A Simplified Approach to Threshold and Proactive RSA

Tal Rabin

IBM T.J. Watson Research Center
PO Box 704
Yorktown Heights, New York 10598
talr@watson.ibm.com

Abstract. We present a solution to both the robust threshold RSA and proactive RSA problems. Our solutions are conceptually simple, and allow for an easy design of the system. The signing key, in our solution, is shared at *all times* in additive form, which allows for simple signing and for a particularly efficient and straightforward refreshing process for proactivization. The key size is (up to a very small constant) the size of the RSA modulus, and the protocol runs in constant time, even when faults occur, unlike previous protocols where either the size of the key has a linear blow-up (at best) in the number of players or the run time of the protocol is linear in the number of faults. The protocol is optimal in its resilience as it can tolerate a minority of faulty players. Furthermore, unlike previous solutions, the existence and availability of the key throughout the lifetime of the system, is guaranteed without probability of error.
These results are derived from a new general technique for transforming distributed computations for which there is a known n-out-n solution into threshold and robust computations.

Keywords: RSA, threshold signatures, proactive signatures, threshold and proactive RSA

1 Introduction

In the distributed computation model there are problems for which the simplest solution is achieved by requiring that all players participate in the computation. Yet, clearly this requirement mandates to give up other very desirable properties, such as enabling a threshold of players to carry out the computation, and to withstand faulty behavior of players. The question which arises is are these things contradictory or whether we can utilize the simple protocol but still preserve the properties of threshold and robustness. In this paper we present a general paradigm for transforming an *all* player protocol into a threshold and robust

protocol, while maintaining the original simplicity of the protocol. We will apply this paradigm to the specific problem of threshold and proactive RSA.

The issue of maintaining secret signing keys in a distributed fashion for long periods of time, while enabling uninterrupted signing capabilities is addressed by threshold signatures and proactive security.

Threshold signature schemes allow a group of players to "hold" the key and enable them, as a group, to produce signatures. We shall refer to such a scheme as a (t, n)-*threshold signature* scheme if there are n players and given a message m, any subset of players of size at least $t + 1$ can generate the signature on m. The scheme is *t-secure* if no subset of at most t players can compute a signature on a message not previously signed. A player will be considered *corrupted* if he does not follow his intended protocol. The scheme is *t-robust* if it can correctly compute signatures even in the presence of up to t corrupted players. The basic assumption for threshold schemes is that there are at most t faults during the lifetime of the system.

Proactive signature schemes use threshold signature schemes as the basis but drastically reduce the assumption concerning failures. The lifetime of the system is split into time periods, and in each such time period it is assumed that no more than t players will be corrupted. We shall say that a scheme is *t-proactive* if it is secure and robust even in the presence of up to t corrupted players in every time period, where the set of corrupted players can change arbitrarily from one time period to the next. The main idea would be to preserve the value of the key but to change its representation from one period to another, so that the representations are independent. Thus, an attacker wishing to compromise the system will need to corrupt $t + 1$ players in a short period of time.

Threshold signatures are part of a general approach known as *threshold cryptography* which was introduced in the works of Desmedt [Des87], Boyd [Boy89], Croft and Harris [CH89] and Desmedt and Frankel [DF89][1]. Secure robust threshold schemes for discrete log based systems have been given: El-Gamal [CMI93,Har94] and DSS [GJKR96a]. Yet, solutions for RSA seemed more elusive; the need to protect the secrecy of the factors of the composite N, among other things, disabled the direct adaptation of the techniques used in the discrete log cases to RSA. In two independent works [Boy89,Fra89] gave a first simple and elegant solution for distributed RSA, which was to maintain the signing key d (the RSA exponent) in additive form, among n players $P_1, ..., P_n$. Player P_i would hold a value d_i, where $d = d_1 + ... + d_n$. This allows for a straight forward signing scheme, as $m^d = m^{d_1} \cdot .... \cdot m^{d_n}$. Thus, each player P_i gives the value m^{d_i} which is called the *partial signature*, and all the partial signatures are combined to form the complete signature on m. The major drawback of the n-out-n additive solution is that it does not provide for threshold (and hence robustness); if each one of these shares is held by a different player, and one of the players crashes we have lost the secret key. Desmedt and Frankel [DF91] initiated the study of threshold-RSA and gave a heuristic solution to the problem, which was followed by provably secure threshold schemes [FD92,DDFY94]. These schemes

[1] For a '94 survey we refer to [Des94].

provided a step forward, however, in order to overcome the problems generated by the structure of the RSA key, very cumbersome mathematical structures were introduced. Two results followed, which used these same structures, but added robustness [FGY96,GJKR96b].

Proactive signatures are part of a general approach known as proactive security which was introduced in the works of Ostrovsky and Yung [OY91] and Canetti and Herzberg [CH94][2]. The notion of proactive signatures and a framework for discrete log based proactive signatures was given in [HJJ+97], in particular incorporating the threshold DSS [GJKR96a] into this framework achieves a proactive DSS. The first solution for proactive RSA was given by Jakobsson et al. [JJKY95], but this solution was exponential in the size of the threshold. Frankel et al. [FGMY97a] used a probabilistic construction to enable a non-exponential solution. While being quite efficient for small instances this solution does not seem to scale gracefully. Furthermore, their solution did not have optimal resilience, i.e. it could not tolerate the maximal possible number of corrupted players. In a recent paper by Frankel et al. [FGMY97b] the first realistic and general solution to proactive RSA was given. By introducing new techniques they modified and proved the protocol of [DF91], and further extended it to solve the problem of proactiveness[3].

Utilizing our new paradigm we present a solution to both the robust threshold RSA and proactive RSA problems. The main advantage of our solutions, over previous ones, is that they are conceptually simple, and allow for an easy design of the system. We return to the construction of sharing the key at *all times* in additive form, which allows for simple signing and for a particularly efficient and straightforward refreshing process for proactivization. The key size is (up to a very small constant) the size of the RSA modulus, and the protocol runs in constant time, even when faults occur, unlike previous protocols where either the size of the key has a linear blow-up (at best) in the number of players or the run time of the protocol is linear in the number of faults. The protocol is optimal in its resilience as it can tolerate a minority of faulty players. Furthermore, the existence and availability of the key throughout the lifetime of the system, is guaranteed without probability of error.

2 Overview

Achieving Threshold.[4] As stated the key will be shared using an n-out-n additive sharing, this results in each player holding an additive share d_i of the key. The threshold will be achieved through the notion of *share back-up*. That is, the share d_i will be further shared using a t-out-n secret sharing scheme. Thus, if

[2] For a survey we refer to [CGHN97].

[3] Some related results appear in [DJ97].

[4] We assume a trusted dealer in our constructions but this can be substituted using the results of [BF97] for generating an RSA key in a distributed manner. Thus, the following description of our protocol starts after the key has been distributed to the players.

at some point a player crashes, we will be able to retrieve that player's additive share by activating the back-up using the reconstruction process. We shall show later ways to avoid the exposure of the additive share. The threshold property of our scheme will be inherited from the threshold property of the secret sharing scheme. To achieve robustness the simple threshold secret sharing scheme can be substituted by a verifiable secret sharing scheme (VSS), which can tolerate t malicious faults. Thus, we achieved our goal of maintaining the sharing in additive form while providing threshold.

Proactive Refreshing of Shares. The above additive scheme also lends itself nicely to proactivization, which requires to maintain the same key value yet to change its representation. A new set of additive shares $d_1^{new}, ..., d_n^{new}$ will be generated by having each additive share d_i sub divided into $d_{i,1}, ..., d_{i,n}$ and recombining these values into new additive shares by summing $d_i^{new} = \sum_{j=1}^{n} d_{j,i}$. We have that $\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} d_i^{new} = d$, i.e. a new representation of d. Measures need to be taken to ensure that this resharing has been executed properly, this will be done via a comparison of publicly known values.

Share Back-up Refreshing. We will say that the share back-up of a player is *valid* if the value represented by the VSS is the share held by the player. Clearly, this is a requirement without which the whole idea of back-up will not work. The initial dealer is trusted, hence we assume that the back-up is valid. But now that each player has a new additive share, the share back-up needs to be refreshed as well, as it is no longer a valid back-up of the new share. This is achieved by having each player share his new share using a VSS protocol. The verification that the share back-up is valid is carried out through a comparison of values. The most important point to note is that *all verifications* performed in this process are computed on publicly known information and *without any zero-knowledge proofs* for possession of discrete log which characterize the solution of [FGMY97b]. Discarding the zero-knowledge proofs from the proactivization process is not only an issue of efficiency in computation/communication, but allows for a simpler design.

Achieving Robustness. The decision on when to activate the back-up is more complex. It can be activated in cases where crashes occur, because then we know which share to reconstruct. But in the case where players are acting in a malicious manner, we first have to determine who they are before reconstructing their share from the back-up. This will be done by employing the robustness techniques which enable to verify that a partial signature was generated correctly [FGY96,GJKR96b].

Avoiding Share Exposure. It should be noted that the exposure of a player's share when he fails does not compromise the security of the system because as long as not all shares are exposed the secrecy of the key is maintained. And as it is an assumption of the model in threshold cryptography that at no time are all the servers crashed or compromised, the security is maintained. Yet, if it is undesirable in some applications (parallel signing) or situations (a simple crash) to expose a player's share then this can be avoided through the following.

The back-up of a value will not be a single VSS but rather the value itself will be split into n parts, one assigned to each player, each such part will be further shared using VSS[5]. When a player does not provide the needed partial signature then each part will be privately reconstructed to the player assigned to it. Parts assigned to faulty players will be reconstructed publicly. Thus, the original share of the player will not be exposed. If at a later time the player recovers from the crash, then his share can be privately reconstructed to him, and he would be required to refresh the back-up (even though it is valid), and then he can resume proper operation.

A further extension would also enable the parallel signatures.

Practical Efficiency. In the signature generation, when no faults occur, each player needs to compute a single exponentiation using a small key which is about 2 times the length of the RSA modulus. It can be optimized to eliminate the zero-knowledge proofs by checking whether the generated signature is valid. When faults occur there is the need to first check which players are faulty and then to recover the additive share of these players from the back-up. The recovery operation is an execution of a VSS Reconstruction Phase, which is an efficient protocol. Both the signature generation and the proactivization protocols are constant round protocols, even when faults occur. The proactivization process does not require zero-knowledge proofs in any case.

General Paradigm The above described solution is in fact a general paradigm for transforming distributed n-out-n computations into threshold and robust computations. Given a distributed n-out-n computation where each player holds inputs which are need for the computation, these inputs need to be shared using a VSS scheme for back-up.

The computation is carried out using all n players, but in the event that one (or up to t) of the players fails to cooperate, or is faulty, then his information can be reconstructed from the "back-up" copy and incorporated into the computation. Thus, the simplicity of the n-out-n protocol can be maintained while adding the properties of threshold and robustness.

3 Model

The following description is taken in part from [HJJ+97].

Communication Model. We assume that our computation model is composed of a set of n *players* $\{P_1, \ldots, P_n\}$ who can be modeled by polynomial-time randomized Turing machines. They are connected by a complete network of private (i.e. untappable) point-to-point channels. In addition, the players have access to a dedicated broadcast channel; by dedicated we mean that if player P_i broadcasts a message, it will be recognized by the other players as coming from P_i. These assumptions (privacy of the communication channels and dedication of the broadcast channel) allow us to focus on a high-level description of the protocols. However, it is worth noting that these abstractions can be substituted with

[5] For related techniques see [GHY87]

standard cryptographic techniques for privacy, commitment and authentication. Yet, in the proactive case, under these substitutions care needs to be given to refreshing the communication keys.

Time. Time is divided into *time periods* which are determined by the common global clock. Each time period consists of a short refresh period, during which the players engage in an interactive refreshing protocol. After the refresh, there is a signature generation period, in which the players generate signatures on given messages.

The Adversary. We assume an adversary, $\mathcal{A}$, which can corrupt up to t of the n players in the network in each time period. A player is considered corrupted during a time period if he was corrupted during that time period or if he was corrupted during the preceeding refresh period. Furthermore, it is a *malicious* adversary, i.e. it learns all the information held by the corrupted players, and may cause them to divert from the specified protocol in any (possible malicious) way. We assume the adversary to be computationally bounded, implying that he cannot forge RSA signatures. Furthermore, we assume the adversary to be static[6].

4 Robust Threshold RSA

Following the method set forth in this paper we shall share the RSA secret key by a sum of shares, refer to these shares as *additive-shares*. We will achieve robustness by generating *share back-up*, i.e. each such additive-share will be further shared using a robust threshold verifiable secret sharing (VSS) scheme. Thus, we are guaranteed that we will not "lose" the additive-shares. Upon request to generate a signature each player will produce a partial signature under his additive-share. These partial signatures are combined to generate the signature. In order to optimize the protocol we will first verify whether this combined signature is valid, and if it is not then we will proceed to detect the players who have given incorrect partial signatures. We shall have a method which enables a player to prove that his partial signature is correct. If a player fails to provide the system with a correct partial signature his additive-share is reconstructed from the back-up.

In order to make the composition of the additive-shares and the shares from back-up work properly, an underlying assumption has been made, and it is that the back-up value of a specific player is in fact the additive-share held by him. For ease of exposition we assume that the original dealer, sharing the secret, is honest and hence this property is satisfied, but when we move into the full proactive solution this will no longer be a valid assumption. Thus, we bear in mind this point, and will return to it in Section 5.

We shall start by directly describing the Robust Threshold RSA while assuming a black-box VSS protocol. This VSS protocol enables to share a secret via

[6] In the full paper we shall give a proof for a static adversary at each time period. And another proof for an adaptive adversary, but at a cost in the computations of the zero-knowledge proofs.

its Sharing Phase, and to reconstruct a secret using its Reconstruction Phase. Denote this protocol as Feldman-Z_n-VSS, (see Section 4.3).

The robust threshold RSA is comprised of two protocols, an initial key distribution which is carried out once at the onset of the system. The second is the protocol for generating signatures. These protocols are described in detail in Sections 4.1 and 4.2.

4.1 Distributing the Secret Key

Before distributing the secret key we assume that a set-up process has been carried out in which the RSA key generation took place and the RSA key pair has been computed[7]. Denote the public key by (N, e) where $N = pq$ and p, q are primes of the form[8] $p = 2p' + 1$, $q = 2q' + 1$ and p', q' are primes. The private key is d where $ed \equiv 1 \bmod \phi(N)$. Furthermore, the parameters of the system are set, i.e. the number of players n and the threshold t, where $n \geq 2t + 1$. An element g of high order is chosen. In order to enable the proof of security the high order element will be computed by setting $g \triangleq g_0^{L^2} \bmod N$ where g_0 is an element of high order and $L \triangleq n!$.

After this set-up process the dealer proceeds to share the key d by generating shares which sum up to the private key, and then backing-up each one of the shares using a VSS protocol. Furthermore, as discussed we will need to be able to verify partial signatures generated under each additive-key d_i. The dealer lays down this ability by generating a *witness signature* for each additive-key in the form $g^{d_i} \bmod N$. This witness will be utilized in the signature generation protocol in order to verify partial signatures. In order to unify the description for the players we added a public share d_{public}, such that $d = d_{public} + \sum_{i=1}^{n} d_i$. The steps carried out by the dealer are described in Figure 1.

The size of the key used for signature generation, in our protocol is comparable to that of [FGMY97b], and is $2\log(nN)$, and much smaller than that of [DDFY94] which is $n \log N$ and [FGMY97a] which is $10^6 \log N$ for $n = 10, t = 4$. Our memory requirements are comparable to those of [DDFY94] and are $2n \log(nN)$, yet larger by a factor of n than those of [FGMY97b].

4.2 Signature Generation

Given a message m, its signature under the public key is $m^d \bmod N$. In our setting this signature needs to be generated by the players in a distributed manner where each individual player uses his partial key. As the secret key d is shared using a sum, i.e. $d = d_{public} + \sum_{i=1}^{n} d_i \in Z$, we have that $m^d =$

[7] For a distributed key generation process see Boneh and Franklin [BF97]

[8] The special form of the primes is required only to enable the efficient protocol of [GJKR96b] for proving that a partial signature is correct. If there is a need to carry out a distributed key generation then the requirement on the format of the primes can be dropped and the techniques of [FGY96] can be employed for verifying the partial signature.

Input: secret key $d \in Z_{\phi(N)}$, composite N, element g of high order in Z_N^*
number of players n and threshold t

1. Choose and hand P_i value $d_i \in_R [-nN^2..nN^2]$ for $1 \leq i \leq n$, set $d_{public} = d - \sum_{i=1}^{n} d_i$.
2. Compute and broadcast the witness $w_i \triangleq g^{d_i} \bmod N$, $1 \leq i \leq n$.
3. Share the value d_i using Feldman-Z_N-VSS Sharing Phase (Figure 3) on input, value d_i, high order element g, composite N, the number of players n, and threshold t.

Fig. 1. Secret Key Distribution

$m^{d_{public}+\sum_{i=1}^{n} d_i} = m^{d_{public}} \prod_{i=1}^{n} m^{d_i} \bmod N$. Thus, if each player publicizes the correct value $m^{d_i} \bmod N$ which we refer to as the partial signature, then the signature can be directly computed from the partial signatures. And in fact the signature can be generated and verified before proceeding to verify each partial signature. Thus, if an error was detected we will require each player to prove that he has generated his partial signature properly. For each share d_i there is a public witness $w_i \bmod N$ which can be viewed as a commitment to the value d_i. When player P_i is required to prove the correctness of his partial signature σ_i, he will prove with respect to the witness w_i that the discrete log of the partial signature is equivalent to the discrete log of the witness. If the player fails this proof, i.e. he did not generate a proper partial signature, his share d_i will be retrieved from the back-up VSS. This results in a constant round protocol even in the presence of faults. The protocol for signature generation is described in Figure 2.

Public information: high order element g, composite N
$w_i \bmod N$ for $1 \leq i \leq n$ witness for P_i's partial key
Input: message m

1. Player P_i publishes partial signature $\sigma_i = m^{d_i} \bmod N$
2. P_i proves using the protocol of [GJKR96b] (see Footnote 8) that $DL_m \sigma_i = DL_g w_i$.
3. If proof fails for player P_i all players reconstruct d_i using the Feldman-Z_N-VSS Reconstruction Phase (Figure 3), and compute $\sigma_i = m^{d_i} \bmod N$.
4. Set $SIG(m) = m^{d_{public}} \prod_{i=1}^{n} \sigma_i \bmod N$.

Fig. 2. Signature Generation

4.3 Verifiable Secret Sharing (VSS) over Z_N

In our protocol we need to share a value $s \in [-nN^2..nN^2]$ in a verifiable method. As explained above we are in effect "storing" the value s using the VSS protocol, as a back-up in case one of the players fails. Given that the only motivation for sharing s would be for back-up it seems that we could use any VSS protocol computed over a prime field, where the prime is larger than the possible range of s. In theory this is true, and we could use for example the VSS protocol of [BGW88], which is information theoretically secure. But as we have alluded to previously we will need to check that the value shared for player P_i using the VSS corresponds to the actual additive-share held by this player. Using general zero-knowledge proofs this could still be done. Yet, we would like to provide more efficient methods for achieving these goals[9]. Thus, we will depart from carrying out the VSS over a prime field, and modify the protocol to work over the integers. Such solutions have been given by [DF91,FGMY97b], yet their solutions need to satisfy additional properties, which we do not require, hence these solutions are slightly more complex[10]. The following protocol is similar to the one described in [FGMY97b], but which satisfies our needs. As the protocol is based on Feldman's VSS [Fel87], we shall refer to it as Feldman-Z_N-VSS.

Theorem 1. *The protocol described in Figure 3 is a verifiable secret sharing scheme satisfying the properties of unanimity, acceptance of good secrets, verifiability and unpredictability*[11].

We shall prove only the unpredictability property in Lemma 1 and 2, as the proof of the other properties follow directly from Feldman's proof [Fel87]. We defer the simulation of the protocol to Section 4.4 where we prove the Robust Threshold RSA.

Lemma 1. *Given a t-adversary who can corrupt at most t players, the view of the adversary of the secret shares generated by the protocol Feldman-Z_N-VSS of a secret s using a polynomial $f(x)$, such that $f(0) = s$, and of the sharing of a random secret r by a polynomial $r(x)$ with coefficients taken from the appropriate range are statistically indistinguishable.*

Proof appears in Appendix A.

---

[9] It may be possible to use Feldman's VSS over a prime field, which also enables efficient proofs of the above mentioned property, but this sharing would also reveal $g^{d_+} \bmod p$. Though this additional information may not weaken the system, we have not found a way to prove its security.

[10] [FGMY97b] need to work over groups where the polynomial interpolation can satisfy that the Lagrangian coefficient of each component is easily computed and is in Z. In our protocol when we decide to use the VSS back-up it will be to directly reconstruct it, thus we can expose all shares and there is an option to carry out the interpolation in a prime field (where the prime will be chosen to be of a certain size).

[11] For the definition of VSS see [FM88].

Sharing Phase:

Input: secret value $s \in [-nN^2..nN^2]$ and composite N
element g of high order in Z_N^*
n number of players, t threshold value

The dealer carries out the following steps:

1. choose $a_t, ..., a_1 \in [-nL^2N^3..nL^2N^3]$ and define $f(x) = a_t x^t + ... + a_1 x + sL$
2. compute $f(i) \in Z$ for $1 \leq i \leq n$
and $b_t \triangleq g^{a_t}, ..., b_1 \triangleq g^{a_1}, b_0 \triangleq g^{sL} \bmod N$
3. hand player P_i the value $f(i)$ and broadcast $b_t, ..., b_0$

Verification steps:

1. Player P_i verifies that $g^{f(i)} = \prod_{j=0}^{t} (b_j)^{i^j}$ if the equation is not satisfied he requests that the dealer make $f(i)$ public.
2. The dealer broadcasts all shares requested in the previous step, if he fails to do so he is disqualified.
3. Player P_i carries out the verification of Step 1 for all public shares. If the verification fails the dealer is disqualified.

Reconstruction Phase:

Input: high order element g, composite N, number of players n, threshold t
values $b_t, ..., b_0 \bmod N$

1. Player P_i broadcasts $f(i)$
2. P_j finds a set I of size $t+1$ of indices such that $\forall i \in I$ it holds that $g^{f(i)} = \prod_{j=0}^{t} (b_j)^{i^j}$
3. P_j chooses a prime $P > 2nN^2$ and computes the secret s, $s \triangleq \sum_{i \in I} f(i) \prod_{j \in I, j \neq i} \frac{-j}{i-j} / L \bmod P$

Fig. 3. Feldman-Z_N-VSS

Lemma 2. *Assume elements $g_0, g \triangleq g_0^{L^2}$ ($L = n!$) of maximal order in Z_N^*. Given values $\alpha_1, ...\alpha_t$ and an additional value $g^s \bmod N$ it is possible to compute values $g^{a_1}, ..., g^{a_t} \bmod N$ such that the polynomial $f(x) \triangleq a_t x^t + ... + a_1 x + sL$ satisfies that $f(i) = \alpha_i$ for $1 \leq i \leq t$.*

Proof. Define the polynomial $f(x) \triangleq \sum_{i=0}^{t} \frac{\alpha_i}{\prod_{j \neq i} i-j} \prod_{j \neq i} (x-j)$ where $\alpha_0 \triangleq sL$. This polynomial satisfies the property that $f(i) = \alpha_i$ for $1 \leq i \leq t$, thus it remains to be shown that we can compute g raised to the coefficients of this polynomial. Rearranging terms we have that the coefficient of x^k is $a_k = \sum_{i=0}^{t} \frac{\alpha_i}{\prod_{j \neq i}(i-j)} \lambda_{k,i}$ for $0 \leq k \leq t$, where the $\lambda_{k,i}$ are computable known constants. A problem may arise that we would need to deal with fractions, i.e.

extract roots, which is infeasible, but this has been solved through the choice of the element g. Thus g^{a_k} can be effectively computed as follows:

$$g^{a_k} = g^{\sum_{i=0}^{t} \frac{\alpha_i}{\prod_{j \neq i}(i-j)} \lambda_{k,i}} = \prod_{i=0}^{t} g^{\frac{\alpha_i}{\prod_{j \neq i}(i-j)} \lambda_{k,i}} = g_0^{\alpha_0 \lambda_{k,0} \frac{L^2}{l!}} \prod_{i=1}^{t} (g_0^{\alpha_i \lambda_{k,i}})^{\frac{L^2}{\prod_{j \neq i}(i-j)}}$$

$$= (g^s)^{\lambda_{k,0} \frac{L}{l!}} \prod_{i=1}^{t} (g_0^{\alpha_i \lambda_{k,i}})^{\frac{L^2}{\prod_{j \neq i}(i-j)}}$$

4.4 Proof of Robust Threshold RSA Protocol

Theorem 2. *Under the assumption that factoring is intractable the protocol described in Figures 1 and 2 is a constant round secure, robust, threshold RSA scheme, in the presence of t malicious faults where the total number of players is $n \geq 2t + 1$.*

In order to prove the security of our scheme we will use a simulation argument for the view of the adversary. Intuitively, this means that the adversary who sees all the information of the corrupted players and the signature on m, could generate by itself all the other public information produced by the protocol. A simulator for the key distribution and the signature generation is shown in Figure 4. It runs on input the elements g, g_0 the value $g^d \bmod N$ which is the commitment to the key d, and the message m and its signature. The proof of Theorem 2 appears in Appendix A.

5 Proactive RSA

In the previous section we have described a robust threshold RSA, and the last property which we would like to add is proactive. Proactiveness is achieved through changing the representation of the key, but not its value. Given that the current representation of the key is $d_1 + ... + d_n$ we would like to change it to $d_1^{new} + ... + d_n^{new}$. Once we have changed the representation of the key we will also need to change the representation of the back-up, and this is for two reasons: the first being that it contains the current representation and thus it should be removed, the second is that we need to have a valid back-up for the new representation, otherwise the back-up would be worthless.

The essence of the proactivization protocol is that each player takes his additive-share d_i of the key d, splits it into sub-shares which sum up to d_i, and gives each player such a sub-share. Then each player adds up all the sub-shares which he received in order to attain his new share for the new time period. If all players act properly it is clear to see that the new sharing is still a sharing of the key d, yet the sharing is totally independent of the previous sharing. Then each player would further share his new additive share using the VSS protocol to generate the share back-up.

As we do not assume that the players act properly we add steps to ensure the correctness of the proactivization. The two issues for verification are the following:

Input: elements $g_0, g \triangleq g_0^{L^2}$, the commitment to the signing key $g^d \bmod N$
the number of players and threshold n, t
message m, and its signature $m^d \bmod N$

Each step of the simulator corresponds to the same numbered step in the appropriate protocol. Note that information held solely by good players is never exposed, and thus we do not simulate it.

$\mathcal{SIM}1$ - Computation

Secret Key Distribution

1. choose shares $\hat{d}_1, ..., \hat{d}_{n-1} \in_R [-nN^2..nN^2]$ and $\hat{d}_{public} \in_R [-n^2N^2..n^2N^2 + N]$
2. (a) compute $\hat{w}_1 \triangleq g^{\hat{d}_1}, ..., \hat{w}_{n-1} \triangleq g^{\hat{d}_{n-1}} \bmod N$
   (b) set $\hat{w}_n = g^{\hat{d}_n} \triangleq g^d / (g^{\hat{d}_{public}} \prod_{i=1}^{n-1} \hat{w}_i) \bmod N$
3. Feldman-Z_N-VSS
   (a) using Steps 1-3 of Figure 3 share the value $\hat{d}_i$ for $1 \leq i \leq n-1$, this results in polynomials $\hat{f}_i(x)$ such that $\hat{f}_i(0) = \hat{d}_i L$, and public information $g^{\hat{a}_{i,t}}, ..., g^{\hat{a}_{i,1}}, g^{\hat{d}_i L} \bmod N$, where $\hat{f}_i(x) = \hat{a}_{i,t}x^t + ... + \hat{a}_{i,1}x + \hat{d}_i L$
   (b) generate a polynomial $r(x)$ with coefficients in $[-nL^2N^3..nL^2N^3]$ and constant term 0. Set $\hat{f}_n(i) \triangleq r(i)$, $1 \leq i \leq t$. Using Lemma 2 on input $\hat{f}_n(i)$ for $1 \leq i \leq t$ and $\hat{w}_n^L$ compute the values $g^{\hat{a}_{n,t}}, ..., g^{\hat{a}_{n,1}} \bmod N$

Signature Generation

1. (a) compute $\hat{\sigma}_i \triangleq m^{\hat{d}_i} \bmod N$ for $1 \leq i \leq n-1$
   (b) set $\hat{\sigma}_n \triangleq m^d / (\hat{\sigma}_{public} \prod_{i=1}^{n-1} \hat{\sigma}_i) \bmod N$
2. execute simulation of zero-knowledge proof as in [FGY96,GJKR96b]

Fig. 4. Simulator for Key Distribution and Signature Generation

1. That the sum of the sub-shares which P_i generated is the value of the share d_i. This verification is easily achieved by generating witness/commitments to the new additive sub-shares, and checking in the exponent that the distribution was done properly.
2. The second verification relates to the need to ensure that the back-up generated using the VSS is in fact the new additive share held by the player. Note that after the above mentioned verification we have a witness for the new additive-share (through some computation), this witness is of the form $w_i^{new} = g^{d_i^{new}} \bmod N$. Furthermore, when the VSS protocol is carried out for sharing a secret s, a by-product of this computation is the value $g^{sL} \bmod N$. Thus in order to verify that the player shared the correct value all that needs to be checked is that $g^{sL} = (w_i^{new})^L \bmod N$.

The share-refreshing protocol for the update period is presented in Figure 5.

Public information: element g, composite N
$w_i \bmod N$ for $1 \leq i \leq n$
Input of player P_i: secret share d_i such that $g^{d_i} = w_i \bmod N$

1. Player P_i randomly chooses values $d_{i,j} \in_R [-N^2..N^2]$ for $1 \leq j \leq n$, sets $d_{i,public} = d_i - \sum_{j=1}^{n} d_{i,j}$, computes and broadcasts $g_{i,j} \triangleq g^{d_{i,j}} \bmod N$.
2. Player P_i sends to player P_j the value $d_{i,j}$.
3. Verification of distribution of proper share size and public commitments: P_j verifies that $d_{i,j} \in [-N^2..N^2]$ and that $g_{i,j} = g^{d_{i,j}} \bmod N$ if not then he requests that $d_{i,j}$ be made public and set $g_{i,j}$ to g raised to this public value.
4. If P_i does not cooperate in Step 3 then his value d_i is reconstructed using the Reconstruction Phase (Figure 3).
5. Verification that the sub shares in fact sum up to the previous share of P_i: P_j verifies that $w_i = g_{i,public} \prod_{j=1}^{n} g_{i,j} \bmod N$, if not then player P_i's share d_i is reconstructed using the Reconstruction Phase.
6. P_i computes his new share $d_i^{new} = \sum_{j=1}^{n} d_{j,i}$ (note that the value $g^{d_i^{new}} \bmod N = \prod_{j=1}^{n} g^{d_{i,j}} \bmod N$ is already public), and shares it using Feldman-Z_N-VSS Sharing Phase. This results in a value $g^{sL} \bmod N$ where s is the secret that P_i shared.
7. If P_i fails to share his secret or $(g^{d_i^{new}})^L \neq g^{sL} \bmod N$ then each player P_j exposes $d_{j,i}$. If P_j fails to expose $d_{j,i}$ then d_j is reconstructed by all players.

Fig. 5. Share-refreshing Protocol

Theorem 3. *Under the assumption that factoring is intractable the protocol described in Figures 1, 2 and 5 is a constant round secure, robust, threshold proactive RSA scheme, in the presence of t malicious faults where the total number of players is $n \geq 2t + 1$.*

We omit the simulator and lemmas related to the size of the shares, and the fact that the repeated distribution of the signing key through additive shares does not reveal any statistically significant information. These proofs are similar in flavor to the ones appearing in [FGMY97a], and will appear in the full paper.

Acknowledgments

Special thanks go to Rosario Gennaro and Hugo Krawczyk for endless conversations on this topic. We thank Jessica Staddon for comments on an earlier version of the paper.

References

[BF97] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Crypto '97*, pages 425–439, 1997. Springer-Verlag. LNCS No. 1294.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations. In *Proc. 20th Annual Symp. on the Theory of Computing*, pages 1–10. ACM, 1988.

[Boy89] C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, editors, *Cryptography and Coding*, pages 241–246. Claredon Press, 1989.

[CGHN97] R. Canetti, R. Gennaro, A. Herzberg, and D. Naor. Proactive Security: Long-term Protextion Against Break-ins. *CryptoBytes*, 3(1):1–8, 1997.

[CH89] R. A. Croft and S. P. Harris. Public-key cryptography and re-usable shared secrets. In H. Baker and F. Piper, editors, *Cryptography and Coding*, pages 189–201. Claredon Press, 1989.

[CH94] R. Canetti and Amir Herzberg. Maintaining security in the presence of transient faults. *Crypto '94*, pages 425–438, 1994. Springer-Verlag. LNCS No. 839.

[CMI93] M. Cerecedo, T. Matsumoto, and H. Imai. Efficient and secure multiparty generation of digital signatures based on discrete logarithms. *IEICE Trans. Fundamentals*, E76-A(4):532–545, 1993.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proc. 26th Annual Symp. on the Theory of Computing*, pages 522–533. ACM, 1994.

[Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *Crypto '87*, pages 120–127, Berlin, 1987. Springer-Verlag. LNCS No. 293.

[Des94] Yvo G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July 1994.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology — Crypto '89*, pages 307–315, Berlin, 1989. Springer-Verlag. LNCS No. 435.

[DF91] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto '91*, pages 457–469, Berlin, 1991. Springer-Verlag. LNCS No. 576.

[DJ97] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Tech. Report ISSE-TR-97-01, George Mason University, July 1997. ftp://isse.gmu.edu/pub/techrep/97_01_ajodia.ps.gz.

[FD92] Yair Frankel and Yvo Desmedt. Parallel reliable threshold multisignature. TR-92-04-02, April, Dept. of EE and CS, U of Wisconsin, 1992.

[Fel87] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In *Proc. 28th Annual FOCS* , pages 427–437. IEEE, 1987.

[FGMY97a] Yair Frankel, P. Gemmell, P. Mackenzie, and M. Yung. Proactive RSA. In *Crypto '97*, pages 440–454, 1997. Springer-Verlag. LNCS No. 1294.

[FGMY97b] Y. Frankel, P. Gemmell, P. Mackenzie, and M. Yung. Optimal resilience proactive public-key cryptosystems. In *Proc. 38th FOCS* , pages 384–393. IEEE, 1997.

[FGY96] Y. Frankel, P. Gemmell, and M. Yung. Witness-based Cryptographic Program Checking and Robust Function Sharing. In *Proc. 28th STOC*, pages 499–508. ACM, 1996.

[FM88] P. Feldman and S. Micali. An Optimal Algorithm for Synchronous Byzantine Agreement. In *Proc. 20th STOC*, pages 148–161. ACM, 1988.

[Fra89] Y. Frankel. A practical protocol for large group oriented networks. In *Eurocrypt '89*, pages 56–61, 1989. Springer-Verlag. LNCS No. 434.

[GHY87] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *Crypto '87*, pages 135–155, 1987. Springer-Verlag. LNCS No. 293.

[GJKR96a] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Eurocrypt '96*, pages 354–371, 1996. Springer-Verlag. LNCS No. 1070.

[GJKR96b] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *Crypto '96*, pages 157–172, 1996. Springer-Verlag. LNCS No. 1109.

[Har94] L. Harn. Group oriented (t, n) digital signature scheme. *IEE Proc.-Comput. Digit. Tech*, 141(5):307–313, Sept 1994.

[HJJ+97] Amir Herzberg, M. Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *1997 ACM Conference on Computers and Communication Security*, 1997.

[JJKY95] M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive RSA for Constant-Size Thresholds. Upublished manuscript, 1995.

[OY91] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proc. 10th PODC* , pages 51–59. ACM, 1991.

A Proofs of Theorem and Lemmas from Section 4

Proof. **of Lemma 1**
Denote the set of players which the adversary has corrupted by P_{bad}, and w.l.o.g it is a set of maximal size, i.e. $|P_{bad}| = t$. We prove that with high probability there exists a sharing of r with a polynomial $r(x)$ which satisfies that for each player $P_i \in P_{bad}$ the share $f(i)$ received in the sharing of s is equal to the share received in the sharing of r, i.e. $r(i) = f(i)$. Furthermore, the coefficients of $r(x)$ are taken from the appropriate range.

Define a polynomial $h(x)$, such that $h(0) = (s - r)L$ and for all $P_i \in P_{bad}$ it holds that $h(i) = 0$. As we have defined the polynomial $h(x)$ in $t + 1$ points we can interpolate and compute its coefficients. It remains to be shown that the coefficients are integers and to compute their range.

$$h(x) = \sum_{P_i \in P_{bad} \cup \{0\}} h(i) \prod_{j \neq i, P_j \in P_{bad} \cup \{0\}} \frac{x - j}{i - j}$$

The only non-zero value of $h(x)$ is at evaluation point 0. Thus we have that the above is equal to $(s - r)L \prod_{P_j \in P_{bad}} \frac{x-j}{-j}$. The coefficient of x^i is:

$$\sum_{B \subset P_{bad}, |B| = i} \frac{(s - r)L}{\prod_{P_j \in P_{bad}} (-j)} \prod_{Pj \in B} (-j)$$

The value $L / \prod_{P_j \in P_{bad}} (-j)$ is an integer (note $L = n!$) hence the coefficient of x_i is an integer. Furthermore, the coefficient can be bounded (in absolute value) by

$$\sum_{B \subset P_{bad}, |B| = i} (s - r)L \leq (s - r)L \binom{t}{i} \leq \frac{(s - r)Lt!}{i!(t - i)!} \leq (s - r)Lt! \leq nL^2N^2$$

The desired polynomial $r(x)$ is $f(x)-h(x)$, its coefficients are integers and are in the range $[-(nL^2N^3+nL^2N^2)..nL^2N^3+nL^2N^2]$. Thus, the probability that the coefficients of $r(x)$ will not be in the right range is at most $t\frac{2nL^2N^2}{2(nL^2N^3+nL^2N^2)} = \frac{t}{N+1}$. □

Proof. **of Theorem 2**

Correctness & Robustness. We assume that the dealer carries out his protocol properly[12], hence $d = d_{public} + \sum_{i=1}^{n} d_i$, $w_i = g^{d_i} \bmod N$ and the VSS corresponding to player P_i in fact shares the value d_i. It is easy to see that the signature generation protocol computes the correct value.

Security. We give our proof by providing a simulator for the protocol, denoted $\mathcal{SIM}1$, which simulates both the key distribution protocol and the signature generation protocol. Fixing an adversary $\mathcal{A}$, the view of the adversary on execution of the protocol and its view on execution of the simulator are indistinguishable. W.l.o.g. assume that the adversary corrupts players $P_1, ..., P_t$[13]. We analyze the information viewed by the adversary which is generated by an execution of the protocol and the simulator.

Secret Key Distribution

1. $\hat{d}_i$ is a random value in $[-nN^2..nN^2]$ and so is d_i for $1 \leq i \leq t$. The value $\hat{d}_{public}$ is a random value in $[-n^2N^2..n^2N^2+N]$ where d_{public} is from $[-n^2N^2+d..n^2N^2+d]$ these distributions are statistically indistinguishable (details are clear).
2. (a) the values $\hat{w}_i$ are generated in the same manner as w_i and as the additive-shares have the same distribution so do the $\hat{w}_i$'s and w_i's for $1 \leq i \leq n-1$
   (b) as the distribution of $\hat{d}_{public}$ is indistinguishable from that of d_{public} so is the distribution of $g^{\hat{d}_{public}} \bmod N$ and $g^{d_{public}} \bmod N$.
   (c) it is easily argued that the distribution of $g^r \bmod N$ where $r \in [1..ord(g)]$ and the distribution of $g^s \bmod N$ where $s \in [-nN^2..nN^2]$ are statistically indistinguishable, hence $g^{\hat{d}_n}$ is indistinguishable from g^{d_n} (resp.).
3. (a) the simulator carries out exactly the same protocol as the true execution with input values which have the same distribution as the inputs of the protocol, hence the resulting information from the VSS protocol has the same distribution.
   (b) the argument for this step follows from Lemma 1.

Signature Generation

1. (a) follows from the argument on the distribution of $\hat{d}_i$.
   (b) same argument as in Step 2c
2. follows from the simulation proof of [FGY96,GJKR96a]

□

[12] We can relax this assumption by introducing more verification steps in the Key Distribution Phase.

[13] W.l.o.g that it is the first t players, and that it is no less than t.