New Efficient and Secure Protocols for Verifiable Signature Sharing and Other Applications*

Dario Catalano[1] and Rosario Gennaro[2]

[1] Dipartimento di Matematica, Universita' di Catania. Viale A. Doria 6, 95100 Catania. Email: dario@liotro.dipmat.unict.it
[2] I.B.M. T.J.Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598. Email: rosario@watson.ibm.com

Abstract. Verifiable Signature Sharing ($V\Sigma S$) enables the recipient of a digital signature, who is not necessarily the original signer, to share such signature among n proxies so that a subset of them can later reconstruct it. The original RSA and Rabin $V\Sigma S$ protocols were subsequently broken and the original DSS $V\Sigma S$ lacks a formal proof of security.

We present new protocols for RSA, Rabin and DSS $V\Sigma S$. Our protocols are efficient and provably secure and can tolerate the malicious behavior of up to half of the proxies. Furthermore we believe that some of our techniques are of independent interest. Some of the by-products of our main result are: a new threshold cryptosystem, a new undeniable signature scheme and a way to create binding RSA cryptosystems.

1 Introduction

The concept of Verifiable Signature Sharing ($V\Sigma S$) was introduced by Franklin and Reiter in [14]. $V\Sigma S$ enables the recipient of a digital signature, who is not necessarily the original signer, to share such signature among n proxies so that a subset of them can later reconstruct it. A $V\Sigma S$ protocol is divided in a *sharing* phase and a *recover* phase. At the end of the sharing phase each proxy can verify that a valid signature for the given document can be reconstructed. At the end of the recover phase such signature is reconstructed no matter what a malicious subset of proxies may do.

PREVIOUS WORK. In [14] efficient protocols were given for RSA, Rabin, ElGamal, Schnorr and DSS signatures. However their RSA and Rabin $V\Sigma S$ protocols were subsequently broken in [7]. Also their DSS $V\Sigma S$ achieves only an heuristic form of security.

In [3] Burmester shows an unifying approach to $V\Sigma S$ based on homomorphism of secret sharing schemes. The approach is very elegant and also secure. However its generality does not yield extremely efficient protocols when applied to typical real-life signature schemes.

* Extended abstract. A final version of this paper can be found at http://www.research.ibm.com/security/vsigmas.ps

Thus the question of efficient and provably secure VΣS schemes for RSA/Rabin and DSS was still open.

OUR CONTRIBUTION. In this paper we present new protocols for RSA, Rabin and DSS VΣS . Our protocols are *efficient* and *provably secure*. They can tolerate a malicious sharer (who tries to share something different from a valid signature) and the malicious behavior of up to half of the proxies during sharing or reconstruction time.

MOTIVATION. It is important to notice that VΣS can be solved in theory using known cryptographic techniques for zero-knowledge proofs [22, 19] and multiparty computation [20, 2, 5]. However these solutions are hardly practical. We focus instead on *practical* solutions since there are several real-life applications which would greatly benefit from secure and efficient VΣS protocols. In order to motivate the problem we present first some of the most interesting applications in which VΣS can be used.

The main application of VΣS is the incorporation of digital cash into multiparty protocols. Consider the example of cash escrow: digital cash can be represented as the bank's signature on a digital coin or e-check. By using VΣS financial institutions can divide the cash among a set of authorities so that only through the cooperation of a threshold of them it is possible to spend it, yet the authorities can verify that they collectively have the cash. A related application is secure distributed auctions: bidders to an auction may be required to verifiably share a signature on a check for the amount of their bid. This way it will be impossible for the winner of the bid to default (since the proxies can reconstruct his check), while the payments of the losers will never be recovered.

More generally VΣS is useful when a signed document should become valid only under certain conditions. By verifiably sharing the signature, one makes sure that the signature will be recovered if and only if such conditions are created.

OTHER APPLICATIONS. We believe that parts of our solution are relevant on their own. For example, in Section 3 we present a new threshold cryptosystem which is possibly of independent interest. VΣS protocols are somewhat related to undeniable signatures (introduced by Chaum in [4]), i.e. signatures that can be verified only with the help of the signer. RSA-based undeniable signatures were recently introduced in [17]. Our RSA construction can be seen as an alternative to [17] (though admittedly a less efficient scheme). Interestingly the structure of our RSA VΣS protocol can also be used to construct binding RSA cryptosystems. In [31] the concept of binding cryptography was introduced. In a *binding public-key cryptosystem* the sender encrypts a message under both the public key of the receiver and the public key of a third party and prove in (non-interactive) ZK that the two ciphertexts contain the same message. In [31] a scheme for binding ElGamal was presented. We show that our RSA VΣS scheme can be used to construct binding RSA public key encryptions. Details about these applications can be found in the final paper.

1.1 Overview of Our Solution

Let Bob be the signer, Alice the recipient of the signature and $P_1, \ldots, P_n$ the proxies (see Section 2 for a detailed description of the model).

THE RSA SCHEME. Let (N_B, v_B) be Bob's RSA public key. The matching signing key is s_B such that $s_B v_B = 1 \bmod \phi(N_B)$. We assume the standard "hash-and-sign" paradigm. Alice receives a message M from Bob and Bob's signature $S = m^{s_B} \bmod N_B$ on it where $m = H(M)$ for some collision-resistant hash function H. Alice wants to verifiably share the value S among the proxies $P_1, \ldots, P_n$. That is, at the end of the sharing phase the proxies must be assured that they hold shares of Bob's signature on m (from now on we will drop the hashing step since it is irrelevant to our purposes).

Our new RSA VΣS scheme is based on a completely novel approach. Alice will not share S using conventional secret sharing schemes, but instead she will encrypt S using a threshold cryptosystem, i.e. a public key whose matching secret key is kept shared at the proxies. That is, she gives to the proxies the values m and $C_S = E_{EK}(S)$ where E is a public key encryption scheme and the decryption key DK is shared at the proxies, i.e. each proxy P_i has a share DK_i of a t-out-of-n secret sharing of DK. At this point all we need is a mechanism to convince the proxies that the decryption of C_S is really the signature S on m without revealing S.

The crux of the problem was to design a threshold cryptosystem that would make such proof efficient. The main idea here is to use the ElGamal encryption scheme [10, 11] over the *same composite modulus* N_B over which the signature was computed. This will allows us to construct efficient methods to convince the proxies that the ciphertext contains the signature. We present two such methods.
(1) One is for Alice to provide a zero-knowledge proof [22] that C_S contains S. We present an efficient ZK proof for this task.
(2) The other variant consists on the proxies using their private key to decrypt a message $\hat{C}_S$ publicly computable from C_S. If such decryption equals the message m, then the proxies are guaranteed that C_S contains the signature S.
Method (1) is more efficient for the proxies, but requires interaction with Alice. Method (2) is more efficient for Alice and requires *no* interaction between her and the proxies.

The solution is described in detail in Section 4

THE DSS SCHEME. The DSS VΣS scheme is a modified version of the ElGamal VΣS scheme from [14] which allows for a proof of security. The solution is described in Section 5

1.2 Related Work

Some ideas in our new RSA VΣS scheme have appeared previously in the literature although in different contexts and with different usage.

Performing secret sharing by encrypting a value with a symmetric key that is shared among the proxy appeared first in [23] as a way to shorten the size of shares in computationally secure secret sharing schemes.

The idea of encrypting a signature and then proving in ZK that the ciphertext contains a valid signature has appeared in several places. In [8] it was proposed as a general paradigm to construct undeniable signatures but the specific efficient solutions work only for ElGamal-like signatures. In [1] this technique was used to construct fair exchange of digital signatures between two parties using an off-line trusted center. The paper is not concerned with using a threshold cryptosystem for encrypting the signature. Moreover they present general solutions for RSA and DSS signatures using any kind of public-key encryption but employing inefficient binary cut-and-choose ZK proofs which require a number of public-key operations which is proportional to the security parameter.

Stadler in [30] uses ElGamal over a composite to verifiably encrypt e-roots. But when it comes to share such encryptions, his scheme can only deal with additive access structures, thus resulting in an $O(n^t)$ exponential blow-up to achieve a t-out-of-n threshold scheme.

The construction of an ElGamal-like threshold cryptosystem over a composite modulus uses techniques from the areas of threshold [9, 15] and proactive [13, 27] RSA signature schemes.

2 Preliminaries

THE MODEL. We assume there are three entities. The *signer* (which in the following we will usually call Bob), the *recipient* (Alice) and a set of n *proxies*. The VΣS protocol will be between Alice and the proxies and must not involve Bob. We assume that Alice and the proxies are connected by a full network of private channels and by a broadcast channel. These assumptions allow us to focus on a high-level description of the protocols. However, it is worth noting that these abstractions can be substituted with standard cryptographic techniques for privacy, commitment and authentication. In some of the variations of our protocols it will not be necessary to have private channels between Alice and the proxies. We assume that there exist an adversary $\mathcal{A}$ who can corrupt Alice and at most t of the proxies. By corrupting a player, $\mathcal{A}$ can read his memory and cause him to deviate arbitrarily from the protocol. We assume the adversary is static i.e. the set of corrupted players is decided at the beginning of the computation of a protocol. Finally we assume communication to be synchronous. We do however allow for *rushing* adversaries, i.e. adversaries who decide the messages of the bad players at round R after having seen the messages of the good players at the same round.

NOTATION. In the rest of the paper n will denote the number of proxies and $L = n!$. If N is a composite modulus, we denote with G_0 an element of maximal order in Z_N^* and with $G = G_0^{L^2} \bmod N$. $DLog_G\ A \bmod N$ is the discrete log in base G of A modulo N.

DEFINITION. We follow the ideas in the definition of VΣS presented in [14], although we believe our formalization to be simpler and more rigorous.

VΣS consists of a pair of protocols (ΣShare , ΣRecover) for Alice and the proxies. The input of ΣShare for all the players is a message m and the public

verification key VK of the signer. The secret input for Alice is a signature S of m under the signer's key. The output of ΣShare for each proxy P_i is a value S_i, which can assume the special value $S_i = \omega$ denoting that the proxy has rejected the sharing. The protocol ΣRecover is then run on the output of ΣShare by the proxies.

We say the VΣS = (ΣShare , ΣRecover) is a Verifiable Signature Sharing protocol with fault-tolerance t if for any adversary $\mathcal{A}$ that can corrupt Alice and at most t proxies the following conditions are met:

completeness If Alice is not corrupted then the output of ΣRecover is a signature S on m under the signer's key VK.

soundness If a good proxy P_i outputs $S_i = \omega$ at the end of ΣShare then each good player P_j outputs $S_j = \omega$. If $S_i \neq \omega$ for good players then the output of ΣRecover is a signature S on m under the signer's key VK.

security Define the view $\mathcal{V}$ of the adversary $\mathcal{A}$ as the set of messages sent and received by the bad players (including the broadcasted ones) at the end of ΣShare . Then there exists an algorithm $\mathcal{S}$ called the *simulator* which on input only m and VK and black-box access to $\mathcal{A}$, produces output strings with a distribution which is computationally indistinguishable from $\mathcal{V}$.

We accept a negligible probability (over the coin tosses of the players) that these conditions are violated. Informally, completeness means that if Alice really shares the right signature, then, no matter what malicious proxies do, the signature will be recovered at the end. Soundness means that if Alice is malicious, then either she will be caught trying to cheat (i.e. sharing something different from a valid signature) or she will share a valid signature anyway. Security finally says that a run of ΣShare gives the adversary no information he could not compute on his own from the message and the public key. In particular (unless the signature scheme is not secure) no information about the signature S.

COMPUTATIONAL ASSUMPTIONS. Our RSA VΣS protocol uses the so-called *Decisional Diffie-Hellman Assumption* (DDH) over a composite modulus. Informally this assumption says that given two random Diffie-Hellman "public keys" $A = G^a$ and $B = G^b$ the resulting shared key G^{ab} is indistinguishable from a random value to an observer who does not know any of the secret keys a, b. More formally stated:

Assumption 1 *Let N be a composite modulus product of two large primes. Let G be an element of Z_N^* and $\mathcal{G}$ the group generated by it. Consider the two probability distributions on $\mathcal{G}^3$ defined as $\mathcal{DH} = (G^a, G^b, G^{ab}) \bmod N$ and $\mathcal{R} = (G^a, G^b, G^c) \bmod N$ for a, b, c chosen randomly and uniformly in Z_N. We assume that the two distributions are computationally indistinguishable.*

The Decisional Diffie-Hellman Assumption is related to the regular Diffie-Hellman assumption that says that given G^a and G^b one cannot compute G^{ab} in polynomial time. Clearly this assumption relies on the hardness of computing discrete logs. Reductions in the inverse direction are not known.

ELGAMAL OVER A COMPOSITE. We recall the functioning of the ElGamal encryption scheme [10,11] over a composite modulus [24,29]. The public encryption key is $EK = (N, G, Y)$ where N is an RSA modulus product of two primes, G is an element of Z_N^* of large order and Y is computed as $Y = G^X \bmod N$ with $X \in_R Z_N$. X is the secret decryption key. A message M is encrypted under EK by choosing a random $K \in_R Z_N$ and computing $A = G^K \bmod N$ and $B = Y^K \cdot M \bmod N$. The ciphertext is the pair (A, B). Decryption of a pair (A, B) is computed by taking $M = B/A^X \bmod N$. Notice that in order to decrypt, all it is required is to raise A to the secret exponent X.

The ElGamal encryption scheme can be thought as a one-time pad with a one-time Diffie-Hellman key (the value $Y^K = G^{XK}$). Thus if the message M is in the same group generated by G it is an easy task to show that the DDH implies the semantic security of the ElGamal encryption scheme (semantic security defined in [21], means that it is impossible for an observer to distinguish between the encryption of two messages). If the message space is larger than the group generated by G then the semantic security of the ElGamal encryption scheme is a seemingly stronger assumption than the DDH.

2.1 Tools

We will use the polynomial-based t-out-of-n secret sharing due to Shamir [28]. Let q be a prime: given a secret $\sigma \in Z_q$, the dealer chooses at random a polynomial $f(z) = \sigma + \sum_{j=1}^{t} a_j z^j$ of degree t and gives to player P_i a share $\sigma_i = f(i) \bmod q$. Clearly t players have no information about the secret while $t+1$ can reconstruct it by polynomial interpolation. Notice that a $(n-1)$-out-of-n secret sharing can be obtained simply by sharing σ as a sum $\sigma = \sigma_1 + \ldots + \sigma_n$.

Basic secret sharing protocols cannot cope with a malicious dealer who gives out random points that do not lie on a polynomial of degree t and/or with malicious players who contribute false shares at reconstruction time. A Verifiable Secret Sharing (VSS) protocol [6] solves these problems. Here we recall Feldman's VSS [12]. The dealer follows Shamir's scheme but in addition he broadcasts the values $\alpha_0 = g^\sigma \bmod p$ and $\alpha_j = g^{a_j} \bmod p$ where p is a prime such that q divides $p-1$ and g is an element of order q in Z_p^*. The α values will allow the players to check that the values σ_i really define a secret by checking that $g^{\sigma_i} = \prod_j \alpha_j^{i^j} \bmod p$. If the value they hold is inconsistent they complain about the dealer who will reveal their share (that should match the above equation). The α values also allow detection of incorrect shares σ_i' at reconstruction time. This protocol can tolerate up to $t = n/2$ malicious faults *including the dealer.*

Notice that the value of the secret is only computationally secure, e.g., the value $g^\sigma \bmod p$ is leaked. To avoid this problem it is possible to use Pedersen's VSS [26] which protects the secret in an information-theoretic sense. In this implementation the dealer chooses a second t-degree polynomial $f' = \sum_j b_j z^j$ and sends the value $\tau_i = f'(i) \bmod p$ to player P_i in addition to the share σ_i as above. The dealer then commits to each coefficient of the polynomials f, f' as follows: he publishes the values $\beta_j = g^{a_j} h^{b_j} \bmod p$ where h is an element in the subgroup generated by g such that the discrete log of h in base g is

unknown. This will allow the players to check the received shares by checking that $g^{\sigma_i} h^{\tau_i} = \prod_j \beta_j^{i^j} \bmod p$. At revealing time the players are required to reveal both σ_i and τ_i and the above equation is used to validate the shares. It is possible to prove that the VSS fails if and only if the adversary is able to compute the discrete log of h in base g. Notice that the value of the secret is unconditionally protected since the only value revealed is $\beta_0 = g^{\sigma} h^{b_0}$.

3 A new threshold cryptosystem

Our RSA VΣS scheme relies on a new ElGamal-based threshold cryptosystem which we present in this section. We believe this new threshold cryptosystem to be of independent interest. Although the construction of this ElGamal-based threshold cryptosystem is new, the techniques used in this section appears in several papers related to threshold [9,15] and proactive [13,27] RSA signature schemes.

THRESHOLD CRYPTOSYSTEMS. Let E be a public key encryption scheme. A *threshold cryptosystem* T_E for a scheme E distributes the operation of key generation and decryption among a set of n parties $P_1, \ldots, P_n$. That is, T_E is defined by two protocols:

T-Key-Gen a randomized protocol that returns as public output the public encryption key EK and as private output for player P_i a value DK_i such that $DK_1, \ldots, DK_n$ constitute a t-out-of-n threshold secret sharing of DK.

T-Decrypt each player P_i takes as public input a ciphertext $C = E_{EK}(M)$ and as secret input his share DK_i and returns as public output the message M

The two protocols should be *secure* i.e. they should function correctly and reveal no extra information even in the presence of an adversary that corrupts maliciously up to t players[1]. In particular notice that the private key DK should not be exposed during T-Decrypt. Formal definitions for threshold cryptosystems appear in the final paper.

A THRESHOLD CRYPTOSYSTEM FOR ELGAMAL OVER A COMPOSITE. There are several complications that arise from trying to generalize the approach in [25] to work for ElGamal modulo a composite.

First of all we will require for our application that the modulus N will be given as a parameter to the key generation protocol[2] without its factorization. This implies that the value $\phi(N)$ is unknown to the parties who have to jointly choose and share X. Our threshold cryptosystem overcomes this problem using techniques discovered by [13] for the application of proactive RSA.

FELDMAN'S VSS OVER A COMPOSITE. First we present how to modify Feldman's VSS to work modulo a composite of unknown factorization. This protocol was

[1] Notice that we are talking about *robust* protocols that work in the presence of malicious faults. Unless otherwise noted when we say "secure" we mean also "robust".

[2] Jumping ahead, N will be the same modulus from Bob's public key

discovered in [13]. They used it as a crucial tool to refresh shares of a proactive RSA signature scheme. We slightly modified it to work as a component of our key generation protocol. The main idea behind the protocol is for the the dealer to share the secret σ over the integers (since he doesn't know $\phi(N)$). The coefficients of the sharing polynomial must be chosen large enough to statistically hide information. The protocol appears in Figure 1.

Lemma 1. Feldman-Z_N-VSS *is a VSS of fault-tolerance t for any t, n such that $n > 2t$.*

Feldman-Z_N-VSS

Input for all players: a composite number N and an element G of maximal order in Z_N^*.
Input for the dealer: a secret $\sigma \in [-N^2 \dots N^2]$.

Sharing Phase.

1. The dealer chooses t random integers $a_1, \dots, a_t \in [-L^2N^3 \dots L^2N^3]$ and consider the polynomial $f(z) = L\sigma + a_1 z + \dots + a_t z^t$
2. The dealer sends to player P_i the integer $\sigma_i = f(i)$ and broadcasts $\alpha_0 = G^\sigma \bmod N$ and $\alpha_i = G^{a_i} \bmod N$.
3. Player P_i checks that

$$G^{\sigma_i} = \alpha_0^L \prod_{j=1}^{t} \alpha_j^{i^j} \bmod N \qquad (1)$$

   If the check fails he asks that the dealer makes σ_i public. If more than t player make this request the dealer is disqualified.
4. The dealer reveals σ_i for the players who asked and the previous check is carried out on all public shares. If it fails the dealer is disqualified.

Reconstruction Phase. Player P_i broadcasts σ_i. Accept those for which Equation (1) is satisfied. Take $t+1$ accepted shares and compute the secret σ via polynomial interpolation on a prime field large enough (i.e. $p > n^{t+2}L^2N^3$).

Fig. 1. Feldman VSS over a composite modulus

KEY GENERATION PROTOCOL. We are now ready to show the key generation protocol for the threshold ElGamal scheme.

The general idea follows the one of Pedersen [25] for the case of discrete-log cryptosystems in a prime field. Each player P_i shares a random value x_i via Feldman's VSS. The secret key x will be the sum of those random values, while the public key $y = g^x$ is easily computable from the public information of the VSS protocols. The key generation continues by having each player sum up the

shares he received to create his own share of the secret key for a t-out-of-n secret sharing. There are two difficulties with the above approach however:

- Since each Feldman's VSS reveals g^{x_i} it is possible for a rushing adversary to choose the x_i's of the bad players so that a specific $y = g^x$ will appear. Even if the adversary were not rushing, it would be possible for her during the complain phase to "pull out" some bad players (by having them disqualified) in order to affect the value of y. At the end it is not possible to prove that the pair x, y is built with the right (uniform) distribution. This problem was first noticed by [16]. Their solution, which we are going to employ on this paper, is to perform an information-theoretically secure VSS first for x_i (for example Pedersen's VSS) and then on top of that (using the same sharing polynomial) perform the checks required by the Feldman's VSS. This has the effect of forcing the decision of the adversary when the x_i's are information-theoretically secure and thus the choice of the adversary is independent from the ones of the good players.
- When working over a composite modulus the threshold decryption protocol is unnecessarily cumbersome if the key is represented in a t-out-of-n fashion. Thus we follow a different approach. Each player will keep as a share of the secret key the random value he originally shared: this is a $(n-1)$-out-of-n secret sharing. The shares a player received during Feldman-Z_N-VSS will be used for backup in case some other player fails during the decryption protocol. This paradigm was introduced by [27] and called *share-backup*.

The protocol TEG-Key-Gen (for Threshold ElGamal Key Generation) appears in figure 2.

DECRYPTION PROTOCOL. We are now left to show the decryption protocol. The approach is the same as the decryption protocol modulo a prime in [25], but it uses the techniques from [13, 27], to make it work modulo a composite.

The idea is to get a partial decryption from each player by exponentiating the ciphertext to his own additive (not threshold!) share of the secret key. Since the secret key is the sum of the additive shares, the product of the partial decryptions will be the correct decryption. To prevent bad players from contributing bad partial decryption we force them to prove in ZK that they are correct with respect to the witnesses generated during the key generation protocol. The ZK proof for this task is described[3] in [15]. The protocol TEG-Decrypt appears in Figure 3.

Theorem 1. TEG = (TEG-Key-Gen, TEG-Decrypt) *is a secure threshold cryptosystem for ElGamal over a composite with fault-tolerance t for any t, n such that $n > 2t$.*

[3] [15] requires the modulus N to be a product of safe primes. It is possible to lift this assumption however by using a different protocol.

TEG-Key-Gen

Input for all players: A composite number N product of two primes and an element G of maximal order in Z_N^*.

1. Each player P_i selects a random number $x_i \in_R [-N^2 \dots N^2]$ and t random integers $a_{i1}, \dots, a_{it} \in [-L^2N^3 \dots L^2N^3]$ and consider the polynomial $f_i(z) = Lx_i + a_{i1}z + \dots + a_{it}z^t$. Player P_i performs an unconditionally secure VSS of x_i using sharing polynomial $f_i(z)$ (for example Pedersen's VSS over a large enough prime field). Let *Good* be the set of players who are not disqualified at the end of this step.
2. Player P_i broadcasts $\alpha_{i0} = G^{x_i} \bmod N$ and $\alpha_{ik} = G^{a_{ik}} \bmod N$.
3. Let $x_{ji} = f_j(i)$ be the integer player P_i received during step 1. Player P_i checks that
$$G^{x_{ji}} = \alpha_{j0}^{L} \prod_{k=1}^{t} \alpha_{jk}^{i^k} \bmod N \qquad (2)$$
If the check fails P_i opens the commitment to the share he received in Step 1 and proves that P_j is cheating.
4. Each P_j caught cheating on the previous step is exposed. That is the value x_j is reconstructed using the VSS of step 1. The value α_{j0} is reset to $G^{x_j} \bmod N$.
5. The public key is set to $Y = \prod_{i \in Good} \alpha_{i0} \bmod N$ (including the exposed values). The private share of player P_i is the vector $(x_i, \{x_{ji}\}_{j \in Good})$.

Fig. 2. A Pedersen-like joint key generation with share-backup

TEG-Decrypt

Input for all players: A composite number N product of two primes, an element G of maximal order in Z_N^*, the public output of TEG-Key-Gen and a ciphertext (A, B).
Private Input for Player P_i: The secret output of TEG-Key-Gen, i.e. the vector $(x_i, \{x_{ji}\}_{j \in Good})$.

1. Each player $P_i \in Good$ broadcasts the partial decryption $A_i = A^{x_i} \bmod N$ and proves in ZK using the protocol from [15] that $DLog_A\ A_i = DLog_G\ \alpha_{0i}$.
2. If $P_i \in Good$ and he fails the ZK proof of the above step, all the players run the reconstruction phase of Feldman-Z_N-VSS and compute x_i and $A_i = A^{x_i} \bmod N$ on their own.
3. Output the message $M = \frac{B}{\prod_{i \in Good} A_i} \bmod N$.

Fig. 3. Threshold Decryption for ElGamal over a composite

4 Sharing an RSA Signature

THE PROBLEM. Let (N_B, v_B) be Bob's RSA public key. The matching signing key is s_B such that $s_B v_B = 1 \bmod \phi(N_B)$. Alice receives a message m from Bob and Bob's signature $S = m^{s_B} \bmod N_B$ on it. Alice wants to verifiably share the value S among the proxies $P_1, \ldots, P_n$. That is at the end of the sharing the proxies must be assured that they hold a sharing of Bob's signature on m.

THE BASIC PARADIGM. We depart from the approach used by Franklin and Reiter [14] of directly sharing the signature S. Instead we follow the alternative approach to obtain secret sharing using an encryption of the secret with a key which is shared at the proxies. Assume that the proxies have established an instance of a public key encryption scheme E with public key EK and that the matching secret key SK is shared among them in a t-out-of-n fashion. Then all Alice has to do to share S among the proxies is to simply give them the value $C_S = E_{PK}(S)$. Indeed t or less proxies have no information about the secret key, thus they cannot decrypt S. In order to achieve the verifiability property we need a proof that C_S indeed contains the signature S.

ACHIEVING VERIFIABILITY. The above proof could be constructed using general zero-knowledge proof techniques, with a loss of efficiency. The key idea to obtain an efficient proof is to encrypt the signature S (created under Bob's public key (N_B, v_B)) using an instance of the ElGamal cryptosystem *over the same composite modulus* N_B.

If $Y = G^X \bmod N_B$ is the public key of the proxies, Alice will encrypt the signature by choosing a random $K \in Z^*_{N_B}$ and computing $A_S = G^K \bmod N_B$ and $B_S = Y^K \cdot S \bmod N_B$.

This will allow for an efficient verifiability check using the ZK proof in Figure 4. Alice will be the prover and the proxies will be the verifiers. The protocol is based on [30] with some efficiency improvements (it is not necessary to repeat it several times.)

Lemma 2. *The protocol* EGRSA-ZK-Proof *is a honest-verifier zero-knowledge proof that* (A_S, B_S) *is an ElGamal encryption under the key* (N_B, G, Y) *of the signature* S *of* m *under the key* (N_B, v_B).

To make the above protocol ZK against any verifier, known techniques [18] can be used. In particular the Verifier can commit to the challenge before the Prover speaks.

THE BASIC PROTOCOL. On input a message m, Bob's signature S on it and Bob's public key N_B, v_B Alice does the following. She generates an instance of the ElGamal encryption scheme over the composite N_B. She generates G as a random power of m (for reasons that will appear clear in the security proof). She selects a random $X \in [-N_B^2, \ldots, N_B^2]$ and the public key $Y = G^X \bmod N_B$. She then encrypts S under by selecting a random $K \in Z_{N_B}$ and setting $A_S = G^K \bmod N_B$ and $B_S = Y^K \cdot S \bmod N_B$. She hands $m, N_B, v_B, G, Y, A_S, B_S$ to the proxies. Then she proves in ZK that A_S, B_S is indeed an encryption of the

EGRSA-ZK-Proof

Input for Prover and Verifier: The message m, Bob's RSA public key N_B, v_B, the ElGamal public key $G, Y = G^X$, the ciphertext $(A_S = G^K, B_S = Y^K \cdot S)$.
Secret input for the Prover: A value S such that $m = S^{v_B} \bmod N_B$ and the random value K.

1. The Prover chooses a random value $R \in_R Z^*_{N_B}$ and sets $r = R^{v_B} \bmod N_B$ (notice that R is a signature on r according to Bob's public key). The Prover encrypts R under the ElGamal key of the proxies, i.e. she chooses $J \in_R Z_{N_B^3}$ and sets $A_R = G^J \bmod N_B$ and $B_R = Y^J \cdot R \bmod N_B$. She sends r, A_R, B_R to the proxies.
2. The Verifier sends a random challenge $c \in Z_{N_B}$
3. The Prover answers with $L = J + cK$ (integer value) and $T = RS^c \bmod N_B$. The Verifier accepts if
   (a) $G^L = A_R \cdot A_S^c \bmod N_B$,
   (b) $B_R \cdot B_S^c = Y^L \cdot T \bmod N_B$ and
   (c) $T^{v_B} = rm^c \bmod N_B$.

Fig. 4. A ZK Proof that an ElGamal encryption contains an RSA signature

signature S using the EGRSA-ZK-Proof from Figure 4. Finally the last thing left is to share X using Feldman-Z_N-VSS from Figure 1. This will guarantee the proxies that they have the correct decryption key to reconstruct S.

To recover S the proxies will run the reconstruction phase of Feldman-Z_N-VSS and then use X to decrypt S. The protocol is described in Figure 5.

Notice that this protocol does not make use of the full threshold cryptosystem we have outlined in the previous section but only of the Feldman-Z_N-VSS protocol.

Theorem 2. *Under the Decisional Diffie-Hellman assumption modulo a composite the protocol* RSA-VΣS-1 *is a secure* VΣS *protocol for RSA with fault-tolerance t, for any n, t with $n > 2t$.*

AN ALTERNATIVE PROTOCOL. In this section we show a variation of the previous protocol. The reason we present an alternative protocol is to improve the efficiency of the scheme for Alice. Indeed in this scheme we take full advantage of the new threshold cryptosystem described in Section 3.

The main difference with respect to the previous protocol is that the key generation for the ElGamal scheme is done distributively by the proxies instead than by Alice. This will also allow for a very efficient verification that the ciphertext contains the required signature. Indeed the proxies can verify that the

RSA-VΣS-1

Input for Alice: The message m, Bob's RSA public key N_B, v_B, the signature S on m, i.e. a value such that $m = S^{v_B} \bmod N_B$.

RSA-ΣShare-1

1. Alice chooses uniformly at random
   - $r \in Z_{N_B}$ and sets $G = m^r \bmod N_B$.
   - $X \in [-N_B^2, \ldots, N_B^2]$ and sets $Y = G^X \bmod N_B$.
   - $K \in Z_{N_B}$ and sets $A_S = G^K \bmod N_B$ and $B_S = Y^K \cdot S \bmod N_B$.

   She sends to the proxies $m, N_B, v_B, r, Y, A_S, B_S$. The proxies compute $G = m^r \bmod N_B$.
2. Alice runs the proof EGRSA-ZK-Proof as the prover on input $m, N_B, v_B, G, Y, A_S, B_S$ with the proxies as the Verifier. The proxies reject if Alice fails the proof.
3. Alice runs Feldman-Z_N-VSS for X as the dealer with the proxies. The proxies reject if Alice fails the VSS protocol.

RSA-ΣRecover-1

The proxies run the Reconstruction Phase of Feldman-Z_N-VSS and compute X and output $S = \frac{B_S}{A_S^X} \bmod N_B$.

Fig. 5. Basic RSA VΣS Protocol

pair (A_S, B_S) is constructed correctly by checking that

$$\frac{B_S^{v_B}}{(A_S^{v_B})^X} = \frac{(Y^K \cdot S)^{v_B}}{(G^K)^{X \cdot v_B}} = \frac{Y^{K \cdot v_B} \cdot S^{v_B}}{Y^{K \cdot v_B}} = m \bmod N_B \tag{3}$$

i.e. just by running the TEG-Decrypt protocol on the pair $(A_S^{v_B}, B_S^{v_B})$. The full protocol appears in Figure 6.

Theorem 3. *Under the Decisional Diffie-Hellman assumption modulo a composite the protocol* RSA-VΣS-1 *is a secure* VΣS *protocol for RSA with fault-tolerance t, for any n, t with $n > 2t$.*

COMMENTS, VARIANTS AND OPTIMIZATIONS. Due to space limitations we refer the reader to the final version of the paper for a description of several variants of the above protocols and a detailed efficiency analysis.

5 Sharing a DSS signature

THE PROBLEM. Recall the DSS signature scheme. The public parameters are a large prime p, a 160-bit prime q that divides $p-1$ and an element g of order q in Z_p^*. Bob (the signer) has a secret key x which is a random number in Z_q. The matching public key is $y = g^x \bmod p$. A message M is signed by first hashing

RSA-VΣS-2

Input for Alice: The message m, Bob's RSA public key N_B, v_B, the signature S on m, i.e. a value such that $m = S^{v_B} \bmod N_B$.

RSA-ΣShare-2

1. Alice sends to the proxies the message m, Bob's public key (N_B, v_B) and a random value $r \in Z_{N_B}$
2. The proxies run TEG-Key-Gen on input the modulus N_B and the basis $G = m^r \bmod N_B$. They return to Alice the public key $Y = G^X \bmod N_B$.
3. Alice encrypts S using the ElGamal encryption scheme with public key (N_B, G, Y). That is, she generates a random number K in Z_{N_B} and computes $A_S = G^K \bmod N_B$ and $B_S = Y^K \cdot S \bmod N_B$. Alice sends (A_S, B_S) to the proxies.
4. The proxies run TEG-Decrypt protocol on the pair $(A_S^{v_B}, B_S^{v_B})$. If the output is m they accept otherwise they reject.

RSA-ΣRecover-2 The proxies run TEG-Decrypt protocol on the pair (A_S, B_S).

Fig. 6. Alternative RSA VΣS Protocol

it down via SHA-1, i.e. $m = SHA\text{-}1(M)$. Then Bob chooses a random number $k \in_R Z_q$, and computes $r = g^k \bmod p \bmod q$ and $s = k^{-1}(m + xr) \bmod q$. The signature is the pair (r, s). A signature is verified by computing $\sigma = s^{-1} \bmod q$ and checking that $r = (g^m y^r)^\sigma \bmod p \bmod q$. Alice receives the message M and the pair (r, s) and she wants to verifiably share the pair (r, s) among the proxies $P_1, \ldots, P_n$.

THE APPROACH FROM [14]. We follow the same provably secure approach used in [14] to verifiably share an ElGamal signature. The main difference between a (plain) ElGamal signature and DSS[4] is that $r = g^k \bmod p$ (without the extra mod$q$ reduction) and $s = k^{-1}(m - xr)$ (a $-$ instead of a $+$). The idea in [14] was for Alice to give out r in the clear to the proxies and to share s via a Feldman VSS using an appropriate basis for the verification. For some reason (probably because the extra reduction mod$q$ for r seems at first sight not to allow this approach) they decided to switch the roles between r and s in the DSS VΣS . Alice gives out s in the clear and shares r via Feldman's VSS. However their simulation does not go through, although they claim heuristically that it is secure.

OUR APPROACH. We show that a careful adaptation of the ElGamal VΣS from [14] can be shown to be secure. Alice gives out r in the clear to the proxies and shares $\sigma = s^{-1}$ via Feldman VSS using the basis $G = g^m y^r$ (which the proxies can compute on their own from g, y, m, r). Notice that by doing this she reveals

[4] Apart from the way messages are hashed which is irrelevant for the purpose of VΣS

$r^* = G^\sigma \bmod p$. So in order to check that the shared value is really the correct signature the proxies must simply check that $r^* \bmod q = r$. Notice that revealing r^* (i.e. the value of r *before* the reduction mod$q$) does not turn into a security problem since this information is easily simulatable (see the proof). Indeed the reduction mod$q$ of r serves only to shorten the signature and has no security purpose. The full protocol appears in Figure 7.

DSS-VΣS

Input for all players: Bob's DSS public key (p, q, g, y), a message M and its hash value m.
Secret Input for Alice: A DSS signature r, s from Bob on M.

DSS-ΣShare

1. Alice broadcasts r to the proxies. She runs Feldman's VSS on input the secret $\sigma = s^{-1}$ and with basis $G = g^m y^r$. Let α_0 be the commitment to the secret generated by the Feldman's VSS, i.e. if Alice is honest $\alpha_0 = G^\sigma \bmod p$.
2. The proxies run the verification phase of Feldman VSS (including the complaint phase). They reject the sharing if either Alice is disqualified during the verification of Feldman's VSS or $\alpha_0 \neq r \bmod q$.

DSS-ΣRecover The proxies run the reconstruction phase of Feldman's VSS to recover σ. They compute $s = \sigma^{-1} \bmod q$ and output r, s.

Fig. 7. A protocol for DSS VΣS

Theorem 4. *Protocol* DSS-VΣS *is a secure* VΣS *protocol for DSS with fault-tolerance* t*, for any* n, t *with* $n > 2t$.

Acknowledgments: We would like to thank Tal Rabin for several discussions over the topic of proactive RSA.

References

1. N. Asokan, V. Shoup and M. Waidner. Optimistic fair exchange of digital signatures. *EUROCRYPT'98.*
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computations. *STOC'88* pp.1–10, 1988.
3. M. Burmester. Homomorphism of secret sharing schemes: a tool for verifiable signature sharing. *EUROCRYPT'96*, pp.96–106. LNCS vol.1070.
4. David Chaum and Hans Van Antwerpen. Undeniable signatures. *CRYPTO'89*, pages 212–217. LNCS vol. 435.
5. D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. *STOC'88*, pp.11–19, 1988.

6. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. *FOCS'85*, pp.383–395, 1985.
7. D. Coppersmith, M. Franklin, J. Patarin and M. Reiter. Low-exponent RSA with related messages. *EUROCRYPT'96*, pp.1–9. LNCS vol.1070.
8. I. Damgard and T. Pedersen. New convertible undeniable signature schemes. *EUROCRYPT'96*, pp.372–386. LNCS vol. 1070.
9. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. *STOC'94*, pp.522–533, 1994.
10. W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, v. IT-22, no. 6, pp. 644-654, November 1976.
11. T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.
12. P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. *FOCS'87*, pp.427–437, 1987.
13. Y. Frankel, P. Gemmell, P. Mackenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. *FOCS'97*, pp.384–393.
14. M. Franklin and M. Reiter. Verifiable Signature Sharing. *EUROCRYPT'95*, pp.50–63. LNCS vol.921.
15. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. *CRYPTO'96*, pp.157–172. LNCS vol. 1109.
16. R. Gennaro, S. Jarecki, and T. Rabin. Securely revisiting distributed key generation. Manuscript.
17. R. Gennaro, H. Krawczyk and T. Rabin. RSA-Based Undeniable Signatures. *CRYPTO'97*, LNCS vol.1294.
18. O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. J. of Cryptology, Vol.9, No.3, pp.167–190, 1996.
19. O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems . *Journal of the ACM*, 38(1):691–729, 1991.
20. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. *STOC'87*, pp.218–229, 1987.
21. S. Goldwasser and S. Micali. Probabilistic Encryption. JCSS, 28(2), pp.270–299, April 1984.
22. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM. J. Computing*, 18(1):186–208, February 1989.
23. H. Krawczyk. Secret sharing made short. *CRYPTO'93*, pp.136–146, LNCS 773.
24. K.S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology*. vol.1, pp.95–105, 1988.
25. T. Pedersen. A threshold cryptosystem without a trusted party. *EUROCRYPT'91*, pp.522-526, LNCS vol. 547
26. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *CRYPTO'91*, pp.129-140, LNCS vol. 576
27. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. To appear in CRYPTO'98.
28. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.
29. Z. Shmuely. Composite Diffie-Hellman public-key generating systems are hard to break. Technion Dept. of Computer Science Technical Report no.356, 1985.
30. M. Stadler. Publicly Verifiable Secret Sharing. *EUROCRYPT'96*, pp.190–199, LNCS vol.1070.
31. E. Verheul and H. van Tilborg. Binding ElGamal. *EUROCRYPT'97*, LNCS 1233.