Further Simplifications in Proactive RSA Signatures

Stanisław Jarecki and Nitesh Saxena

School of Information and Computer Science,
UC Irvine,
Irvine, CA 92697, USA
{stasio,nitesh}@ics.uci.edu

Abstract. We present a new robust proactive (and threshold) RSA signature scheme secure with the optimal threshold of $t < n/2$ corruptions. The new scheme offers a simpler alternative to the best previously known (static) proactive RSA scheme given by Tal Rabin [36], itself a simplification over the previous schemes given by Frankel et al. [18, 17]. The new scheme is conceptually simple because all the sharing and proactive re-sharing of the RSA secret key is done modulo a *prime*, while the reconstruction of the RSA signature employs an observation that the secret can be recovered from such sharing using a simple equation over the *integers*. This equation was first observed and utilized by Luo and Lu in a design of a simple and efficient proactive RSA scheme [31] which was not proven secure and which, alas, turned out to be completely *insecure* [29] due to the fact that the aforementioned equation leaks some partial information about the shared secret. Interestingly, this partial information leakage can be proven harmless once the polynomial sharing used by [31] is replaced by top-level *additive* sharing with second-level polynomial sharing for back-up.
Apart of conceptual simplicity and of new techniques of independent interests, efficiency-wise the new scheme gives a factor of 2 improvement in speed and share size in the general case, and almost a factor of 4 improvement for the common RSA public exponents 3, 17, or 65537, over the scheme of [36] *as analyzed in* [36]. However, we also present an improved security analysis and a generalization of the [36] scheme, which shows that this scheme remains secure for smaller share sizes, leading to the same factor of 2 or 4 improvements for that scheme as well.

1 Introduction

The idea of distributing a cryptosystem so that to secure it against corruption of some threshold, e.g. a minority, of participating players is known as *threshold cryptography*. It was introduced in the works of Desmedt [13], Boyd [4], Croft and Harris [9], and Desmedt and Frankel [14], which built on the *polynomial secret-sharing* technique of Shamir [39]. A threshold signature scheme [14] is an example of this idea. It allows a group of n players to share the private signature key in such a way that the signature key remains secret, and the signature scheme remains secure, as long as no more than t of the players are corrupt. Simultaneously, as long as at least $n - t$ players are honest, these players can efficiently produce correct signatures on any message even if the other t players act in an arbitrarily malicious way.

Proactive signature schemes [28, 27] are threshold signature schemes which offer an improved resistance against player corruptions. Time is divided into *update rounds*, and the proactive signature scheme offers the same combination of security and robustness even in the presence of so-called *mobile* faults [34], where a potentially new group of up to t players becomes corrupted in each update round. Technically, this is done by the players randomly re-sharing the shared private key at the beginning of each update round. A proactive signature scheme offers stronger security guarantee then a threshold scheme, especially in an application which might come under repeated attacks, like a certification authority or a timestamping service. Moreover, a proactive scheme offers more secure management of a system whose size and make-up need to change throughout its lifetime. Efficiency of the distributed signature protocol involved in a proactive signature scheme is very important in some applications, like in a timestamping service, or in the decentralized control of peer-to-peer groups, ad-hoc groups, or sensor networks [30, 37]. An efficient proactive scheme for RSA signatures is especially important because RSA signatures are widely used in practice, and because verification of RSA signatures is several orders of magnitude faster than verification of other signatures.

Prior Work on Threshold and Proactive RSA. While the work of Herzberg et al. [28, 27] and Gennaro et al. [25, 26] quickly yielded efficient secure proactive signature schemes for discrete-log based schemes like Schnorr [38] or DSS [33] signatures, the work on secure proactive RSA schemes progressed more slowly, and the initial threshold RSA scheme of Desmedt and Frankel [14] was robust only against crashes and not malicious faults, and had only heuristic security. The difficulty in adopting Shamir's polynomial secret-sharing technique to threshold RSA was caused by the fact that the RSA private key d is an element of a group $\mathbb{Z}_{\phi(N)}$, where $\phi(N)$ needs to remain hidden from all players because it allows immediate computation of the private key d from the RSA public exponent e. This difficulty was overcome by the schemes of Frankel et al. [16, 12] which provided a proof of security but used secret shares which were elements of a polynomial extension field of $\mathbb{Z}_n$, which increased the cost of the signature operation by a factor of at least t. These schemes were then extended to provide robustness against malicious faults by [19, 24]. Subsequently, Victor Shoup [40] presented a threshold RSA signature scheme which was robust and provably secure with optimal adversarial threshold $t < n/2$, and which did away with the extension field representation of the shares, thus making the cost of the signature operation for each participating player comparable to the standard RSA signature generation.

Proactive RSA scheme is a harder problem because it requires the players to re-share the private key d in each update round even if no single player is allowed to know the secret modulus $\phi(N)$. The first proactive RSA scheme of Frankel et al. [18] solved this problem using additive secret sharing over integers in conjunction with combinatorial techniques which divide the group of n players into two levels of families and sub-families. However, the resulting proactive protocol did not achieve optimal adversarial threshold $t < n/2$ and did not scale well with the group size n. These shortcomings were later overcome by the same authors [17], who showed that the RSA private key d can be shared over integers using polynomials with specially chosen large integer coefficients that simultaneously allowed interpolation without knowing $\phi(N)$ and unpredictability of the value d given any t polynomial shares. In this solution, even though the underlying secret sharing was polynomial, the players need to create a one-time additive sharing for every group of players participating in threshold signature generation. A simpler and more efficient proactive RSA scheme was then given by Tal Rabin [36]. Her solution also used sharing of the private key over integers, and employed shares of size about twice the length of the private key. The new idea was that the secret d was shared additively among the players, every share was backed-up by a secondary level of polynomial secret sharing, and the proactive update consisted of shuffling and re-sharing of the additive shares.

Limitations and Open Problems in Proactive RSA. The proactive RSA schemes of [18, 17, 36] leave at least two important problems unaddressed. While the new proactive RSA scheme we present in this paper does not solve these problems either, the techniques we present might help solve these problems in the future. The first problem is that of handling *adaptive* rather than *static* adversaries. The static adversary model assumes that the adversary decides which player to corrupt obliviously to the execution of the protocol, while the adaptive model allows the adversary to decide which player to corrupt based on his view of the protocol execution. This difference in the adversarial model is not known to be crucial for the security of the above protocols in practice. However, the above protocols are not known to be adaptively secure, while the known adaptively secure RSA schemes [20, 7, 21, 22] are significantly less efficient.

The second problem is that of requiring some form of additive rather than polynomial secret-sharing. The additive sharing implies that the shares of all temporarily unavailable players need to be reconstructed by the active players that participate in the signature generation protocol. This hurts both the efficiency and the resilience of a scheme in applications where one player might be temporarily unavailable to another without an actual corruption by the adversary. Since the threshold (but not proactive) RSA signature schemes discussed above do not resort to additive sharing, this is a disadvantage of the currently known proactive RSA schemes.

This somewhat unsatisfactory state of the known proactive RSA schemes led a group of network security researchers to design a proactive RSA scheme [30] which attempted to solve these problems by using polynomial secret sharing. The technique they employed was very simple. It relied on an observation that the secret sharing of the private key d modulo any modulus which has only large prime factors enables efficient reconstruction of d over *integers*. Consequently, by the homomorphic properties of exponentiation, it also enables reconstruction of the RSA signature $m^d \bmod N$. This scheme, however, did not come with a security proof, and indeed upon a closer examination [29], the proposed interpolation over the integers leaks a few most significant bits of the shared private key d, which together with the adversarial ability to manipulate the choice of shares in the proactive update protocol allows the threshold attacker to stage a binary search for d. Nevertheless, the above technique, which was utilized by the insecure scheme of [30], can be corrected, resulting in the *provably secure* proactive RSA scheme we present here.

Our Contribution: Further Simplification and Efficiency Improvements in Proactive RSA. Based on the corrected use of a technique discovered by Lu and Luo [31], we present a new robust and provably secure optimal-threshold proactive RSA scheme. Our scheme is known to be secure only in the static model, and it employs top-level additive sharing similarly as the Rabin's scheme [36], but it is interesting for the following reasons: (1) It is simpler than the previous schemes; (2) It offers factor of 2 improvement in share size and signature protocol efficiency for general RSA public keys, and factor of 4 improvement for the common case of public exponents like 3, 17, or 65537, over the most efficient previously known proactive RSA scheme [36] *as originally analyzed* by [36]; (3) The new scheme led us to a tighter security analysis of the [36] scheme, which resulted in similar, up to a logarithmic factor, efficiency improvements for the [36] scheme; (4) The new scheme offers an interesting case of a technique, invented by Lu and Luo [31], which makes a distributed protocol run faster but leaks some partial information about the shared secret. This partial information leakage led to an efficient key-recovery attack [29] on the original scheme of [31]. Yet, with some fixes, this partial information leakage can be provably neutralized and the same technique results in a provably secure scheme presented here; (5) Finally, our scheme offers new techniques which could aid in overcoming the two problems that still haunt proactive RSA solutions, namely achieving efficient adaptive security and the removal of additive sharing.

Paper Organization. Section 2 describes our adversarial model; section 3 presents the new scheme; section 4 contains the security proof; and section 5 shows an efficiency improvement for the proactive RSA scheme of [36].

2 Our Computation Model and the Adversarial Model

We work in the standard model of threshold cryptography and distributed algorithms known as synchronous, secure links, reliable broadcast, trusted dealer, static, and proactive adversary model. This is the same model as employed for example in [28, 27, 18, 17, 36] discussed in the introduction, with the exception that the first two did not need a trusted dealer (but did not handle RSA).

This model involves n players $M_1, ..., M_n$ equipped with synchronized clocks and an ability to erase information. The players are connected by weakly synchronous communication network offering secure point-to-point channels and a reliable broadcast. The time is apriori divided into evenly spaced update rounds, say of length of one day. We assume the presence of the so-called "mobile" adversary, modeled by a probabilistic polynomial time algorithm, who can *statically*, i.e., at the beginning of the life time of the scheme, schedule up to $t < n/2$ arbitrarily malicious faults among these n players, independently for every update round. We also assume a trusted dealer who initializes the distributed scheme by picking an RSA key and securely sharing the private key among the players. Since the adversary attacks a proactive *signature* scheme, the adversary can also stage a chosen-message attack [CMA], i.e. it can ask any of the n players to run a signature protocol on any message it chooses. The adversary's goal is to either (1) forge a signature on a message he did not request a signature on, exactly as in the CMA attack against a standard (non-threshold) signature scheme, or (2) to prevent the efficient generation of signatures on messages which at least $t+1$ uncorrupted players want to sign.

3 The New Proactive RSA Signature Scheme

3.1 Overview of the Proposed Scheme

The sharing of the private RSA key d is done additively modulo a *prime* q s.t. $q \geq r2^{|N|+\tau}$, where r is the maximal number of rounds in the lifetime of the system, $|N|$ is the bit length of the RSA modulus N, and τ is a security parameter, e.g. $\tau = 80$. Namely each player M_i holds a share d_i which is a random number in $\mathbb{Z}_q$ s.t. $d_1 + ... + d_n = d \bmod q$. Each of these top-level additive shares is also polynomially shared for backup reconstruction of d_i in case M_i is corrupted, using the information-theoretically secret verifiable secret sharing (VSS) of Pedersen [35], similarly as in the proactive RSA scheme of Rabin [36]. In order to handle the common case of a small public RSA exponent e more efficiently, the most significant $l = \frac{|N|}{2}$ bits of the private key d can be publicly revealed as d_{pub}, and only the remaining portion of the private key d, namely $d - 2^{|N|-l}d_{pub}$, is shared as above modulo q, for any $q \geq r2^{|N|-l+\tau}$.

The proactive update is very easy in this setting, adopting the original proactive secret sharing of [28] to additive sharing. Such method was used before e.g. in [7]. To re-randomize the the sharing, each M_i picks random partial shares d_{ij} in $\mathbb{Z}_q$ s.t. $d_i = d_{i1} + ... + d_{in} \bmod q$, and sends d_{ij} to M_j. Each M_j computes then his new share as $d'_j = d_{1j} + ... + d_{nj} \bmod q$, and shares it polynomially for backup again. All this can be easily verified using Pedersen's VSS, and the new shares sum to the same secret d modulo q.

For the threshold signature protocol, we use the observation of [31] that if $\sum_{j=1}^{n} d_j = d \pmod q$ and $0 \leq d_j \leq q-1$ for all j's, then

$$d = \sum_{j=1}^{n} d_j - \alpha q \quad \text{(over the integers)} \tag{1}$$

for some integer $\alpha \in \{0, ..., n-1\}$. Consequently, if $\forall_j,\ s_j = m^{d_j} \bmod N$ then

$$m^d = (\prod_{j=1}^{n} s_j) m^{-\alpha q} \pmod N$$

Therefore the signature $m^d \bmod N$ can be reconstructed if players submit their partial signatures as $s_j = m^{d_j} \pmod N$, and the correct value of α is publicly reconstructed by cycling over the possible n choices of α, which adds at most $2n$ modular exponentiations to the cost of the signature generation protocol. (Note that in most applications $n < 100$.) In the (rare) case of a malicious fault causing a failure in this procedure, each player has to prove in zero-knowledge that it used a correct value d_i in its partial signature, i.e. the value committed in the Pedersen VSS that shares this d_i. Efficient zero-knowledge proofs to handle such statement were given by Camenisch and Michels [5], and Boudot [2], and while not blazing fast, they have constant number of exponentiations, and they are practical. This procedure is more expensive than the robustness procedure in [36], but we believe that this efficiency difference does not matter since active corruptions of this type should be unlikely, as active faults are rare in general and the adversary would not gain much by using his corrupted player in this way.

In the attack [29] on a similar scheme involving *polynomial* rather than additive top-level sharing, the adversary uses the fact that the above procedure reveals whether d is greater or smaller than some value in the $[0, q]$ interval which the adversary can easily compute from his shares. Since the adversary can perfectly control his shares in the proactive update protocol for this (top-level) polynomial secret sharing scheme, the adversary can use this partial information leakage to stage a binary search for the shared secret d.

However, the scheme we present fixes the above problem. Assume that the adversary corrupts players $M_1, ..., M_t$. Giving the adversary the extra knowledge of shares $d_{t+1}, ..., d_{n-1}$, the only information about the secret key revealed by value α is, by equation (1), whether or not the secret d is smaller or larger than $R = (D \bmod q)$ where $D = d_1 + ... + d_{n-1}$. Since the adversary does not have enough control over the shares created by our "additive" proactive update protocol, shares $d_{t+1}, ..., d_{n-1}$ are random in $\mathbb{Z}_q$, and hence so is value R. Therefore, if q is significantly larger than the maximal value of d, then the α value almost never reveals anything about d, because d is almost always smaller than R. For this reason, if $q \geq r2^{|N|+\tau}$ then the modified scheme keeps d indistinguishable from a value uniform in $\mathbb{Z}_n$, with the statistical difference of $2^{-\tau}$. The additional factor r in the bound on q appears because of the linear increase in the statistical difference with every update round. This captures the security proof of our scheme in a nutshell.

We now give the detailed description of our scheme.

3.2 Setup Procedure

We require a trusted dealer to securely set up the system. The dealer generates RSA private/public key pair, i.e. an RSA modulus N, public exponent e, and private key $d = e^{-1} \bmod \phi(N)$. Optionally, $l \leq \frac{|N|}{2}$ most significant bits of d can be publicly revealed as d_{pub} (otherwise $d_{pub} = 0$ and $l = 0$). The dealer also chooses an instance of Pedersen commitment [35], i.e. primes p and q s.t. $q|(p-1)$, and two random elements g, h of order q in $\mathbb{Z}_p^*$, for $|q| = \log_2 r + |N| - l + \tau + 1$, where τ is a security parameter ($\tau \geq 80$) and r is the number of rounds the system is expected to run. The dealer then runs the sharing protocol of Figure 1.

Input: private key $d \in \mathbb{Z}_{\phi(n)}$, public value d_{pub} corresponding to l MSBs of d, public RSA modulus N, Pedersen commitment instance (p, q, g, h).

1. Select shares $d_j \in \mathbb{Z}_q$ uniformly at random for $j = 1, \ldots, n-1$ and set $d_n = d - 2^{|N|-l} d_{pub} - \sum_{j=1}^{n-1} d_j \pmod q$.
2. Share each d_j using Pedersen's VSS protocol [35]. Namely, select random polynomials $f_j(z) = d_j + f_{j1}z + \cdots + f_{jt}z^t$ and $f'_j(z) = d'_j + f'_{j1}z + \cdots + f'_{jt}z^t$ over $\mathbb{Z}_q$ of degree t s.t. $f_j(0) = d_j$. Compute and publish the witnesses $w_{j0} = g^{d_j} h^{d'_j} \pmod p$ and $w_{jk} = g^{f_{jk}} h^{f'_{jk}} \pmod p$ for $k = 1, \ldots, t$.
3. Compute the secret shares ss_{ij} and ss'_{ij} as $ss_{ij} = f_j(i) \pmod q$ and $ss'_{ij} = f'_j(i)$, deliver d_i, d'_i, ss_{ij} and ss'_{ij} $(\forall j)$ to each M_i over a secure channel.

Fig. 1. Trusted Dealer's Protocol: Sharing of the Private Key d

3.3 Threshold Signature Protocol

The goal of the threshold RSA signature protocol is to generate in a distributed manner an RSA signature $s = m^d \pmod N$ under the secret-shared key d, where $m \in \mathbb{Z}_n^*$ is some hashed/padded function of the signed message, e.g. $m = H(M)$ for the Full Domain Hash RSA [1]. Our protocol consists of two parts. First each player M_j creates its *partial signature* on the intended message $s_j = m^{d_j} \bmod N$, and sends it to the signature recipient. The recipient then locally reconstructs the RSA signature from these partial signatures using the n-bounded reconstruction algorithm of [31]. The threshold signature generation and reconstruction protocol is summarized in Figure 2, and we explain the details of the reconstruction algorithm below.

Signature Reconstruction with n-Bounded Offsetting. On receiving n partial signatures s_j from the n players, the signature recipient reconstructs the RSA signature s using the *n-bounded-offsetting* algorithm [30] which works as follows. Since $\sum_{j=1}^{n} d_j = d - 2^{|N|-l} d_{pub} \pmod q$ and $0 \leq d_j \leq q-1$ for all j's, therefore

$$d = 2^{|N|-l} d_{pub} + \sum_{j=1}^{n} d_j - \alpha q \quad \text{(over the integers)} \tag{2}$$

Input: (hashed) message $m \in \mathbb{Z}_n^*$, outputs of the Setup procedure

1. Player M_i broadcasts its partial signature $s_i = m^{d_i} \pmod N$.
2. If M_i fails to provide its partial signature, all players reconstruct d_i and compute $s_i = m^{d_i} \pmod N$.
3. Reconstruct RSA signature using the *n-bounded offsetting* algorithm (see below).
4. If signature reconstruction fails, trace the faulty signer(s) by executing the protocol $ZKPK(d_i : w_{i0} = g^{d_i} h^{d'_i} \pmod p) \wedge s_i = m^{d_i} \pmod N)) \wedge d_i \in [0, q-1]))$ with each M_i (see Appendix).
5. If M_i fails this proof, any set of $t+1$ players reconstruct d_i and compute and broadcast $s_i = m^{d_i} \pmod N$.

Fig. 2. Signature Generation and Reconstruction

for some integer $\alpha \in \{0, \ldots, n-1\}$, which implies that

$$s = m^d = m^{2^{|N|-l} d_{pub}} (\prod_{j=1}^{n} s_j) m^{-\alpha q} \pmod N$$

Since there can be at most n possible values of α, the signature recipient can recover $s = m^d \pmod N$ by trying each of the n possible values $Y_\alpha = Y(m^{-q})^\alpha \pmod N$ for $Y = m^{2^{|N|-l} d_{pub}} (\prod_{j=1}^{n} s_j)$ and $\alpha = 0, ..., n-1$, and returning $s = Y_\alpha$ if $(Y_\alpha)^e = m \pmod N$. The decisive factor in the cost of this procedure is the cost of the full exponentiation $m^q \bmod N$, where q can be e.g. 613-bit long for $N = 1024$, $e = 3$, $l = |N|/2$, $\tau = 80$, and $r \leq 2^{20}$.

As discussed in the overview subsection above, this procedure reveals value α which contains some partial information on the shared secret d. Namely, granting to the adversary some extra knowledge and assuming he knows shares $d_1, ..., d_{n-1}$, the α value reveals whether $d \in \mathbb{Z}_{\phi(n)}$ lies in the interval $[0, R[$ or in $[R, N]$, where $R = (D \bmod q)$ and $D = d_1 + \ldots + d_{n-1}$, if $l = 0$. More generally, α reveals if d is smaller or larger than $R + 2^{|N|-l} d_{pub}$.

Robustness Mechanisms. In case some player M_u does not issue a partial signature, share d_u of M_u needs to be reconstructed to recover partial signature $s_u = m^{d_u} \pmod N$. In reconstruct d_u, every player M_i broadcasts its shares ss_{iu}, ss'_{iu} of d_u. The validity of these shares can be ascertained by checking

$$g^{ss_{iu}} h^{ss'_{iu}} = \prod_{k=0}^{t} (w_{uk})^{i^k} \pmod p.$$

Share d_u can then be recovered using the interpolation

$$d_u = \sum_{j \in G} ss_{ju} l_j(u) \pmod q$$

where G is a subgroup of $t+1$ players who broadcast valid shares and $l_j(u) = \prod_{j \in G, j \neq i} \frac{(u-j)}{i-j} \bmod q$ is the Lagrange interpolation polynomial computed at u.

If all the partial signatures are present but the above n-bounded signature reconstruction algorithm fails, then at least one out of n players did not issue a correct partial signature. The signature recipient must then trace the faulty players(s) by verifying the correctness of each partial signature. Once a player is detected as faulty, the share(s) of the faulty player(s) can be reconstructed as above. To prove correctness of its partial signature, each M_i proves in zero-knowledge that there is a pair of integers (d_i, d'_i) s.t.

$$w_{i0} = g^{d_i} h^{d'_i} \bmod p \quad , \quad s_i = m^{d_i} \bmod N \quad , \quad 0 \leq d_i < q$$

It is crucial that the range of d_i is checked because otherwise player M_i can submit its partial signature as $m^{d'_i} \bmod N$ where $d'_i = d_i + kq$ for some k. An efficient zero-knowledge proof system for the proof of equality of discrete logarithms (and representations) in two different groups was given in [3, 6], and the efficient proof that a committed number lies in a given range appeared in [2]. The resulting ZKPK proof system is in the appendix. It is non-interactive in the random oracle model and involves a (small) constant amount of exponentiations.

3.4 Proactive Update Protocol

At the beginning of every update round, the players perform the share update protocol of Figure 3 to re-randomize the sharing of d.

Input: Outputs of the Setup procedure or the previous Update protocol.
Let $r \geq 1$ be the round number. Denote current values $d_i^{(r-1)}, d_i'^{(r-1)}, w_{ij}^{(r-1)}$, etc.

1. Each player M_i selects (sub)shares d_{ij} and $d'_{ij} \in \mathbb{Z}_q$, uniformly at random for $j = 1, \ldots, n-1$, and sets $d_{in} = d_i^{(r-1)} - \sum_{k=1}^{n-1} d_{ik} \pmod q$ and $d'_{in} = d_i'^{(r-1)} - \sum_{k=1}^{n-1} d'_{ik} \pmod q$. M_i broadcasts witness values $w_{ij}^{(r)} = g^{d_{ij}} h^{d'_{ij}} \pmod p$, and hands (d_{ij}, d'_{ij}) to M_j $(\forall j)$ over a secure channel.
2. M_j verifies the validity of the received shares using witness values as $w_{ij}^{(r)} = g^{d_{ij}} h^{d'_{ij}} \pmod p$, and ascertains whether the sub-shares in fact sum up to the previous share of M_i by checking that $\prod_{j=1}^{n} w_{ij}^{(r)} = w_{i0}^{(r-1)} \pmod p$.
3. M_j computes its new additive shares as $d_j^{(r)} = \sum_{i=1}^{n} d_{ij} \pmod q$ and $d_j'^{(r)} = \sum_{i=1}^{n} d'_{ij} \pmod q$. (Note that $\sum_{j=1}^{n} d_j^{(r)} = d - 2^{|N|-l} d_{pub} \pmod q$.)
4. M_j shares its new additive shares $d_j^{(r)}, d_j'^{(r)}$ using Pedersen's VSS, as in the setup phase described in Section 3.2. In order to check if M_j is indeed sharing its new additive share, every player checks that the witness value in this VSS instance corresponding to the shares $d_j^{(r)}, d_j'^{(r)}$ equals to $\prod_{i=1}^{n} w_{ij}^{(r)} \pmod p$.

Fig. 3. Proactive Share Update

4 Security Analysis of the New Proactive RSA Scheme

Theorem 1 (Security) *If there is a t-threshold proactive adversary for $t < n/2$, which in time T succeeds with probability β in a chosen-message attack against our new proactive (full domain hash) RSA signature scheme running for up to r rounds, for any $l \leq |N|$ and prime $q \geq r2^{|N|-l+\tau}$, then there is a CMA attack against the standard (full domain hash) RSA signature scheme, which succeeds in time $T + poly(n, |N|)$ with probability $\beta - 2^{-\tau}$ given the l most significant bits of the secret key d as an additional public input.*

Proof. We show that if the adversary succeeds in staging the CMA attack on our (Full Domain Hash) proactive RSA signature scheme in time T with probability β, then there is also an efficient CMA attack against the standard (non-threshold) FDH-RSA signature which given the l most significant bits of d succeeds in time comparable to T by an amount polynomial in $|N|$ and n, with probability no worse than $\beta - 2^{-\tau}$. We show it by exhibiting a very simple simulator, which the adversary against the standard FDH-RSA scheme can run to interact with the proactive adversary which (T, β)-succeeds in attacking the proactive scheme. We will argue that the statistical difference between the view presented by this simulator on input of the public RSA parameters, l MSBs of d, and (message,signature) pairs acquired by the CMA attacker from the CMA signature oracle, and the adversarial view of the run of the real protocol on these parameters, for any value of the private key d with these l most significant bits, is at most $2^{-\tau}$, which will complete the proof.

The simulator $\mathcal{SIM}$ is described in Figure 4. The simulation procedure is very simple. The simulator picks a random value $\hat{d}$ in $\mathbb{Z}_n$ with the given l most-significant bits, and runs the secret-sharing protocol in the setup stage using this $\hat{d}$. Similarly in every update, the simulator just runs the actual protocol, but on the simulated values which we denote $\hat{d}_i$, $\hat{d}_{ij}$, etc. The only deviation from the protocol is that in the simulation of the threshold signature protocol, assuming w.l.o.g. that the M_n is an uncorrupted player, the simulator runs the actual protocol for all uncorrupted players except of M_n, i.e. it outputs $\hat{s}_j = m^{\hat{d}_j}$ for each uncorrupted M_j, $j \neq n$. The simulator then determines the $\hat{\alpha}$ value, which is an approximation to the actual value α the adversary would see in the protocol, by computing $D = \sum_{j=1}^{n-1} \hat{d}_j$, and taking $\hat{\alpha} = \lfloor D/q \rfloor + 1$. In this way we have $D = (\hat{\alpha} - 1)q + R$ where $R = (D \bmod q)$. Finally, the simulator computes the missing partial signature $\hat{s}_n$ corresponding to the player M_n as $\hat{s}_n = s * m^{\hat{\alpha}q} / (m^{2^{|N|-l} d_{pub}} \prod_{j=1}^{n-1} \hat{s}_j) \pmod{N}$. In this way, partial signatures $\hat{s}_j$ add up to a valid RSA signature $\hat{s}$, and value $\hat{\alpha}$ the adversary sees in the *simulation* of the signature reconstruction algorithm is equal to the above α with an overwhelmingly high probability.

For ease of the argument, assume that the adversary corrupts players $M_1, ..., M_t$ throughout the lifetime of the scheme. We will argue that the adversarial views of the protocol and the simulation are indistinguishable with the statistical difference no more than $2^{-\tau}$, even if the adversary additionally sees shares $d_{t+1}, ..., d_{n-1}$ *and* the shared secret key d.

Input: Pedersen commitment instance (p, q, g, h), RSA public parameters (N, e), optional values $l > 0$ and $d_{pub} < 2^l$ (otherwise set $l = d_{pub} = 0$).
Additionally, for every simulation of the threshold signature protocol, the simulator gets pair (m, s) s.t. $s = m^d \bmod N$.

Setup Procedure
Pick random $\hat{d} \in \mathbb{Z}_n$ and proceed as in the Setup of the actual protocol:

1. Select random shares $\hat{d}_j, \hat{d}'_j \in \mathbb{Z}_q$, for $j = 1, \ldots, n-1$, and set $\hat{d}_n = \hat{d} - 2^{|N|-l} d_{pub} - \sum_{i=1}^{n-1} \hat{d}_j \pmod q$, as in step 1 of the Setup procedure.
2. Share each $\hat{d}_j$ and $\hat{d}'_j$ using the Pedersen's VSS: Choose random polynomials $\hat{f}_j(z) = \hat{d}_j + \hat{f}_{j1} z + \cdots + \hat{f}_{jt} z^t$ and $\hat{f}'_j(z) = \hat{d}'_j + \hat{f}'_{j1} z + \cdots + \hat{f}'_{jt} z^t$ over $\mathbb{Z}_q$ of degree t; compute and publish the witnesses $\hat{w}_{j0} = g^{\hat{d}_j} h^{\hat{d}'_j} \pmod p$ and $\hat{w}_{jk} = g^{\hat{f}_{jk}} h^{\hat{f}'_{jk}} \pmod p$ for $k = 1, \ldots, t$.
3. Compute the secret shares $\hat{s}s_{ij}$ and $\hat{s}s'_{ij}$ as $\hat{s}s_{ij} = \hat{f}_j(i) \pmod q$ and $\hat{s}s'_{ij} = \hat{f}'_j(i)$ and distribute $\hat{d}_i, \hat{d}'_i, \hat{s}s_{ij}$ and $\hat{s}s'_{ij}$ $(\forall j)$ to each M_i over a secure channel.

Threshold Signature Protocol (on additional input (m, s)):

1. Generate partial signatures $\hat{s}_i$ for $i = 1, \ldots, n-1$ as $\hat{s}_i = m^{\hat{d}_i} \pmod N$. Compute $D = \hat{d}_1 + \ldots + \hat{d}_{n-1}$, and $\hat{\alpha} = \lfloor D/q \rfloor + 1$. Compute $\hat{s}_n = s * m^{\hat{\alpha} q} / (m^{2^{|N|-l} d_{pub}} \prod_{j=1}^{n-1} \hat{s}_j) \pmod N$.
2. Output values $\hat{s}_i$ on behalf of the uncorrupted players M_i.
3. If needed, execute the ZKPK proof for $M_i \neq M_n$, and simulate it for M_n.

Proactive Update
Proceed in exactly the same manner as the Proactive Update protocol:

1. At the beginning of round r, for all uncorrupted players M_i, select (sub)shares $\hat{d}_{ij}$ and $\hat{d}'_{ij}$ uniformly in $\mathbb{Z}_q$ for $j = 1, \ldots, n-1$, and set $\hat{d}_{in} = \hat{d}_i^{(r-1)} - \sum_{k=1}^{n-1} \hat{d}_{ik} \pmod q$ and $\hat{d}'_{in} = \hat{d}_i'^{(r-1)} - \sum_{k=1}^{n-1} \hat{d}'_{ik} \pmod q$. Broadcast witness values $\hat{w}_{ij}^{(r)} = g^{\hat{d}_{ij}} h^{\hat{d}'_{ij}} \pmod p$, and hand $(\hat{d}_{ij}, \hat{d}'_{ij})$ to M_j $(\forall j)$ over a secure channel.
2. Compute M_j's new secret shares $\hat{d}_j^{(r)} = \sum_{i=1}^{n} \hat{d}_{ij} \pmod q$ and $\hat{d}_j'^{(r)} = \sum_{i=1}^{n} \hat{d}'_{ij} \pmod q$, as in the Proactive Update protocol.
3. Re-share the new additive share $\hat{d}_j^{(r)}, \hat{d}_j'^{(r)}$ using Pedersen's VSS, as in the Proactive Update protocol.

Fig. 4. Simulator Construction ($\mathcal{SIM}$)

Setup Procedure: Since d_i and d'_i in the protocol and $\hat{d}_i$ and $\hat{d}'_i$ in the simulation are all picked uniformly from $\mathbb{Z}_q$ for $i = 1, \ldots, n-1$, the two ensembles $(d, \{d_i, d'_i\}_{i=1,\ldots,n-1})$ and $(d, \{\hat{d}_i, \hat{d}'_i\}_{i=1,\ldots,n-1})$ have identical distributions.

By the information theoretic secrecy of Pedersen VSS, the second-layer shares and the associated verification values visible to the adversary are also distributed identically in the protocol and in the simulation.

Threshold Signature Protocol: Since d_i and $\hat{d}_i$, for $i = 1, \ldots, n-1$, have the identical distributions, therefore distributions of the corresponding partial signatures s_i and $\hat{s}_i$, are also identical. However, values s_n and $\hat{s}_n$ are the same only in the event that value α in the protocol and value $\hat{\alpha}$ in the simulation are the same. Recall that $\hat{\alpha}$ in the simulation is computed as $\hat{\alpha} = \lfloor D/q \rfloor + 1$ where $D = \sum_{j=1}^{n-1} \hat{d}_j$. Note that $D = (\hat{\alpha} - 1)q + R$ where $R = (D \bmod q)$. By equation (2), value α computed by the protocol would satisfy equation

$$d = 2^{|N|-l} d_{pub} + D + d_n - \alpha q = 2^{|N|-l} d_{pub} + R + d_n + (\hat{\alpha} - \alpha - 1)q$$

because $d_1, \ldots, d_{n-1}$ are distributed identically to $\hat{d}_1, \ldots, \hat{d}_{n-1}$.

Since d_n and R are elements in $\mathbb{Z}_q$ for $q \geq 2^{|N|-l+\tau+\log r}$, and since $d \in [2^{|N|-l} d_{pub}, 2^{|N|-l} d_{pub} + 2^{|N|-l}]$, the above equation implies that there are only two possible cases: $\alpha = \hat{\alpha} - 1$ and $\alpha = \hat{\alpha}$. The first case happens if $d \geq 2^{|N|-l} d_{pub} + R$ and the second if $d < 2^{|N|-l} d_{pub} + R$. However, the probability that $d < 2^{|N|-l} d_{pub} + R$, and hence that $\alpha = \hat{\alpha}$, is at least $1 - 2^{-(\tau+\log r)}$ because the probability of the other case is at most the probability that R is less than $2^{|N|-l}$, which, given that R is a uniformly distributed element in $[0, q]$, is at most $2^{-(\tau+\log r)}$.

Note that value α stays the same in all instances of the threshold signature protocol in any given update round. Since the same holds for the $\hat{\alpha}$ value in the simulation, the probability that the adversary's view of all these protocol instances is different from the view of all the simulation instances remains at most $2^{-(\tau+\log r)}$. In other words, the statistical difference between the adversary's view of the real execution and the simulation in any update round, is at most $(1/r)2^{-\tau}$.

Proactive Update Protocol: Since values $\{d_i\}_{i=1..n-1}$ and $\{\hat{d}_i\}_{i=1..n-1}$ are distributed identically, the only difference in the execution and the simulation of the update protocol can come from sharing of the d_n value in the protocol and $\hat{d}_n$ in the simulation. However, since this sharing is a "additive" equivalent of Pedersen VSS, and the second-layer sharing of the shares of the d_n or $\hat{d}_n$ value is done with Pedersen VSS too, the whole protocol hides the shared value d_n perfectly, and hence the adversarial view in the simulation of the update protocol is identical to the adversarial view of the actual protocol.

Since the statistical difference between the protocol and the simulation is zero in the setup stage and in any proactive update stage, and at most $(1/r)2^{-\tau}$ in any single update round, given r rounds the overall difference between adversarial view of the protocol execution and its simulation is at most $2^{-\tau}$, which completes our argument.

Theorem 2 (Robustness) *Under the Discrete Logarithm and Strong RSA assumptions, our proactive signature scheme is robust against a t-threshold proactive adversary for $t < n/2$.*

Proof. Note that the only way robustness can be broken is if some malicious player M_i cheats either in the proactive update protocol, by re-sharing a value different than its proper current share d_i committed in Pedersen commitment $w_i = g^{d_i} h^{d_i} \bmod p$, or M_i cheats in the signature protocol, by proving correct the wrong partial signature $s_i \neq m^{d_i} \bmod N$. Since the first type of cheating is infeasible under the discrete logarithm assumption and the second type is infeasible under the strong RSA assumption, the claim follows.

4.1 Security Implications

Taking $l = 0$, Theorem 1 implies that the new proactive signature scheme is as secure as the standard RSA:

Corollary 1. *Under the RSA assumption in the Random Oracle Model, our scheme is a secure t-threshold proactive signature, for $l = 0$ and $q \geq r2^{|N|+80}$.*

On the other hand, note that the RSA adversary can always correctly guess the most significant half of the bits of d with probability $1/(e-1)$.[1] Together with theorem 1, this implies the following corollary:

Corollary 2. *Under the RSA assumption (in the Random Oracle Model), the time T_{PRSA} to break the new proactive signature scheme for $e = 2^i+1$, $l = |N|/2$ and $q \geq r2^{|N|/2+80}$, is at least $T_{PRSA} \geq 2^{-i} T_{RSA}$, where T_{RSA} is the time required to break the CMA security of the standard (FDH) RSA signature scheme for modulus of length $|N|$.*

For the most popular value of $e = 3$, this implies that if the 1024-bit modulus RSA has a 2^{80} security then our proactive RSA scheme running on the same modulus for $l = 512$ and $q \geq r2^{512+80}$ would have at least 2^{79} security. For $e = 17$ the provable security would be 2^{76}. Of course, our scheme could be executed with slightly larger N to compensate for the 2^i factor in security degradation, but with key shares sizes still limited by $q < r2^{|N|/2+80}$. The efficiency of the resulting schemes resulting from Corollary 2 should be compared with the straightforward settings implied by Corollary 1, where same 2^{80} security is given by 1024 bit N but with larger bound of $r2^{|N|+80}$ on the share size q.

However, since there are no known attacks against RSA which speed up the factorization of N when half of the most significant bits of d are revealed for small values of e, it can be plausibly hypothesized that for small e's, the proposed

[1] Note that $ed = 1 \pmod{\phi(N)}$ implies that $d = 1/e(1 + k\phi(N))$ for some integer $k = 1, ..., e-1$. Therefore, since $N - \phi(N) < \sqrt{N}$, it follows that $0 \leq \hat{d}_k - d < \sqrt{N}$ for $\hat{d}_k = \lfloor 1/e(1+kN) \rfloor$ for one of the $e-1$ choices of k. Thus any adversary facing the RSA cryptosystem can with probability $1/(e-1)$ guess the $|N|/2$ most significant bits of d by picking the right k and computing $\hat{d}_k$ as above.

proactive RSA scheme remains as secure as standard RSA for the same modulus size even with half of the most significant bits of d are revealed.

Finally we remark that the security analysis of our scheme given in Theorem 1 grants the adversary the knowledge of $n-1$ shares instead of just t shares he can see in the protocol, which suggests that our security analysis can be improved and that our scheme is possibly secure using smaller share sizes than our analysis recommends.

5 Improved Security Analysis of Rabin's Proactive RSA

Overview of Proactive RSA Scheme of [36]. During the setup, a trusted dealer generates the RSA public (N, e) and private $(d, \hat{p}, \hat{q})$ key pairs. The signature key d is shared additively among the players. Each M_i gets a share d_i, chosen uniformly in $[-R, R]$ where $R = nN^2$, and the dealer publishes public value d_{public} such that

$$d_{public} = d - \sum_{i=1}^{n} d_i \quad (\text{over } \mathbb{Z}) \tag{3}$$

This can be easily extended, so that like our new scheme, l most significant bits of d are publicly revealed and added to the d_{pub} value, and only the remaining $(|N|-l)$-bit value $d - 2^{|N|-l}d_{pub}$ is shared as above. The witness value $w_i = g^{d_i}$ $\pmod N$ corresponding to each d_i is published, where g is an element of high order in $\mathbb{Z}_n^*$. Each share d_i is then itself shared using the Feldman VSS [15] over $\mathbb{Z}_n$. To sign a message m, each player M_i, generates a partial signature $s_i = m^{d_i}$ $\pmod N$. Since the signature key d is shared over integers, the RSA signature can be easily reconstructed by simply multiplying n partial signatures, i.e.,

$$s = m^{d_{public}} \prod_{i=1}^{n} s_i \pmod N$$

The detection of faults during the signing process can be performed using the protocols of [24, 19]. The secret share of the faulty player is then reconstructed by pooling in the shares of any $t+1$ players using a special variant of polynomial interpolation (refer to [36] for details). In the share update protocol each M_i additively re-shares its secret share d_i with (sub)shares $d_{ij} \in [-R/n, R/n]$ and

$$d_{i,public} = d_i - \sum_{j=1}^{n} d_{ij} \ (\text{over } \mathbb{Z})$$

is made a public value. The new secret share for $d_i^{(r)}$ of M_i is then computed as $d_i^{(r)} = \sum_{j=1}^{n} d_{ji}$, and M_i shares it using Feldman VSS over $\mathbb{Z}_n$.

Improved Security Analysis and Improved Performance. First, we note that the simulator for the setup phase presented in [36] has a small error. That simulator for the key distribution protocol picks random shares $\hat{d}_i \in [-R, R]$, for $i =$

$1, \ldots, n-1$, and it picks $\hat{d}_{public}$ *uniformly* at random in $[-nR, nR+N]$. However, values generated in this way are not statistically indistinguishable from the values in the protocol, because if the d_i values are chosen uniformly in $[-R, R]$, then by equation (3), value d_{public} has a normal probability distribution, which is immediately *distinguishable* from the uniform distribution of $\hat{d}_{public}$.

The corrected simulation of the key distribution (and the subsequent update protocols) works exactly in the same manner as the actual protocol. The simulator should choose some secret value $\hat{d} \in [0, N-1]$ at random, and share this new value in exactly the same manner as in the protocol. After r update rounds, the overall statistical difference between the view of the adversary interacting with the protocol and the view of the adversary interacting with the (new) simulator is at most rN/R. This difference is negligible if $R = rN2^{\tau}$, where $\tau \geq 80$, instead of the $R = nN^2$ value recommended in [36].

This shows that secret shares can be picked from range $[-rN2^{\tau}, rN2^{\tau}]$, instead of range $[-nN^2, nN^2]$ of the original scheme, which means an almost factor of 2 improvement in the share size. Since the computational cost of this scheme is driven by cost of the exponentiation $s_i = m^{d_i} \bmod N$ done by each player, factor of 2 improvement in the size of d_i speeds the signature generation by the same factor.

References

1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
2. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 431–444, 2000.
3. F. Boudot and J. Traor. Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery. In *Second International Conference on Information and Communication Security (ICICS)*, pages 87–102, November 1999.
4. C. Boyd. Digital multisignatures. In *Cryptography and Coding*, pages 241–246. Claredon Press, May 1989.
5. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 107–122, 1999.
6. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 106–121, 1999.
7. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115, 1999.
8. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. In *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 561–575, 1998.
9. R. Croft and S. Harris. Public-key cryptography and re-usable shared secrets. In *Cryptography and Coding*, pages 189–201. Claredon Press, May 1989.
10. I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 418–430, 2000.

11. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT'02*, volume 2501 of *LNCS*, pages 125–142. Springer, 2002.
12. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 522–533, Montreal, Canada, 1994.
13. Y. Desmedt. Society and Group Oriented Cryptosystems. In *CRYPTO '87*, number 293 in LNCS, pages 120–127, 1987.
14. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO '89*, number 435 in LNCS, pages 307–315, 1990.
15. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Symposium on Foundations of Computer Science (FOCS)*, pages 427–437, 1987.
16. Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Technical Report TR-92-04-02, Dept. of EE and CS, U. of Winsconsin, April 1992.
17. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal-resilience proactive public-key cryptosystems. In *38th Symposium on Foundations of Computer Science (FOCS)*, pages 384–393, 1997.
18. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In *Crypto'97*, volume 1294 of *LNCS*, pages 440–454, 1997.
19. Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking and robust function sharing. In *Proc. 28th ACM Symp. on Theory of Computing*, pages 499–508, Philadelphia, 1996.
20. Y. Frankel, P. MacKenzie, and M. Yung. Adaptively-secure distributed threshold public key systems. In *Proceedings of ESA 99*, 1999.
21. Y. Frankel, P. MacKenzie, and M. Yung. Adaptively-secure optimal-resilience proactive RSA. In *ASIACRYPT'99*, volume 1716 of *LNCS*, 1999.
22. Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptive security for the additive-sharing based proactive rsa. In *Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 240–263, 2001.
23. E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *CRYPTO '97*, volume 1294 of *LNCS*, pages 16–30, 1997.
24. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. In *CRYPTO '96*, volume 1109 of *LNCS*, pages 157–172, 1996.
25. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. In *EUROCRYPT '96*, number 1070 in LNCS, pages 354–371, 1996.
26. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete log based cryptosystems. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310, 1999.
27. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *ACM Conference on Computers and Communication Security*, pages 100–110, 1997.
28. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing, or how to cope with perpetual leakage. In *CRYPTO '95*, volume 963 of *LNCS*, pages 339–352, 1995.
29. S. Jarecki, N. Saxena, and J. H. Yi. An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 1–9, October 2004.

30. J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing Robust and Ubiquitous Security Support for MANET. In *IEEE 9th International Conference on Network Protocols (ICNP)*, pages 251–260, 2001.
31. H. Luo and S. Lu. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks. Technical Report TR-200030, Dept. of Computer Science, UCLA, 2000. Available online at http://citeseer.ist.psu.edu/luo00ubiquitous.html.
32. D. Micciancio and E. Petrank. Simulatable Commitments and Efficient Concurrent Zero-Knowledge. In *EUROCRYPT'03*, volume 2656 of *LNCS*, pages 140–159, 2003.
33. NIST. Digital signature standard (DSS). Technical Report 169. National Institute for Standards and Technology, August 30, 1991.
34. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *10th ACM Symp. on the Princ. of Distr. Comp.*, pages 51–61, 1991.
35. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Crypto 91*, volume 576 of *LNCS*, pages 129–140, 1991.
36. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 89 – 104, 1998.
37. N. Saxena, G. Tsudik, and J. H. Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 104–114, October 2003.
38. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
39. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
40. V. Shoup. Practical Threshold Signatures. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 207–220, 2000.

A Zero Knowledge Proof of Partial Signature Correctness

For the purpose of proving the correctness of partial signatures in the proposed proactive RSA scheme, we apply the zero knowledge proofs for the equality of committed numbers in two different groups and for the range of a committed number. All these proofs are honest verifier zero-knowledge and can be converted either into standard zero-knowledge proof either at the expense of 1-2 extra rounds using techniques of [10, 11, 32], or into a non-interactive proof in the random oracle model using the Fiat-Shamir heuristic. We adopt the notation of [5] for representing zero-knowledge proof of knowledge protocols. For example, $\boxed{ZKPK\{x : R(x)\}}$ represents a ZKPK protocol for proving possession of a secret x which satisfies statement $R(x)$. In the protocols to follow, u (≥ 80) and v (≥ 40) are security parameters.

Protocol for proving the correctness of a partial signature:

$$\boxed{ZKPK\{d_i, d'_i : w_{i0} = g^{d_i} h^{d'_i} \pmod{p} \;\wedge\; s_i = m^{d_i} \pmod{N} \;\wedge\; d_i \in [0, q-1]\}}$$

The signer (or prover) M_i proves to the verifier the possession of its correct secret share d_i by using the following zero-knowledge proof system. The verifier can either be one of the players or an outsider who has inputs $w_{i0}, g, h, p, s_i, m, N, q$. All the protocols run in parallel, and failure of these protocols at any stage implies the failure of the whole proof.

1. The verifier follows the setup procedure of the Damgard-Fujisaki-Okamoto commitment scheme [23, 11], e.g. it picks a safe RSA modulus n and two elements G, H

in $\mathbb{Z}_n^*$ whose orders are greater than 2. (We refer to [11] for the details of this commitment scheme.) If N is a safe RSA modulus then set $n = N$, $G = (G')^2 \bmod N$, $H = (H')^2 \bmod N$ for random $G', H' \in \mathbb{Z}_n^*$.

2. The prover computes the commitment $C = G^{d_i} H^R \pmod{n}$, where R is picked randomly from $[0, 2^v(q-1)]$ and uses **Protocol (1)** (see below), by substituting $(x, x_1', x_2', g_1, h_1, g_2, h_2, n_1, n_2, w_1, w_2, b, b')$ with $(d_i, R, d_i', G, H, g, h, n, p, C, w_{i0}, q-1, 2^v(q-1))$, respectively, to execute:
   $ZKPK\{d_i, R, d_i' : C = G^{d_i} H^R \pmod{n} \ \wedge\ w_{i0} = g^{d_i} h^{d_i'} \pmod{p}\}$.
3. The prover then uses **Protocol (1)** (see below), by substituting $(x, x_1', x_2', g_1, h_1, g_2, h_2, n_1, n_2 w_1, w_2, b, b')$ with $(d_i, R, 0, G, H, m, m, n, N, C, s_i, q-1, 2^v(q-1))$, respectively, to execute:
   $ZKPK\{d_i, R : C = G^{d_i} H^R \pmod{n} \ \wedge\ s_i = m^{d_i} \pmod{N}\}$.
4. The prover uses **Protocol (2)** (see below), by substituting (x, x', b) with $(d_i, R, q-1)$, respectively, to execute:
   $ZKPK\{d_i, R : C = G^{d_i} H^R \pmod{n} \ \wedge\ d_i \in [0, q-1]\}$

Protocol (1). $\boxed{ZKPK\{x, x_1', x_2' : w_1 = g_1^x h_1^{x_1'} \pmod{n_1} \ \wedge\ w_2 = g_2^x h_2^{x_2'} \pmod{n_2}\}}$

Assumption: $x, x_2' \in [0, b]$ and $x_1' \in [0, b']$.

This protocol is from [5], [2], and is perfectly complete, honest verifier statistical zero-knowledge and sound under the strong RSA assumption [23] with the soundness error 2^{-u+1}, given than (g_1, h_1, n_1) is an instance of the Damgard-Fujisaki-Okamoto commitment scheme [23, 11].

1. The prover picks random $r \in [1, \ldots, 2^{u+v}b - 1]$, $\eta_1 \in [1, \ldots, 2^{u+v}b' - 1]$, $\eta_2 \in [1, \ldots, 2^{u+v}b - 1]$ and computes $W_1 = g_1^r h_1^{\eta_1} \pmod{n_1}$ and $W_2 = g_2^r h^{\eta_2} \pmod{n_2}$. It then sends W_1 and W_2 to the verifier V.
2. The verifier selects a random $c \in [0, \ldots, 2^u - 1]$ and sends it back to the prover.
3. The prover responds with $s = r + cx$ (in $\mathbb{Z}$), $s_1 = \eta_1 + cx_1'$ (in $\mathbb{Z}$) and $s_2 = \eta_2 + cx_2'$ (in $\mathbb{Z}$)
4. The verifier verifies as $g_1^s h_1^{s_1} = W_1 {w_1}^c \pmod{n_1}$ and $g_2^s h_2^{s_2} = W_2 {w_2}^c \pmod{n_2}$.

Protocol (2). $\boxed{ZKPK\{x, x' : C = G^x H^{x'} \pmod{n} \ \wedge\ x \in [0, b]\}}$

Assumption: $x \in [0, b]$ and $x' \in [0, 2^v b]$.

This protocol (from [2]) is an exact range proof, honest verifier statistical zero-knowledge, complete with a probability greater than $1 - 2^{-v}$, and sound under the strong RSA assumption given that (G, H, n) is an instance of the Damgard-Fujisaki-Okamoto commitment scheme, similarly as in protocol (1).

1. The prover sets $T = 2(u + v + 1) + |b|, X = 2^T x, X' = 2^T x', \beta = 2^{u+v+1}\sqrt{b}$ and $C_T = G^X H^{X'} \pmod{n}$.
2. The prover uses **Protocol (3)** (see below), by substituting (x, x', com, B, γ) with $(X, X', C_T, 2^T b, 2^{T/2}\beta)$, respectively, to execute the following (note that $X \in [0, 2^T b]$):
   $ZKPK\{X, X' : C_T = G^X H^{X'} \pmod{n} \ \wedge\ X \in [-2^{T/2}\beta, 2^T b + 2^{T/2}\beta]\}$

   Proving that $X \in [-2^{T/2}\beta, 2^T b + 2^{T/2}\beta]$ implies that $x \in [0, b]$, since $2^{T/2}\beta < 2^T$.

Protocol (3). $\boxed{ZKPK\{x, x' : com = G^x H^{x'} \pmod n \ \wedge \ x \in [-\gamma, B + \gamma]\}}$
Here $\gamma = 2^{u+v+1}\sqrt{B}$.

Assumption: $x \in [0, B]$ and $x' \in [0, 2^v B]$.

This proof was proposed in [2] and is honest verifier statistical zero-knowledge, complete with a probability greater than $1 - 2^{-v}$, and sound under the strong RSA assumption just like protocol (2).

1. The prover executes $ZKPK\{x, x' : com = G^x H^{x'} \pmod n\}$
2. The prover sets $x_1 = \lfloor\sqrt{x}\rfloor, x_2 = x - x_1^2$, $\hat{x}_1 = \lfloor\sqrt{B - x}\rfloor, \hat{x}_2 = B - x - \hat{x}_1^2$, and chooses randomly r_1, $r_2, \hat{r}_1, \hat{r}_2$ in $[0, 2^v B]$, such that $r_1 + r_2 = x'$ and $\hat{r}_1 + \hat{r}_2 = -x'$.
3. The prover computes new commitments $e_1 = G^{x_1^2} H^{r_1} \pmod n, \hat{e}_1 = G^{\hat{x}_1^2} H^{\hat{r}_1} \pmod n, e_2 = G^{x_2} H^{r_2} \pmod n, \hat{e}_2 = G^{\hat{x}_2} H^{\hat{r}_2} \pmod n$, and sends e_1 and $\hat{e}_1$ to the verifier.
4. The verifier computes $e_2 = com/e_1 \pmod n$ and $\hat{e}_2 = G^B/(com * \hat{e}_1) \pmod n$.
5. The prover uses **Protocol (4)** (see below), by substituting (x, x', com_{sq}) with (x_1, r_1, e_1) and then with $(\hat{x}_1, \hat{r}_1, \hat{e}_1)$, to execute the following:
   $ZKPK\{x_1 : e_1 = G^{x_1^2} H^{r_1} \pmod n\}$
   $ZKPK\{\hat{x}_1 : \hat{e}_1 = G^{\hat{x}_1^2} H^{\hat{r}_1} \pmod n\}$
   This proves that e_1 and $\hat{e}_1$ hide a square.
6. The prover uses **Protocol (5)** (see below), by substituting (x, x', com_2, B_1) with $(x_2, r_2, e_2, 2\sqrt{B})$, respectively and then with $(\hat{x}_2, \hat{r}_2, \hat{e}_2, 2\sqrt{B})$, respectively, to execute the following (note that x_2 and $\hat{x}_2 \in [0, 2\sqrt{B}]$):
   $ZKPK\{x_2 : e_2 = G^{x_2} H^{r_2} \pmod n \ \wedge \ x_2 \in [-\gamma, \gamma]\}$
   $ZKPK\{\hat{x}_2 : \hat{e}_2 = G^{\hat{x}_2} H^{\hat{r}_2} \pmod n \ \wedge \ \hat{x}_2 \in [-\gamma, \gamma]\}$
   This proves that e_2 and $\hat{e}_2$ hide numbers belonging to $[-\gamma, \gamma]$.

   Steps 2, 5 and 6 above, imply that $x \in [-\gamma, B + \gamma]$.

Protocol (4). $\boxed{ZKPK\{x, x' : com_{sq} = G^{x^2} H^{x'} \pmod n\}}$

This protocol first appeared in [23], generalized (and corrected) in [11] and proves that a committed number is a square. The protocol is honest verifier statistical zero-knowledge, perfectly complete, and sound under the strong RSA assumption just like protocol (2).

Protocol (5). $\boxed{ZKPK\{x, x' : com_2 = G^x H^{x'} \pmod n \ \wedge \ x \in [-2^{u+v} B_1, 2^{u+v} B_1]\}}$

Assumption: $x \in [0, B_1]$, and $x' \in [0, 2^v B_1]$.

This proof was proposed in [8], allows a prover to prove the possession of a discrete logarithm x lying in the range $[-2^{u+v} B_1, 2^{u+v} B_1]$ given x which belongs to a smaller interval $[0, B_1]$. Using the commitment scheme of [23, 11], this proof is honest verifier statistical zero-knowledge, complete with a probability greater than $1 - 2^{-v}$, and sound under the strong RSA assumption with soundness error 2^{-u+1}.