \bmod N$
6. The dealer broadcasts the private check shares and the encrypted shares: [DISTRIBUTE.2, $\{\epsilon_{i,j}\}_{(i,j) \in I \times J}$, $\{\mathrm{ENC}_{i,j}(a^0_{i,j}, w^0_{i,j})\}_{(i,j) \in I \times J}$)].
7. Every server checks for each family $i \in I$ that $\prod_{j \in J} \epsilon_{i,j} \equiv y \bmod N$ and each server in $C_{i,j}$ checks the correctness of its public share, namely that $\epsilon_{i,j} \equiv g^{a^0_{i,j}} h^{w^0_{i,j}} \bmod N$.
8. Every server sets the signature family indicator $f^0 = 0$, and for each $(i,j) \in I \times J$, every server sets $b^0_{i,j} = \epsilon_{i,j}$.

Recently, a method for generating a distributed RSA key was developed by [BF97]. Later a robust distributed RSA key generation system was developed

[2] As in the Basic Protocol we assume here that g and h are of maximal order.

[FMY98]. We note that, given this distributed generation, we could distributively perform steps 2-7 (with the need to replicate super logarithmic number of random elements g's and generate them and their corresponding h's distributedly). Thus, we could remove the trusted dealer assumption from our proactive RSA model. The initial assignment can also be chosen distributedly via a coin flipping protocol by the shareholders. (A discussion on distributed initialization is omitted from this version).

Now that the system is initialized, we have to describe the the signing protocol and the proactive update (renewal) protocol.

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. For succinctness, let $\rho = Nm^2r^2c\psi(\eta)$, where $\psi()$ is an agreed upon super-polynomial function. This value ρ will be used to compute the possible range of secret shares in each period.

3.4 Signature Protocol (Operational Period $2t$)

This is the protocol to be followed when the gateway obtains a message M to be signed in period $2t$. It differs from the Basic Protocol in that families work sequentially to try to generate a signature, instead of in parallel. Also, in each family there is a re-randomization of the shares (in steps 2(a)-(h) below, which are just like the share renewal protocol but only involving one family).

As in the Basic Protocol, we use the fact that since $d = \sum_{j \in J} a_{i,j}^{2t}$ then $M^d \equiv \prod_{j \in J} M^{a_{i,j}^{2t}} \bmod N$.

During an operational period (say period $2t$), when a message M needs to be signed, only one family will initially attempt to generate a signature. This will be indicated by the *signature family indicator* f^{2t}. Let $i = f^{2t}$. If F_i is unable to generate a signature, then all its data is erased, f^{2t} is incremented, and the next family will attempt to generate a signature. This is repeated until some family succeeds in generating a signature. (Success is guaranteed by the assignment choice, which was shown to include good families.) When a family attempts to generate a signature, it will randomize its shares, and use the randomized shares for computing the partial signatures. The signal family indicator f^{2t} is set to the family that should be attempting to generate a signature. There are two possible cases: (1) the family has already signed at this stage, and thus the randomized shares have been produced already and the current signature session can reuse them, or (2) the family has to start by re-randomizing the shares.

The protocol proceeds as follows:

1. The gateway broadcasts $[\mathrm{SIGN}.1, M]$.
2. Let $i = f^{2t}$. The following steps are performed for family F_i.

(a) If the period's randomized shares $\tilde{a}^{2t}_{i,j}$, $\tilde{w}^{2t}_{i,j}$, and $\tilde{b}^{2t}_{i,j}$ values have already been produced, skip to Step 3.
Otherwise, run share-randomization: the members of the family perform additive re-randomization of shares: for all $j \in J$, each server s in $C_{i,j}$ does the following: s chooses $\tilde{c}^{2t}_{s,i,j,j'} \in_R [-\rho(t+1)^2, \rho(t+1)^2]$ and $\tilde{v}^{2t}_{s,i,j,j'} \in_R [-N\rho(t+1)^2, N\rho(t+1)^2]$ for $j' \in J \setminus \{r\}$. Then it sets $\tilde{c}^{2t}_{s,i,j,r} = a^{2t}_{i,j} - \sum_{j' \in J \setminus \{r\}} \tilde{c}^{2t}_{s,i,j,j'}$ and $\tilde{v}^{2t}_{s,i,j,r} = w^{2t}_{i,j} - \sum_{j' \in J \setminus \{r\}} \tilde{v}^{2t}_{s,i,j,j'}$. Then for all $j' \in J$, s computes $\alpha_{s,i,j,j'} = \mathrm{ENC}_{i,j'}[\tilde{c}^{2t}_{s,i,j,j'}, \tilde{v}^{2t}_{s,i,j,j'}]$ and $\beta^{2t}_{s,i,j,j'} = g^{\tilde{c}^{2t}_{s,i,j,j'}} h^{\tilde{v}^{2t}_{s,i,j,j'}} \bmod N$.

(b) **Share-of-share distribution**: Each server in a committee redistributes the committee's share amongst the other committees. One of the committee's correct resharings will be used. For all $j \in J$, each server s in $C_{i,j}$ broadcasts
$[\mathrm{SIGN.RANDOMIZE}.1, s, i, j, \{\beta^{2t}_{s,i,j,j'}\}_{j' \in J}, \{\alpha_{s,i,j,j'}\}_{j' \in J}]$.

(c) **Verification of shares of check shares**: Every server verifies, for all $j \in J$ and all $s \in C_{i,j}$, that $\prod_{j' \in J} \beta^{2t}_{s,i,j,j'} = b^{2t}_{i,j} \bmod N$, and informs the system management if it doesn't hold for some s (The servers can easily agree that this s is faulty). From this point on, we only deal with messages from those s where the above equality does hold.

(d) **Verification of shares of shares**: For all $j' \in J$, each $s \in C_{i,j'}$ decrypts shares to $C_{i,j'}$ and verifies $g^{\tilde{c}^{2t}_{s',i,j,j'}} h^{\tilde{v}^{2t}_{s',i,j,j'}} \equiv \beta^{2t}_{s',i,j,i',j'} \bmod N$ for all s'. For all $j' \in J \setminus \{r\}$, each $s \in C_{i,j'}$ also verifies $|\tilde{c}^{2t}_{s',i,j,j'}| \leq \rho(t+1)^2$ and $|\tilde{v}^{2t}_{s',i,j,j'}| \leq N\rho(t+1)^2$ for all s'.

(e) **Dispute resolution**: If server s finds that verification fails for a message from server s', s broadcasts
$[\mathrm{SIGN.ACCUSE}, s, i, j, j', s']$, to which s' responds by broadcasting
$[\mathrm{SIGN.DEFEND}, s', i, j, j', \tilde{c}^{2t}_{s',i,j,j'}, \tilde{v}^{2t}_{s',i,j,j'}]$.

(f) **Agreement on bad servers**: All servers check all accusations and inform the system management of any bad servers (i.e., those that defended with invalid values of $\tilde{c}^{2t}_{s',i,j,j'}$ and $\tilde{v}^{2t}_{s',i,j,j'}$). Again, from this point on, we only deal with messages from the good servers.

(g) **Reconstructing the family's randomized shares for signing**: For all $j' \in J$, each $s \in C_{i,j'}$, using the lexicographically first servers in each committee with shares that passed verification (call them $s_{i,j}$), computes $\tilde{a}^{2t}_{i,j'} = \sum_{j \in J} \tilde{c}^{2t}_{s_{i,j},i,j,j'}$, $\tilde{w}^{2t}_{i,j'} = \sum_{j \in J} \tilde{v}^{2t}_{s_{i,j},i,j,j'}$, and $\tilde{b}^{2t}_{i,j'} \equiv \prod_{j \in J} \beta^{2t}_{s_{i,j},i,j,j'} \bmod N$.

(h) Everything produced at previous steps in this protocol is erased except $\tilde{a}^{2t}_{i,j}$, $\tilde{w}^{2t}_{i,j}$, and $\tilde{b}^{2t}_{i,j}$ for all $j \in J$.

3. Now the family has re-randomized shares for signing at this period. They follow the procedure below:

(a) **Generating partial signatures based on randomized shares**: For all $j \in J$, each server $s \in C_{i,j}$ computes $r_{i,j} \equiv M^{\tilde{a}^{2t}_{i,j}} \bmod N$ and broadcasts the message $[\mathrm{SIGN}.2, s, i, j, M, r_{i,j}]$.

(b) **Disputes within committees**: For all $j \in J$, 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 $r_{i,j} \not\equiv M^{\tilde{a}^{2t}_{i,j}} \bmod N$, then s' broadcasts the challenge $[\text{SIGN.CHALLENGE}, s', i, j, \tilde{a}^{2t}_{i,j}, \tilde{w}^{2t}_{i,j}]$.

(c) **Agreement on bad servers in committees**: All servers verify all challenges (by checking if $\tilde{b}^{2t}_{i,j} \equiv g^{\tilde{a}^{2t}_{i,j}} h^{\tilde{a}^{2t}_{i,j}} \bmod N$) and inform the system management of any bad servers (i.e., those servers that sent a message with $r_{i,j} \not\equiv M^{\tilde{a}^{2t}_{i,j}} \bmod N$).

(d) **Combining the partial signatures and verifying correctness of the final signature**: After the checks are completed, and for the $r_{i,j}$ values that passed the tests, the gateway tests if $(\prod_{j \in J} r_{i,j})^e \equiv M \bmod N$. If this does not hold, then for all $j \in J$, all servers in $C_{i,j}$ erase their shares $a^{2t}_{i,j}$ and $w^{2t}_{i,j}$ and anything else produced at previous steps in this protocol, f^{2t} is incremented, and we go to Step 2 (thus proceeding to the next family). Note that if a signature is not computed, family F_i must already be bad so erasing the remaining shares does not affect the system. Note also that there are good families, and such a family will produce a valid signature.

3.5 Renewal Protocols (Update Period $2t + 1$)

Only the Share Renewal/Lost Share Recovery protocol changes from the Basic Protocol, whereas individual key renewal and committee key renewal do not. If the non-committing encryption scheme from [BeHa92] is used, the key renewal phases include generation of pseudorandom pads, and the actual keys are erased. We refer the reader to [BeHa92] for details.

Share Renewal/Lost Share Recovery: The only change from the Basic Protocol is to perform renewal on the $w^{2t}_{i,j}$ shares (the companion shares) along with the $a^{2t}_{i,j}$ shares.

Now we give the protocol:

1. **Share-resharing**: For all $(i, j, i') \in I \times J \times I$, each server s in $C_{i,j}$ does the following: s chooses $c^{2t}_{s,i,j,i',j'} \in_R [-\rho(t+1)^2, \rho(t+1)^2]$ and $v^{2t}_{s,i,j,i',j'} \in_R [-N\rho(t+1)^2, N\rho(t+1)^2]$ for $j' \in J \setminus \{r\}$. Then it sets $c^{2t}_{s,i,j,i',r} = a^{2t}_{i,j} - \sum_{j' \in J \setminus \{r\}} c^{2t}_{s,i,j,i',j'}$ and $v^{2t}_{s,i,j,i',r} = w^{2t}_{i,j} - \sum_{j' \in J \setminus \{r\}} v^{2t}_{s,i,j,i',j'}$. Then for all $j' \in J$, s computes $e_{s,i,j,i',j'} = \text{ENC}_{i',j'}[c^{2t}_{s,i,j,i',j'}, v^{2t}_{s,i,j,i',j'}]$ and $\epsilon_{s,i,j,i',j'} = g^{c^{2t}_{s,i,j,i',j'}} h^{v^{2t}_{s,i,j,i',j'}} \bmod N$.
2. **Share-of-share distribution**: For all $(i, j) \in I \times J$, each server s in $C_{i,j}$ broadcasts $[\text{UPDATE.1}, s, i, j, \{\epsilon_{s,i,j,i',j'}\}_{(i',j') \in I \times J}, \{e_{s,i,j,i',j'}\}_{(i',j') \in I \times J}]$.
3. **Public verification of of re-shared witnesses against old witnesses**: 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. **Private verification re-shared witnesses**: For all $(i', j') \in I \times J$, each $s \in C_{i',j'}$ decrypts shares to $C_{i',j'}$ and verifies $g^{c^{2t}_{s',i,j,i',j'}} h^{v^{2t}_{s',i,j,i',j'}} \equiv \epsilon_{s',i,j,i',j'} \bmod N$ for all s'. For all $(i', j') \in I \times J \setminus \{r\}$, each $s \in C_{i',j'}$ also verifies $|c^{2t}_{s',i,j,i',j'}| \leq \rho(t+1)^2$ and $|v^{2t}_{s',i,j,i',j'}| \leq N\rho(t+1)^2$ for all s'.
5. **Dispute resolution**: If server s finds that verification fails for a message from server s', s broadcasts $[\text{UPDATE.ACCUSE}, s, i, j, i', j', s']$, to which s' responds by broadcasting $[\text{UPDATE.DEFEND}, s', i, j, i', j', c^{2t}_{s',i,j,i',j'}, v^{2t}_{s',i,j,i',j'}]$.
6. **Agreement on bad servers**: All servers check all accusations and inform the system management of any bad servers (i.e., those that defended with invalid values of $c^{2t}_{s',i,j,i',j'}$ and $v^{2t}_{s',i,j,i',j'}$). Again, from this point on, we only deal with messages from the good servers.
7. **Correct share update**: For all $(i', j') \in I \times J$, each $s \in C_{i',j'}$, 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 $s_{i,j}$), computes $a^{2t+2}_{i',j'} = \sum_{j \in J} c^{2t}_{s_{i,j},i,j,i',j'}$, $w^{2t+2}_{i',j'} = \sum_{j \in J} v^{2t}_{s_{i,j},i,j,i',j'}$, and $b^{2t+2}_{i',j'} \equiv \prod_{j \in J} \epsilon_{s_{i,j},i,j,i',j'} \bmod N$.
8. **Terminate update**: Everything is erased except $a^{2t+2}_{i,j}$, $w^{2t+2}_{i,j}$, and $b^{2t+2}_{i,j}$ for all $(i, j) \in I \times J$.
9. f^{2t+2} is set to 0.

4 Proof of Adaptive Security

We claim that:

Theorem 8. *The proactive RSA system above is (k', k, l)-secure against a fully adaptive adversary.*

We prove the security of our system in two stages. First, we construct a simulator which produces views of the adversary in polynomial time (in Subsection 4.1). Then (in Subsection 4.2) we show how, using the simulator, one reduces the security of the RSA function to the security of a slightly modified version of our scheme against the adaptive mobile adversary. This slightly modified system simply gives more information to the adversary, so it is trivial to conclude the security of our actual system. Also, we actually use a more stringent definition of security in which not only should it be difficult for an adversary to generate a signature for a random message, but it also should be difficult to generate the discrete log of g with respect to h, and vice-versa. More concretely,

Definition 9. (Security) *Let η be the security parameter. Let key generator GE define a family of RSA functions (i.e., $(e, d, N) \leftarrow GE(1^\eta)$ is an RSA instance with security parameter η). Let $S(e, d, N, g, h)$ denote a system S in which the RSA instance (e, d, N) was chosen by the trusted dealer D using GE and*

$g, h \in_R Z_N^*$. Then S is a (k', k, l)-secure proactive RSA system if it generates instances of RSA using $GE(1^\eta)$, and if for any probabilistic polynomial-time (k', k, l)-restricted adversary $\mathcal{A}$, for any list L of randomly chosen messages submitted to S to be signed, for any probabilistic polynomial-time function A: $\Pr[(u^e \equiv w \bmod N) \vee (g^\alpha \equiv h^\beta \bmod N) : (e, d, N) \leftarrow GE(1^\eta); g, h \in_R Z_N^*; w \in_R \{0,1\}^\eta; u, \alpha, \beta \leftarrow A(1^\eta, w, view_{\mathcal{A},L}^{S(e,d,N,g,h)})]$ *is negligible.*

We will use the following lemma, which can be proven using the RSA security assumption.

Lemma 10. *Let η be the security parameter. Let modulus generator GE define a family of modulus generating functions (i.e., $N \leftarrow GE(1^\eta)$ be an RSA modulus with security parameter η). For any probabilistic polynomial-time adversary $\mathcal{A}$,* $\Pr[u^e \equiv w^d \bmod N; (e \neq 0) \vee (d \neq 0) : N \leftarrow GE(1^\eta); u, w \in_R \{0,1\}^\eta; e, d \leftarrow A(1^\eta, w, u)]$ *is negligible.*

Proof. Similar to [B84].

In the slightly modified system we consider, we assume the adversary is given some extra information each time a set of shares is erased. We assume that after the shares are erased for a given family, the adversary is given the values of shares and "randomized signature shares" from all committees in that family except for the highest numbered committee (hereinafter called the *designated committee*) in which no servers were corrupted (up to that point in the period). Additionally, the adversary is given all values used in constructing those shares, and all values produced from those shares, i.e., the c, $\tilde{c}$, v and $\tilde{v}$ values. Finally, for any families that did not generate randomized signature shares, random ones are generated and revealed to the adversary, except for the ones for the designated committee. Thus the adversary always obtains from each family all but one committee's share and "randomized signature share." The choice of this committee is a deterministic function of the adversary's random bits and the adversary's view of the protocol to that point.

4.1 The Simulator

Here we construct SIM to simulate the view of $\mathcal{A}$ in the (secure-channel) system we describe above. We assume the random bits of $\mathcal{A}$ are fixed. The inputs to the simulator are an RSA public key (e, N), generators g and h,[3] the description of the adversary $\mathcal{A}$, and a history H.

Whenever f^{2t} is set to a value i, SIM chooses $j \in_R J$, indicating that SIM is guessing that $C_{i,j}$ is the designated committee in F_i for period $2t$. Note that SIM has exactly a $1/r$ probability of correctly guessing the one designated committee in F_i for period $2t$. If the guess is incorrect, SIM repeatedly backtracks to the point

[3] This is for the more stringent definition of security we want to prove for the Adaptive protocol. The simulator for the basic protocol needed to be able to choose the generator g itself, whereas for the adaptive protocol, it will not actually be necessary to choose these generators. This fact is needed in our proof of robustness.

where f^{2t} is set to i, randomly guessing which committee will be the designated committee, until it guesses correctly. Recall that after backtracking, the view of $\mathcal{A}$ could change, and since $\mathcal{A}$ is fully adaptive, it could corrupt different servers. Thus the designated committee may change. Still, SIM will have exactly a $1/r$ probability of guessing correctly each time it backtracks, and thus there will be no bias in the simulation with respect to the corruptions made by $\mathcal{A}$. There are m families, and the guesses are made sequentially. Therefore, SIM backtracks at most an average of mr times per period.

From now on, we will assume that SIM guesses the designated committee correctly. Below we discuss details of the simulations of the subprotocols. Note that the simulations of the key renewal protocol, the lost share detection protocol, and the share renewal/lost share recovery protocol are all exactly the same as the normal protocols.

Share Distribution Simulation: The Share Distribution Simulation is similar to Share Distribution, except that d is chosen to be 0. We let $a_{i,j}^{2t,sim}$, $w_{i,j}^{2t,sim}$, and $b_{i,j}^{2t,sim}$ denote the simulated values corresponding to the shares with the corresponding names.

Signature Simulation: Simulating a signature of some $M \in L$ during period $2t$ is done as in the Signature protocol with the following exception:

- In step 3, for the family F_i that is currently attempting to sign M and designated committee $C_{i,j}$, each $s \in C_{i,j}$ computes (with SIM's help, using the history H) $r_{i,j} \equiv M^d / \prod_{j' \in J, j' \neq j} r_{i,j'} \bmod N$.

4.2 Security Proofs

The outline of the full security proof is as follows. We will first prove security in the secure channels model. Specifically, in Lemma 11 we will prove that the view of an adversary attacking our simulation of the protocol is statistically indistinguishable from the view of that adversary attacking the real protocol. Then, in Lemma 16, we show that statistical indistinguishability in the secure channels model implies polynomial indistinguishability when using non-committing encryption for sending private messages. Finally, we prove the theorem by showing that breaking the security of the protocol implies breaking RSA.

Simulation over Secure Channels For the rest of this section, let S be the system from the previous section with the modification described at the beginning of this section, and let SIM be the simulator described above.

Lemma 11. *Assuming secure channels, for any probabilistic (k', k, l)-restricted fully adaptive adversary $\mathcal{A}$, $view_{\mathcal{A},L}^{\mathrm{SIM}(e,N,\mathcal{A},H,g,h)}$ is statistically indistinguishable from $view_{\mathcal{A},L}^{S(e,d,N,g,h)}$, where $H = hist(d, N, L)$.*

Proof. To simplify the proof 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. We will assume that the protocol is run through period $2T$.

Say that, for each family F_i, j_i^{2t} is defined so that $C_{i,j_i^{2t}}$ is the designated committee.

For the sake of a consistent representation, let X be the random variable consisting of the cross product of the values from the set

$$\bigcup_{(i,j)\in I\times J}\{a_{i,j}^0, w_{i,j}^0\}\cup\{y\}\cup \bigcup_{t\in\{0\ldots T-1\},(i,j,i',j')\in I\times J\times I\times J}\{c_{i,j,i',j'}^{2t}, v_{i,j,i',j'}^{2t}\} \cup \bigcup_{t\in\{0\ldots T-1\},(i,j,j')\in I\times J\times J}\{\tilde{c}_{i,j,j'}^{2t}, \tilde{v}_{i,j,j'}^{2t}\},$$

with $*$'s in certain positions indicating information the adversary does not see. Let X^{sim} denote the corresponding random variable in the simulated protocol.

Note that $y(= g^d h^{d'} \bmod N)$ along with the other non-$*$ values of a tuple from X or X^{sim} uniquely determine the check shares.

Proposition 12. *diff*$(view_{\mathcal{A},L}^{\mathrm{SIM}(e,N,\mathcal{A},H,g,h)}, view_{\mathcal{A},L}^{S(e,d,N,g,h)}) \leq$ *diff*(X, X^{sim}).

Proof. The remaining parts of the view of $\mathcal{A}$ are exactly determined by X (or X^{sim} in the simulation) g, h, the history tape, and $\mathcal{A}$'s random bits, and the history tape and $\mathcal{A}$'s random bits are assumed fixed. The remaining parts of the view therefore make no contribution to the statistical difference of the views.

We call the cross-products generated by X and X^{sim} *tuples*. Note that they include $*$'s in certain locations to indicate the value is unspecified (unknown to $\mathcal{A}$). Let Z (Z^{sim}) be the completely specified tuples for the real (simulated) protocol. Of course, Z and Z^{sim} can be distinguished simply by examining the initial share values for one family F_i and determining whether they sum to 0 or not.

Say that two tuples are *overlapping*, if for all positions in which neither has a $*$, the values at those positions are the same. Given that $\mathcal{A}$ is a fully adaptive adversary, even though the random bits of $\mathcal{A}$ are fixed, tuples generated from X and X^{sim} may have $*$'s in differing positions. As an example, assume the tuples generated from $\mathcal{X}$ have length 5. Then one tuple might be $(125, *, 30, 39, 27)$. If the adversary were not fully adaptive, the position with the $*$ would be the same for every other tuple in X. However, since the adversary is fully adaptive, another tuple in X might be $(32, 35, *, 39, 27)$. These two tuples are not overlapping, but the tuple $(125, 35, *, 39, 27)$ would overlap with the first tuple.

The tuples in X, X^{sim}, Z, and Z^{sim} can be logically separated into sub-tuples of length r corresponding to the values that the r committees in a family receive during some protocol. In each sub-tuple of length r, there will be either zero or one $*$, and a sub-tuple with no $*$ indicates that the value shared to that committee was previously known. (This can occur during share renewal, for example, when a share known to the adversary is split into shares of shares.)

Proposition 13. *Given that the random bits of $\mathcal{A}$ are fixed, no distinct tuples in $X \cup X^{sim}$, may be overlapping.*

Proof. Consider two distinct tuples in $X \cup X^{sim}$, and consider the runs of the protocol that produced those tuples. Consider the first time that the behavior of

$\mathcal{A}$ diverges in these two runs. Before this time, the positions of $*$'s in the tuples must be the same. Remember that the random bits of $\mathcal{A}$ are fixed, so its actions are only dependent on its view, and when a designated committee is chosen, it is a deterministic function of $\mathcal{A}$'s random bits and $\mathcal{A}$'s view of the protocol to that point. Therefore, the values at some position in the two distinct tuples must be different, and thus the tuples are not overlapping.

Proposition 14. *There is a bijection between the tuples in X (X^{sim}) and the tuples in Z (Z^{sim})*

Proof. Naturally, each tuple in X (X^{sim}) corresponds to at least one tuple in Z (Z^{sim}), since it is $\mathcal{A}$'s view of the tuple in Z (Z^{sim}). Now, each tuple in X (X^{sim}) has at most one $*$ in each r-subtuple. But each of those can only be completed in one way, since the sum of the values in each r-tuple is determined. (For instance, for the r-subtuple of initial shares, the sum is d (0).) Thus there is at most one tuple in Z (Z^{sim}) that can correspond to the tuple in X (X^{sim}).

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

Proposition 15.

$$\mathit{diff}(X, X^{sim}) \leq \frac{2}{N} + \frac{6}{\psi(\eta)}$$

Proof. Let $\hat{X}$ be the distribution X restricted to the case where d' is not in the range $\lfloor N^2/\lambda(N) \rfloor \lambda(N)$ to N^2. Let $\hat{Z}$, $\hat{X}^{sim}$, and $\hat{Z}^{sim}$ be defined similarly. Then $\mathit{diff}(X, X^{sim}) \leq \mathit{diff}(\hat{X}, \hat{X}^{sim}) + 2\lambda(N)/N^2 \leq \mathit{diff}(\hat{X}, \hat{X}^{sim}) + 2/N$. Now we bound $\mathit{diff}(\hat{X}, \hat{X}^{sim})$. We will use the fact that in $\hat{X}$ and $\hat{X}^{sim}$, $d' \bmod \lambda(N)$ is uniform for both the actual protocol and the simulation.

Recall that $\rho = Nm^2r^2c\psi(\eta)$. The probability distribution on tuples generated by $\hat{Z}$ is uniform, meaning each tuple is generated with exactly the same probability. The same holds for $\hat{X}$, $\hat{Z}^{sim}$ and $\hat{X}^{sim}$.

Say a tuple is an *unmatched $\hat{X}$ tuple* if it is in $\hat{X} \setminus \hat{X}^{sim}$, and a tuple is an *unmatched $\hat{X}^{sim}$ tuple* if it is in $\hat{X}^{sim} \setminus \hat{X}$. Finally say a tuple is *unmatched* if it is an unmatched $\hat{X}$ tuple or an unmatched $\hat{X}^{sim}$ tuple. Then $\mathit{diff}(\hat{X}, \hat{X}^{sim})$ is the number of unmatched tuples multiplied by the probability of a given tuple. Note that by symmetry, the number of unmatched tuples will be twice the number of unmatched $\hat{X}$ tuples, so $\mathit{diff}(\hat{X}, \hat{X}^{sim})$ can be calculated by simply doubling the fraction of unmatched $\hat{X}$ tuples. In what follows, we will find an upper bound on the fraction of unmatched X tuples.

To determine if a tuple $v \in \hat{X}$ is an unmatched $\hat{X}$ tuple, we determine whether each $*$ in v could be replaced by a value to produce a tuple in $\hat{Z}^{sim}$. Remember that there could be at most one tuple in $\hat{Z}^{sim}$ that corresponds to v. Basically, we can replace any $*$ in v using the following procedure: Say v_z is the tuple in $\hat{Z}$ corresponding to v, and let x be the value at the location of the $*$ in v_z. Say $\Delta = d' - d'^{sim} \bmod \lambda(N)$. If the $*$ is in an r-subtuple of companion shares or shares of companion shares, then replace the $*$ with $x - \Delta$. If the $*$ is in an

r-subtuple of shares or shares of shares, replace the $*$ with $x-d$. Note that the only way we do not end up with a tuple in $\hat{Z}^{sim}$ is (1) if we replace a $*$ in one of the first $r-1$ shares of the r-subtuple of shares at period 0 with a number not in the range $[-\rho,\rho]$, (1') if we replace a $*$ in one of the first $r-1$ companion shares of an r-subtuple of companion shares at period 0 with a number not in the range $[-N\rho,N\rho]$, (2) if we replace a $*$ in one of the first $r-1$ positions of an r-subtuple of update shares with a number not in the range $[-\rho(t+1)^2,\rho(t+1)^2]$, or (2') if we replace a $*$ in one of the first $r-1$ positions of an r-subtuple of companion update shares with a number not in the range $[-N\rho(t+1)^2,N\rho(t+1)^2]$. Say an r-subtuple is *discriminating* one of these cases holds. (1) and (2) will each hold for d possible values of x. (1') and (2') will each hold for Δ possible values of x. To get a bound on the fraction of unmatched X tuples, we can place an upper bound on the fraction of tuples containing discriminating r-subtuples.

For each r-subtuple containing shares, for each position with a given value α, there are $(2\rho)^{r-2}$ r-subtuples. For each r-subtuple containing companion shares, for each position with a given value α, there are $(2N\rho)^{r-2}$ r-subtuples. For each r-subtuple containing shares of shares from period $2t+1$, for each position with a given value α, there are $(2\rho(t+1)^2)^{r-2}$ r-subtuples. For each r-subtuple containing shares of companion shares from period $2t+1$, for each position with a given value α, there are $(2N\rho(t+1)^2)^{r-2}$ r-subtuples.

To simplify the computation of the upper bound we desire, we will count each tuple with more than one discriminating r-subtuple multiple times, once for each discriminating r-subtuple. The fraction of tuples with discriminating subtuples is then bounded by

$$dm(r-1)\left[(2\rho)^{r-2}(2\rho)^{-(r-1)}+mrc\sum_{t=0}^{T-1}\frac{(2\rho(t+1)^2)^{r-2}}{(2\rho(t+1)^2)^{r-1}}\right]$$
$$+\Delta m(r-1)\left[(2N\rho)^{r-2}(2N\rho)^{-(r-1)}+mrc\textstyle\sum_{t=0}^{T-1}\frac{(2N\rho(t+1)^2)^{r-2}}{(2N\rho(t+1)^2)^{r-1}}\right].$$

Then $\mathit{diff}(\hat{X},\hat{X}^{sim})$ can be bounded by twice that number, which can be simplified to:

$$2dm(r-1)\left(\frac{1}{2\rho}+\sum_{t=0}^{T-1}\frac{mrc}{2\rho(t+1)^2}\right)+2\Delta m(r-1)\left(\frac{1}{2N\rho}+\sum_{t=0}^{T-1}\frac{mrc}{2N\rho(t+1)^2}\right)$$
$$=\frac{md(r-1)}{\rho}\left(1+mrc\sum_{t=0}^{T-1}(t+1)^{-2}\right)+\frac{m\Delta(r-1)}{N\rho}\left(1+mrc\sum_{t=0}^{T-1}(t+1)^{-2}\right)$$
$$\leq\frac{2mN(r-1)(3mrc)}{\rho}=\frac{2mN(r-1)(3mrc)}{Nm^2r^2c\psi(\eta)}<\frac{6}{\psi(\eta)}.$$

This ends the proof of Lemma 11.

Simulation with Semantically-Secure Encryption

Lemma 16. *Given a probabilistic polynomial time (k',k,l)-restricted fully adaptive adversary adversary $\mathcal{A}$, views of $\mathcal{A}$ from executions of the real and simulated*

protocols using non-committing encryption are polynomial-time indistinguishable.

Proof. This follows directly from [BeHa92].

Proof of the Theorem Now we finish the proof of Theorem 8.

Let η be the security parameter, and let GE be the key generator described in the definition of a (k', k, l)-secure proactive RSA system. Assume that the Adaptive Protocol (call it S) is not a (k', k, l)-secure proactive RSA system.

Then there exists a probabilistic polynomial-time (k', k, l)-restricted fully adaptive adversary $\mathcal{A}$, and a polynomial-time function A such that for a list L of randomly chosen messages submitted to be signed, $\Pr[(u^e \equiv w \bmod N) \vee (g^\alpha \equiv h^\beta \bmod N) : (e, d, N) \leftarrow GE(1^\eta); g, h \in_R Z_N^*; w \in_R \{0,1\}^\eta; u, \alpha, \beta \leftarrow A(1^\eta, w, \text{view}_{\mathcal{A},L}^{S(e,d,N,g,h)})]$ is non-negligible.

However, no polynomial-time function A could compute a signature u for a random challenge message $w \in_R \{0,1\}^h$ with more than negligible probability given only $\text{view}_{\mathcal{A},L}^{\text{SIM}(e,N,\mathcal{A},H,g,h)}$, (where $H = \text{hist}(d, N, L)$) since then a polynomial-time function could first compute $\text{view}_{\mathcal{A},L}^{\text{SIM}(e,N,\mathcal{A},H,g,h)}$ and then produce a signature for w, contradicting the RSA security assumption. Similarly, no polynomial-time function A could compute α and β such that $g^\alpha \equiv h^\beta \bmod N$ given only $\text{view}_{\mathcal{A},L}^{\text{SIM}(e,N,\mathcal{A},H,g,h)}$, since then a polynomial-time function could first compute $\text{view}_{\mathcal{A},L}^{\text{SIM}(e,N,\mathcal{A},H,g,h)}$ and then produce the desired α and β contradicting Lemma 10.

Therefore (still assuming the system is not secure), one could construct a distinguisher between $\text{view}_{\mathcal{A},L}^{\text{SIM}(e,N,\mathcal{A},H,g,h)}$ and $\text{view}_{\mathcal{A},L}^{S(e,d,N,g,h)}$, which contradicts Lemma 16. Thus S must be (k', k, l)-secure.

5 Robustness of the Adaptively-Secure Protocol

For a public RSA modulus N and two generators $g, h \in Z_N^*$, we say that one *breaks* (g, h, N) if one finds α and β with $(\alpha \neq 0) \vee (\beta \neq 0)$ such that $g^\alpha \equiv h^\beta \bmod N$.

In the following definitions, for each good committee $C_{i,j}$, we only consider the $a_{i,j}^{2t}$ and $w_{i,j}^{2t}$ values agreed on by the good servers in $C_{i,j}$, if they exist.

We say a proactive RSA system is *correct at period* $2t$ if $d \equiv \sum_{j \in J} a_{i,j}^{2t} \equiv \sum_{j \in J} \tilde{a}_{i,j}^{2t} \bmod \lambda(N)$ and $d' \equiv \sum_{j \in J} w_{i,j}^{2t} \equiv \sum_{j \in J} \tilde{w}_{i,j}^{2t} \bmod \lambda(N)$ for all good families F_i. We say a proactive RSA system is *verifiable at period* $2t$ if for each good committee $C_{i,j}$ $b_{i,j}^{2t} = g^{a_{i,j}^{2t}} h^{w_{i,j}^{2t}} \bmod N$ and $\tilde{b}_{i,j}^{2t} = g^{\tilde{a}_{i,j}^{2t}} h^{\tilde{w}_{i,j}^{2t}} \bmod N$. We say a proactive RSA system is *compact at period* $2t$ if all shares at time $2t$ are between $-r\rho(t+1)^2$ and $r\rho(t+1)^2$, and all companion shares at time $2t$ are between $-Nr\rho(t+1)^2$ and $Nr\rho(t+1)^2$. We say a proactive RSA system is *available at period* $2t$ if, after a signature protocol is run for a message M in operational period $2t$, a gateway G can efficiently identify a good family F_i,

identify or compute correct partial (blinded) signatures ($\{M^{\tilde{a}^{2t}_{i,j}} \bmod N\}_{j\in J}$) for that family, and thus compute the correct signature $M^d \bmod N$ for M.

Lemma 17. *With overwhelming probability, for all $t > 0$, the proactive RSA system above is correct, verifiable, and compact at time $2t$.*

Proof. The basic idea is to show that if the system is not correct, verifiable, and compact at a specific time t, then there is a polynomial-time function A that can sign a random message w, or break (g, h, N), using as input the adversary's view of the protocol, thus breaking the security of the system. By Theorem 8 this happens with negligible probability.

We prove the lemma by induction on t. We will prove results related to the $a^{2t}_{i,j}$, $w^{2t}_{i,j}$, and $b^{2t}_{i,j}$ values. Similarly, we can prove the results related to the $\tilde{a}^{2t}_{i,j}$, $\tilde{w}^{2t}_{i,j}$, and $\tilde{b}^{2t}_{i,j}$ values. Since the dealer is trusted, our initial share distribution protocol guarantees the following properties:

- for all good families F_i, $d \equiv \sum_{j\in J} a^0_{i,j} \bmod \lambda(N)$ and $d' \equiv \sum_{j\in J} w^0_{i,j} \bmod \lambda(N)$, i.e., it is correct at period 0,
- for all good committees $C_{i,j}$, $b^0_{i,j} \equiv g^{a^0_{i,j}} h^{w^0_{i,j}} \bmod N$, where g, h are maximal order elements $\bmod N$, i.e., it is verifiable at period 0, and
- all shares are in the range $-r\rho$ to $r\rho$ and all companion shares are in the range $-Nr\rho$ to $Nr\rho$, i.e., it is compact at period 0.

For the inductive step, we will show correctness, verifiability, and compactness for each good family $F_{i'}$. Note that 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 period $2t-1$ to construct the shares $\{a^{2t}_{i',j'}, w^{2t}_{i',j'}\}_{j'\in J}$. In each committee $C_{i',j'}$, it is verified in the Share Renewal protocol that for all $j \in J$, $\epsilon^{2t-2}_{s,i,j,i',j'} \equiv g^{c^{2t-2}_{s,i,j,i',j'}} h^{v^{2t-2}_{s,i,j,i',j'}} \bmod N$. Then $b^{2t}_{i',j'}$ is set to $\prod_{j\in J} \epsilon^{2t-2}_{s,i,j,i',j'} \bmod N$, $a^{2t}_{i',j'}$ is set to $\sum_{j\in J} c^{2t-2}_{s,i,j,i',j'}$. and $w^{2t}_{i',j'}$ is set to $\sum_{j\in J} v^{2t-2}_{s,i,j,i',j'}$. Thus $b^{2t}_{i',j'} \equiv g^{a^{2t}_{i',j'}} h^{w^{2t}_{i',j'}} \bmod N$, which implies verifiability at period $2t$ for family $F_{i'}$.

Now let $\hat{d} = \sum_{j'\in J} a^{2t}_{i',j'} \bmod \lambda(N)$ and $\hat{d}' = \sum_{j'\in J} w^{2t}_{i',j'} \bmod \lambda(N)$. It is also verified in the Share Renewal 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 by induction, it is guaranteed that $\prod_{j\in J} b^{2t-2}_{i,j} \equiv g^d h^{d'} \bmod N$. Then $g^{\hat{d}} h^{\hat{d}'} \equiv \prod_{j'\in J} b^{2t}_{i',j'} \equiv \prod_{j\in J} b^{2t-2}_{i,j} \equiv g^d h^{d'} \bmod N$. If $\hat{d} \neq d$, then it is easy to see that a polynomial-time function A could use the view of $\mathcal{A}$ to break (g, h, N), and thus the security of the system. This implies correctness at period $2t$ for family $F_{i'}$.

For compactness, we first show that unless the system has been broken, $\sum_{j'\in J} a^{2t}_{i',j'} = d$ and $\sum_{j'\in J} w^{2t}_{i',j'} = d'$. Assume $\sum_{j'\in J} a^{2t}_{i',j'} \neq d$. By correctness (which relies on the fact that (g, h, N) is not broken), $\sum_{j'\in J} a^{2t}_{i',j'} \equiv d \bmod \lambda(N)$, and thus $\sum_{j'\in J} \sum_{j\in J} c^{2t}_{s_{i,j},i,j,i',j'} = \sum_{j'\in J} a^{2t}_{i',j'} = d + \kappa\lambda(N)$, for some integer constant κ. Note that for all non-corrupted committees $C_{i,j}$, $a^{2t-2}_{i,j} = \sum_{j'\in J} c^{2t-2}_{s,i,j,i',j'}$, and (from the discussion for verifiability and correctness above) for all other committees $C_{i,j}$, $a^{2t-2}_{i,j} \equiv \sum_{j'\in J} c^{2t-2}_{s_{i,j},i,j,i',j'} \bmod N$. Let J^C_i be the

set of corrupted committees in F_i. Then

$$\sum_{j\in J_i^C}\sum_{j'\in J} c^{2t-2}_{s_{i,j},i,j,i',j'} = d + \kappa\lambda(N) - \sum_{j\in J\setminus J_i^C}\sum_{j'\in J} c^{2t-2}_{s_{i,j},i,j,i',j'} =$$

$$d + \kappa\lambda(N) - \sum_{j\in J\setminus J_i^C} a^{2t-2}_{i,j} = d + \kappa\lambda(N) - (d - \sum_{j\in J_i^C} a^{2t-2}_{i,j}) =$$

$$\kappa\lambda(N) + \sum_{j\in J_i^C} a^{2t-2}_{i,j}$$

Since $\mathcal{A}$ knows $\sum_{j\in J_i^C} a^{2t-2}_{i,j}$ and $\sum_{j\in J_i^C}\sum_{j'\in J} c^{2t-2}_{s_{i,j},i,j,i',j'}$, it must know $\kappa\lambda(N)$, a multiple of $\lambda(N)$, which implies that a polynomial-time function A can use the view of $\mathcal{A}$ to break the underlying RSA function, and thus the security of the system. A similar argument shows that $\sum_{j'\in J} w^{2t}_{i',j'} = d'$.

If the security of the system has not been broken, $\sum_{j'\in J} a^{2t}_{i',j'} = d$ and $\sum_{j'\in J} w^{2t}_{i',j'} = d'$ From the verification in step 4 of the Share Renewal protocol, for any good committee $C_{i',j'}$ with $j' \in J \setminus \{r\}$, $-\rho(t+1)^2 \le a^{2t}_{i',j'} \le \rho(t+1)^2$. Thus $d - (r-1)(\rho(t+1)^2) \le a^{2t}_{i',r} \le d + (r-1)(\rho(t+1)^2)$. Since $d \le N$, $-r\rho(t+1)^2 \le a^{2t}_{i,r} \le r\rho(t+1)^2$. Also for any good committee $C_{i',j'}$ with $j' \in J \setminus \{r\}$, $-N\rho(t+1)^2 \le w^{2t}_{i',j'} \le N\rho(t+1)^2$. Thus $d - (r-1)(N\rho(t+1)^2) \le w^{2t}_{i',r} \le d+(r-1)(N\rho(t+1)^2)$. Since $d \le N$, $-Nr\rho(t+1)^2 \le w^{2t}_{i,r} \le Nr\rho(t+1)^2$. This implies compactness at step $2t$ for family $F_{i'}$.

Theorem 18. *The proactive RSA system above is (k', k, l)-robust against a fully adaptive adversary.*

Proof. Recall that by Lemma 7, with all but negligible probability, the assignment is (k', k, l)-terrific, and thus 90 percent of the families will be good in any period $2t$.

Note that the gateway G is given the outputs of all the committees in all the families, the opened shares at the committees with disputed outputs, and the public witnesses $\{b^{2t}_{i,j}\}_{(i,j)\in I\times J}$. It can easily be verified (using the fact that at least one family is good in any operational period $2t$) that correctness, verifiability, compactness, and availability imply robustness.

Lemma 17 proves that with overwhelming probability, the system is correct, verifiable, and compact during each operational period $2t$. Then for each committee $C_{i,j}$ with bad servers, our protocol allows a good server on that committee to prove its partial signature to be correct (by opening the shares $\tilde{a}^{2t}_{i,j}$ and $\tilde{w}^{2t}_{i,j}$ and verifying them using the public value $\tilde{b}^{2t}_{i,j}$, if there is a dispute). Thus, a gateway G can efficiently identify or compute a correct partial signature for each committee $C_{i,j}$ with a good server. Consequently, it can efficiently identify a good family and compute the correct signature. This implies availability at time $2t$.

Thus the system is (k', k, l)-robust.

References

ACGS. W. Alexi, B. Chor, O. Goldreich and C. Schnorr. RSA and Rabin Functions: Certain Parts are as Hard as the Whole. In *SIAM Journal of Computing*, volume 17, n. 2, pages 194–209, April 1988.

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

AS92. N. Alon and J. H. Spencer, *The Probabilistic Method* Wiley-Interscience, New York, NY, 1992.

B84. E. Bach, Discrete Logarithms and Factoring. Tech. Report No. UCB/CSD 84/186. Computer Science Division (EECS), University of California, Berkeley, CA. June 1984.

B97. D. Beaver, Plug and Play Encryption, Crypto 97, pp. 75-89.

BeHa92. D. Beaver and S. Haber, *Cryptographic protocols provably secure against dynamic adversaries*, EuroCrypt 92, Springer-Verlag, 1993, 307–323.

B79. G.R. Blakley, *Safeguarding Cryptographic Keys*, AFIPS Con. Proc (v. 48), 1979, 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, pp. 425-439.

CFGN96. R. Canetti, U. Feige, O. Goldreich, and M. Naor, *Adaptively Secure Multiparty Computation*, ACM STOC 96, 639–648.

CGJKR. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive Security for Threshold Cryptosystems. In CRYPTO 99, pp. 98-115.

CH94. R. Canetti and A. Herzberg, *Maintaining Security in the presence of transient faults*, Crypto 94, Springer-Verlag, 1994, pp. 425-438.

Che52. H. Chernoff, *A Measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Annals of Mathematical Statistics, 23:493–509, 1952.

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, pp. 522-533. (First version May 92).

DF89. Y. Desmedt and Y. Frankel, *Threshold cryptosystems*, Crypto 89,

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-Verlag 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.

FGMY97. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In Crypto 97, pages 440–454.

FGMY97b. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal-resilience proactive public-key cryptosystems. In FOCS'97, pages 384–393.

FMY99. Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptively-secure distributed public-key systems. ESA 99, July 99.

FMY99b. Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptively-secure Proactive RSA. Asiacrypt 99, pp. 180-194.

FMY98. Y. Frankel, P. MacKenzie, and M. Yung, *Robust Efficient Distributed RSA-Key Generation* STOC, 1998, pp. 663–672.

FMY00. Y. Frankel, P. D. MacKenzie, and M. Yung. "Pseudorandom Intermixing": A Tool for Shared Cryptography, PKC'00.

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, 1989. (Previous version in FOCS 85).

GJKR. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, *Robust Threshold DSS Signatures*, Eurocrypt 96, pp. 354-371.

GJKR96. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, *Robust and Efficient Sharing of RSA*, Crypto 96, pp. 157-172.

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, pp. 339-352.

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.

JJKY95. M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, *Proactive RSA for constant shareholders*, manuscript.

JL00. S. Jarecki and A. Lysyanskaya, Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. In Eurocrypt 2000, pp. 221-242.

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.

P91. T.P. Pedersen, *Non-interactive and information theoretic secure verifiable secret sharing*, Crypto 91, pp. 129-140.

R98. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. In Crypto 98, pp. 89-104.

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.

Sh00. V. Shoup. Practical Threshold Signatures. In Eurocrypt 2000, pp. 207-220.

A Proof of Lemma 7

First we bound the probability that there is a family without an excellent committee. We start with some easily computed quantities:

- Probability that a committee has no bad servers: $(1-\sigma)^c$.
- Probability that a committee has at least one non-bad server: $1-(1-\sigma)^c$.
- Probability of a given family having at least one non-bad server on each committee: $(1-(1-\sigma)^c)^r$.
- Probability of any family having at least one non-bad server on each committee: at most $m(1-(1-\sigma)^c)^r$.

The last quantity is the probability that there is a family without an excellent committee. We bound this as follows, using the fact that $1+x \le e^x$ for all x:

$$m(1-(1-\sigma)^c)^r = 10\eta(1-(1-\sigma)^c)^{\frac{(1-\tau)^{-c}}{\eta}} \le 10\eta(e^{-(1-\sigma)^c})^{\frac{(1-\tau)^{-c}}{\eta}} =$$

$$10\eta\left(e^{\left(\frac{1-\sigma}{1-\tau}\right)^c/\eta}\right) = 10\eta(e^{-\eta^2/\eta}) = 10\eta e^{-\eta}$$

which is negligible in η.

Next we bound the probability that more than 10 percent of the families each have a committee with no good server. Again we start with some easily computed quantities:

- Probability that a committee has no good server: $(1-\tau)^c$.
- Probability that a committee has at least one good server: $1-(1-\tau)^c$.
- Probability that all committees in a family have at least one good server: $(1-(1-\tau)^c)^r$.
- Probability that some committee in a family has no good servers: $1-(1-(1-\tau)^c)^r$.

Now we bound $(1-\tau)^c$.

$$(1-\tau)^c = (1-\tau)^{\frac{2\log\eta}{\log((1-\sigma)/(1-\tau))}} = 2^{\frac{2\log(1-\tau)\log\eta}{\log((1-\sigma)/(1-\tau))}} = 2^{\frac{-2\log((1-\tau)^{-1})\log\eta}{\log((1-\sigma)/(1-\tau))}}$$

$$\le 2^{-2\log\eta} \le \frac{1}{2}$$

Now we bound the probability that some committee in a family has no good servers, using the fact that $1-x \ge e^{-2x}$ for $0 \le x \le \frac{1}{2}$:

$$1-(1-(1-\tau)^c)^r \le 1-(e^{-2(1-\tau)^c})^{(1-\tau)^{-c}/\eta} \le 1-e^{-2/\eta} \le \frac{2}{\eta} \le \frac{1}{50}$$

Now we state a Chernoff bound [AS92,Che52]:

Fact 1 *Let X be a random variable with binomial distribution $B(n,p)$, i.e., n independent trials each with probability of success p. Then* $\Pr[X \ge (1+\epsilon)np] \le [e^{\epsilon}(1+\epsilon)^{-1+\epsilon}]^n p$.

A simple corollary is: $\Pr[X \ge 5np] \le 2^{-np}$.

Using this fact, we can bound the probability of more than 10 percent of families having some committee with no good servers by $2^{-m/50} = 2^{-\eta/5}$, which is negligible in η.