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

Dario Catalano[†] Rosario Gennaro[‡]

June 4, 1998

Abstract

Verifiable Signature Sharing (VΣS) was introduced by Franklin and Reiter in [20]. VΣS 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. In [20] efficient protocols were given for RSA, Rabin, ElGamal, Schnorr and DSS signatures. However their RSA and Rabin VΣS protocols were subsequently broken and their DSS VΣS lacks a formal proof of security.

We present new protocols for RSA, Rabin and DSS VΣS . Our protocols are efficient and provably secure and can tolerate the malicious behavior of up to half of the proxies.

The RSA VΣS scheme is based on a completely novel approach. The recipient of the signature will not share such string using conventional secret sharing schemes, but instead will simply encrypt it using a threshold cryptosystem, i.e. a public key whose matching secret key is kept shared at the proxies. She will then also provide the proxies with a proof that such ciphertext indeed contains a signature. The crux of the problem was to design a threshold cryptosystem that would make such proof efficient. We present several variants of our basic scheme, one of which requires *no* interaction between the recipient of the signature and the proxies to establish such a proof and one in which the reconstruction is completely non-interactive. The RSA VΣS scheme can be easily adapted to Rabin's signatures.

The DSS VΣS scheme is a modified version of the ElGamal VΣS scheme from [20] which allows for a proof of security.

The main application of VΣS is the incorporation of digital cash into multiparty protocols, e.g. cash escrow and secure distributed auctions. Our protocols thus provide simple, efficient and secure solutions for those applications. 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.

---

*A preliminary version of this paper appears in the proceedings of CRYPTO'98.

[†]Dipartimento di Matematica, Universita' di Catania. Viale A. Doria 6, 00100 Catania. Email: `dario@liotro.dipmat.unict.it`

[‡]I.B.M. T.J.Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598. Email: `rosario@watson.ibm.com`

1 Introduction

The concept of Verifiable Signature Sharing (VΣS) was introduced by Franklin and Reiter in [20]. VΣS 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 VΣS 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 [20] efficient protocols were given for RSA, Rabin, ElGamal, Schnorr and DSS signatures. However their RSA and Rabin VΣS protocols were subsequently broken in [8]. Also their DSS VΣS achieves only an heuristic form of security.

In [4] Burmester shows an unifying approach to VΣS 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.

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 [31, 28] and multiparty computation [29, 3, 6]. 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 [5]), i.e. signatures that can be verified only with the help of the signer. RSA-based undeniable signatures were recently introduced in [26]. Our RSA construction can be seen as an alternative to [26] (though admittedly a less efficient scheme).

Interestingly the structure of our RSA VΣS protocol can also be used to construct binding RSA cryptosystems. In [42] 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 [42] 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 Section 6.

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 (although our technique extends to other schemes like [2]). 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 [13, 14] 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 [31] 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 [20] 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 [32] as a way to shorten the size of shares in computationally secure secret sharing schemes. The specific idea of using threshold cryptosystems to "bootstrap" secret sharings appeared (exclusively for efficiency reasons) in [9, 22].

The idea of encrypting a signature and then proving in ZK that the ciphertext contains a valid signature has appeared in several places. In [10] 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 [41] 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 [16, 12, 11, 17, 24] and proactive [18, 19, 38] RSA signature schemes. In particular we combine the techniques of [19, 38] in order to improve efficiency.

It is important also to remind the reader of the difference between VΣS and related cryptographic objects (see [20] for a good discussion of this issue). VΣS is related but different from *threshold signature schemes*: in the latter the secret key is shared so that a set of n people can produce signatures if a threshold of t cooperates. Notice that a threshold signature scheme trivially gives a VΣS scheme when the signature sharer coincides with the signer. But in general this is not true. VΣS also bear some relationship to *fair public-key cryptosystems* (FPKC) [34] in which one has to share a secret key and prove that the shared secret is indeed the inverse of a given public key. Indeed both VΣS and FPKC are examples of *structured VSS* protocols (to use the terminology in [34]), i.e. verifiable secret sharing protocols in which the dealer has to prove some additional property about the secret being shared (and not just that such a unique secret exists).

2 Preliminaries

2.1 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.

2.2 Definition

We follow the ideas in the definition of VΣS presented in [20], 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 $\Sigma\mathsf{Share}$ 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.

2.3 Computational Assumptions

Our RSA $\mathsf{V\Sigma S}$ 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 [13, 14] over a composite modulus [33, 40]. 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 [30], 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.

2.4 Tools

We will use the polynomial-based t-out-of-n secret sharing due to Shamir [39]. 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 [7] solves these problems. Here we recall Feldman's VSS [15]. 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 [36] 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 dealer or of the players 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 [16, 12, 17, 24] and proactive [18, 19, 38] 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 Appendix A.

An example of a threshold cryptosystem for ElGamal over a prime modulus can be found in [35]. In this section we show how to extend that construction to a composite modulus.

A THRESHOLD CRYPTOSYSTEM FOR ELGAMAL OVER A COMPOSITE. There are several complications that arise from trying to generalize the approach in [35] 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 [19] for the application of proactive RSA.

3.1 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 discovered in [19]. 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 (this is evident from the proof of security). The protocol appears in Figure 1.
The following Lemma is from [19]. The proof is mostly based on a similar proof that appears in [38].

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

Proof First of all Feldman-Z_N-VSS inherits all the correctness properties of a VSS from Feldman's original VSS (that is, the shares determine a unique secret at the end of the sharing phase). We need however to prove that an adversary controlling t players has no information about the secret σ except for the value G^σ. We prove this by showing that the view of the adversary at the end of the sharing phase can be produced by a simulator on input the value $\alpha_0 = G^\sigma \bmod N$. Without loss of generality we can assume that the adversary corrupts the first t players $P_1, \ldots, P_t$.

First of all we show that the distribution of t shares of a secret σ with polynomial $f(z)$ is statistically indistinguishable from the distribution of t shares from sharing a random secret ρ with polynomial $r(z)$. We prove this by proving that with high probability there is a sharing of a random value $\rho \in [-N^2 \ldots N^2]$ using a polynomial r with integer coefficients

[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

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
$$G^{\sigma_i} = \alpha_0^L \prod_{j=1}^{t} \alpha_j^{i^j} \bmod N \tag{1}$$
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. $p > n^{t+2}L^2N^3$).

Figure 1: Feldman VSS over a composite modulus

in the same range as f such that $r(i) = f(i)$ for $i = 1, \ldots, t$. Define a t-degree polynomial $h(z)$ such that $h(0) = (\sigma - \rho)L$ and $h(1) = \ldots = h(t) = 0$. That is,

$$h(z) = \sum_{i=0}^{t} h(i) \prod_{j \neq i, j=0,\ldots,t} \frac{z-j}{i-j} = L(\sigma - \rho) \prod_{j=1,\ldots,t} \frac{z-j}{-j}$$

and the coefficient of z^i is

$$L(\sigma - \rho) \sum_{B \subseteq \{1,\ldots,t\}, |B|=i} \frac{\prod_{j \in B}(-j)}{\prod_{j=1\ldots,t}(-j)}$$

Note that because $L = n!$ this value is an integer. It can also be bounded in absolute value by

$$\sum_{B \subseteq \{1,\ldots,t\}, |B|=i} L(\sigma - \rho) \leq (\sigma - \rho) L \binom{t}{i} \leq \frac{(\sigma - \rho)Lt!}{i!(t-i)!} \leq (\sigma - \rho)Lt! \leq L^2 N^2$$

The desired polynomial is $r(z) = f(z) - h(z)$. $r(0) = \rho$, its coefficients are integers in the range $[-L^2N^3 - L^2N^2 \ldots L^2N^3 + L^2N^2]$, thus the probability that the coefficients are outside the legal range is $t \, \frac{2L^2N^2}{2(L^2N^3+L^2N^2)} \leq \frac{t}{N}$ which is negligible.

Now we have to show the full simulation of Feldman-Z_N-VSS. The simulator receives the value $\alpha_0 = G^\sigma$ and it has to produce an indistinguishable view for the adversary. The simulator shares a random value ρ with a random polynomial $r(z)$ and gives to the adversary the t shares $\sigma_i = r(i) \bmod N$ for $i = 1, \ldots, t$. As we proved above the distribution of these values is statistically close to the real one. Now the simulator has to produce the public output α_i's. Define the polynomial $f(z) = \sum_{i=0}^{t} \sigma_i \prod_{j \neq i} \frac{z-j}{i-j}$ where $\sigma_0 = L\sigma$. Clearly $f(i) = \sigma_i$ for $i = 1, \ldots, t$. We now need to show that we can efficiently compute G to the coefficients of $f(z)$. Rearranging the terms we can see that the coefficient of z^k is $a_k = \sum_{i=0}^{t} \frac{\lambda_{i,k}\sigma_i}{\prod_{j\neq i}(i-j)}$ where the $\lambda_{i,k}$ are known constants. the only problem now to compute G^{a_k} is the presence of the fractions since we cannot extract roots modulo N. But recall that we defined $G = G_0^{L^2}$. Thus

$$G^{a_k} = G^{\sum_{i=0}^{t} \frac{\lambda_{i,k}\sigma_i}{\prod_{j\neq i}(i-j)}} = \prod_{i=0}^{t} G^{\frac{\lambda_{i,k}\sigma_i}{\prod_{j\neq i}(i-j)}} = G^{\frac{\sigma_0 \lambda_{0,k}}{(-1)^t t!}} \prod_{i=1}^{t} G^{\frac{\lambda_{i,k}\sigma_i}{\prod_{j\neq i}(i-j)}} = (\alpha_0)^{\frac{L\lambda_{0,k}}{(-1)^t t!}} \prod_{i=1}^{t} (G_0^{\sigma_i \lambda_{i,k}})^{\frac{L^2}{\prod_{j\neq i}(i-j)}}$$

Notice that all the exponents are integers now. ■

3.2 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 [35] 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 [25]. 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 [38] and called *share-backup*.

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

3.3 Decryption Protocol

We are now left to show the decryption protocol. The approach is the same as the decryption protocol modulo a prime in [35], but it uses the techniques from [19, 38], 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 [24]. 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 $n > 2t$.*

Proof First of all we have to prove that the distribution of the public key generated by the protocol is "almost" the same as if it was generated by a centralized user. The distribution of Y is induced by the distribution of X mod $phi(N)$ (since $ord(G)$ is a divisor of $\phi(N)$). In the centralized case X is chosen in Z_N with uniform probability. This results in a distribution which is statistically close to uniform for X mod $\phi(N)$. So we need to prove

[3] [24] 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 $x_i \in_R [-N^2 \ldots N^2]$ and t random integers $a_{i1}, \ldots, a_{it} \in [-L^2N^3 \ldots L^2N^3]$ and consider the polynomial $f_i(z) = Lx_i + a_{i1}z + \ldots + a_{it}z^t$. 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
$$G^{x_{ji}} = \alpha_{j0}^L \prod_{k=1}^{t} \alpha_{jk}^{i^k} \bmod N \tag{2}$$
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 $(x_i, \{x_{ji}\}_{j \in Good})$.

Figure 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 [24] 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 $M = \frac{B}{\prod_{i \in Good} A_i} \bmod N$.

Figure 3: Threshold Decryption for ElGamal over a composite

that when X is generated by the protocol $X \bmod \phi(N)$ has a distribution which is also statistically close to uniform. Here $X = \sum_{i \in Good} x_i$, where some of the x_i's are under the control of the adversary. In other words we can write $X = X_{\mathcal{A}} + X_H \bmod \phi(N)$ where $X_{\mathcal{A}}$ is chosen by the adversary while X_H is determined by the honest players. It is important to notice that $X_{\mathcal{A}}$ can follow any arbitrary distribution but it is independent from X_H since the adversary decides on it at the end of the Pedersen's VSS when she has no information about the x_i's chosen by the honest players. Thus we can consider $X_{\mathcal{A}}$ as a fixed constant and it is enough to prove that X_H is distributed almost uniformly over $Z_{\phi(N)}$. Assume w.l.o.g. that the first $t+1$ players are the honest ones, then $X_H = \sum_{i=1}^{t+1} x_i \bmod \phi(N)$. Choose two arbitrary values Z_1 and Z_2 in $Z_{\phi(N)}$ and consider

$$\Delta_{Z_1,Z_2} = |Prob[X_H = Z_1] - Prob[X_H = Z_2]|$$

If a vector $(x_1, \ldots, x_{t+1})$ where each $x_i \in [-N^2, \ldots, N^2]$ generates Z_1 then the $(x_1, \ldots, x_t, x_{t+1} - Z_1 + Z_2)$ will generate Z_2. This vector is "legal" iff $x_{t+1} < N^2 - N < N^2 - \phi(N)$ which does not happen with probability $< \frac{1}{2N}$. Since we can "fix" any of the components of the vector it turns out that we can upper bound

$$\Delta_{Z_1,Z_2} < \left(\frac{1}{2N}\right)^{t+1}$$

Thus the difference between the distribution of X_H and the uniform distribution over $\phi(N)$ is at most $1/N^t$ which is negligible.

Also the correctness of the decryption protocol should be obvious. Since $X = \sum_{i \in Good} x_i$ we have that $A^X = \prod_{i \in Good} A^{x_i}$. Thus, unless a bad player passes the ZK proof with an incorrect partial decryption (which happens with negligible probability), the message is correctly decrypted.

We need to prove simulatability now. Assume w.l.o.g. that $\mathcal{A}$ corrupts players $P_1, \ldots, P_t$.

The simulator $\mathcal{S}$ works as follows. First it simulates a run of TEG-Key-Gen that produces Y as public key. It runs step 1 of TEG-Key-Gen for the good players and receives the sharing of the players controlled by $\mathcal{A}$. At this point the simulator knows which of those players are in the *Good* set and what are the values they shared (since $\mathcal{S}$ controls more than t players.) Let $Y_{\mathcal{A}} = \prod_{i \in Good, i \in \{1,\ldots,t\}} x_i$ be the component of the key generated by the players controlled by $\mathcal{A}$. $\mathcal{S}$ resets the adversary with the same random coins and restarts the simulation. In the new execution $\mathcal{A}$ will contribute $Y_{\mathcal{A}}$ again (this is one place where the non-rushing assumption comes into play). $\mathcal{S}$ will share a random value $\hat{x}_i$ for players P_i, $i = t+1, \ldots, n-1$ and run the simulation of Feldman-Z_N-VSS for player P_n with input the value $\hat{\alpha}_{0n} = \frac{Y}{Y_{\mathcal{A}} \prod_{i=t+1}^{n-1} x_i}$. Clearly this is a statistically close simulation of a real TEG-Key-Gen since the only difference is the statistical difference in the simulation of the behavior of player P_n.

Now we need to simulate one run of TEG-Decrypt with input the ciphertext A, B and output M. $\mathcal{S}$ computes $C = \frac{B}{M} \bmod N$ (i.e., $C = A^X$). For each player P_i, $i = t+1, \ldots, n-1$ the simulator broadcasts $\hat{A}_i = A^{\hat{x}_i} \bmod N$. For P_n the simulator broadcasts the value $\hat{A}_n = \frac{C}{\prod_{i \le t} A^{x_i} \prod_{t < i \le n} \hat{A}_i}$. For players $P_{t+1}, \ldots, P_{n-1}$ the simulator can run the ZK proof since he knows $\hat{x}_i$. For P_n the simulator runs a simulation of the ZK proof (notice that the theorem being proven is true, but the simulator does not know $\hat{x}_n$ such that $\hat{A}_n = A^{\hat{x}_n} \bmod N$ and

$\hat{\alpha}_{0n} = G^{\hat{x}_n} \bmod N$). Clearly this part of the simulation is indistinguishable from the real one since the only difference between the simulated and the real view is the ZK proof carried out by players P_n. But since the simulation of a ZK proof is indistinguishable from the real proof the whole view also is. ■

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 [20] 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.

4.1 Achieving Verifiability

A FIRST ATTEMPT. The above proof could be constructed using general zero-knowledge proof techniques, with a loss of efficiency. We tried to devise direct and efficient proofs for the most commonly used encryption schemes E (e.g. RSA itself and ElGamal) but we were not able to come up with protocols of acceptable[4] efficiency. Before describing our successful approach, let us give you a hint of the difficulties we encountered in this attempt. Assume that the proxies used RSA as their encryption scheme and that their public key was (N_P, e_P) (with matching decryption key being d_P). Alice would have created the ciphertext $C_S = S^{e_P} \bmod N_P$ and then must have proven in ZK that

$$(CS^{d_P} \bmod N_P)^{v_B} \bmod N_B = m$$

without knowing either the factorization of N_P or N_B! The interaction between the two groups $Z^*_{N_P}$ and $Z^*_{N_B}$ did not allow us to find an efficient way to prove the above equation. Similar problems would arise if the encryption scheme of the proxies was ElGamal with a prime modulus.

ACHIEVING VERIFIABILITY: A DIFFERENT APPROACH. The above problems were induced by the fact that the ciphertext C_S lived in a different, mathematically unrelated, group than the one in which the signature S is computed and verified. But this needs not necessarily

[4]The best protocol we came up with had soundness probability of 1/2 and thus needed to be repeated several times, causing a blow-up in the computational effort.

to be the case. Indeed the signature S (created under Bob's public key (N_B, v_B)) can be encrypted 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 [41] with some efficiency improvements (it is not necessary to repeat it several times.)

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) $T^{v_B} = rm^c \bmod N_B$.

Figure 4: A ZK Proof that an ElGamal encryption contains an RSA signature

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).

Proof *Completeness:* Obvious by inspection.

Soundness: Let's assume that the Prover has a non-negligible probability of making the Verifier accept in the above protocol. We need to prove that $S = \frac{B_S}{A_S^X} \bmod N_B$ where X is the secret key $Y = G^X \bmod N_B$.

If the Prover has a non-negligible probability of success, it means that after step 1 she must be able to answer a non-negligible fraction of all possible challenges c. In particular the Prover on the same first message can answer two challenges[5] c_1, c_2 such that $c_2 - c_1$

[5] Notice that we are not constructing an extractor for S so it is not necessary to constructively find such a pair of challenges, but only to show that they exist

is relatively prime to $\phi(N_B)$. Let $d = (c_2 - c_1)^{-1} \bmod \phi(N_B)$. Let L_i, T_i be the answer to challenge c_i for $i = 1, 2$. Because of tests (3a) and (3b) we know that

$$T_i = \frac{B_R}{A_R^X}\left(\frac{B_S}{A_S^X}\right)^{c_i} \bmod N_B \tag{3}$$

From test (3c) we know that

$$\left(\frac{T_2}{T_1}\right)^{v_B} = m^{c_2 - c_1} \bmod N_B$$

Raising the above equation to the product ds_B (where $s_B = v_B^{-1} \bmod \phi(N_B)$) we get

$$\left(\frac{T_2}{T_1}\right)^{d} = S \bmod N_B$$

but plugging in Equation (3) we get that $S = \frac{B_S}{A_S^X} \bmod N_B$.

Zero-Knowledge: We need to show a simulator for the honest Verifier. The simulator first chooses $\hat{c}$ uniformly at random in Z_{N_B}. It also chooses $\hat{T}$ uniformly at random in $Z_{N_B}^*$ and sets $\tau = \hat{T}^{v_B} \bmod N_B$. It then sets $\hat{r} = \tau m^{-\hat{c}} \bmod N_B$.

The simulator then chooses $\hat{L}$ uniformly at random in $Z_{N_B^3}$. It then sets $\hat{A_R} = G^{\hat{L}} A_S^{-\hat{c}}$ and $\hat{B_R} = Y^{\hat{L}} T B_S^{-\hat{c}}$. The simulated transcript is then $\hat{r}, \hat{A_R}, \hat{B_R}, \hat{c}, \hat{L}, \hat{T}$.

All the quantities have exactly the same distribution as in the real execution except for the difference between L and $\hat{L}$ (and the values induced by the choice of L vs. $\hat{L}$). If we show that the distribution of $\hat{L}$ is statistically indistinguishable from the one of L we are done. Notice that $\hat{L}$ is uniformly distributed in $[0, \ldots, N_B^3]$ while L is uniformly distributed in the shifted interval $[cK, \ldots, N_B^3 + cK]$. This means that the distance between the two distributions is exactly $\frac{2cK}{N_B^3}$ which is bounded by $\frac{2}{N_B}$ which is negligible. ■

To make the above protocol ZK against any verifier, known techniques [27] can be used. In particular the Verifier can commit to the challenge before the Prover speaks.

4.2 The Basic Protocol

OUTLINE. 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 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. In the next section we will show an alternative protocol which uses the full threshold cryptosystem in order to improve the efficiency of the scheme for Alice.

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 $S = \frac{B_S}{A_S^X} \bmod N_B$.

Figure 5: Basic RSA VΣS 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 $n > 2t$.*

Proof Completeness is obvious by inspection of the protocol.

To prove soundness first we notice that since the proxies are connected by a broadcast channel, the honest ones will be in agreement as to accept or reject the sharing. If they all accept the sharing it means that

- Alice passed the ZK proof, thus the ciphertext contains Bob's signature on the message *and*
- Alice passed the sharing phase of Feldman-Z_N-VSS which means that the proxies have shares of the correct decryption key which will allow them to reconstruct the signature later.

We are left with proving the security of the scheme. Assume w.l.o.g. that the adversary $\mathcal{A}$ corrupts[6] the first t proxies $P_1, \ldots, P_t$. Here is a simulator $\mathcal{S}$ for RSA-ΣShare-1 (rounds are numbered as in the protocol): recall that $\mathcal{S}$ works on input m, N_B, v_B but *not* the signature S:

1. $\mathcal{S}$ chooses uniformly at random
   - $\hat{r} \in Z_{N_B}$ and sets $\hat{G} = m^{\hat{r}} \bmod N_B$.
   - $\hat{X} \in [-N_B^2, \ldots, N_B^2]$ and sets $\hat{Y} = \hat{G}^{\hat{X}} \bmod N_B$.
   - $\hat{K} \in Z_{N_B}$ and sets $\hat{A}_S = \hat{G}^{\hat{K}} \bmod N_B$ and $\hat{B}_S = \hat{Y}^{\hat{K}} \bmod N_B$ (i.e. an encryption of 1 since $\mathcal{S}$ does not know the signature).

   She sends to the proxies $m, N_B, v_B, \hat{r}, \hat{Y}, \hat{A}_S, \hat{B}_S$. The proxies compute $\hat{G} = m^{\hat{r}} \bmod N_B$.
2. $\mathcal{S}$ runs the simulation of the ZK proof EGRSA-ZK-Proof on input $m, N_B, v_B, \hat{G}, \hat{Y}, \hat{A}_S, \hat{B}_S$.
3. $\mathcal{S}$ runs Feldman-Z_N-VSS for $\hat{X}$ as the dealer with the proxies.

Why is the adversary's view from the simulation indistinguishable from the real one? There is only one difference: in the real execution A_S, B_S is an encryption of S, while in the simulated one $\hat{A}_S, \hat{B}_S$ is an encryption of 1. To distinguish between the real view and the simulated view would imply to distinguish between encryption of two different messages in the ElGamal encryption scheme. which would contradict the semantic security of the ElGamal scheme. Since the basis G is chosen as a power of m this means that both messages (1 and S) are in the group generated by G, thus the DDH implies the semantic security of ElGamal which concludes the proof.

A remark is in order with respect to the simulation of the ZK proof in step (2). The simulation of a ZK proof is guaranteed to be indistinguishable from the real execution when the theorem being proven is true. In this case we are running the simulator on a false theorem (since $\hat{A}_S, \hat{B}_S$ is *not* an encryption of the signature of m. However notice that under the DDH the simulation of the ZK proof must also be indistinguishable when the theorem is false, otherwise such simulator could be used to break the DDH. ■

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

$$\frac{B_S^{v_B}}{(A_S^{v_B})^X} = \frac{(Y^K \cdot S)^{v_B}}{(G^K)^{X \cdot v_B}} = \frac{Y^{K \cdot v_B} \cdot S^{v_B}}{Y^{K \cdot v_B}} = m \bmod N_B \tag{4}$$

[6] The case in which Alice is corrupted is easily simulated, $\mathcal{S}$ just follows the protocol for the honest proxies.

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.

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).

Figure 6: Alternative RSA VΣS Protocol

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 $n > 2t$.*

Proof The correctness of the protocol is based on the correctness of TEG-Decrypt. Since an honest Alice would encrypt the signature, clearly at the end of RSA-ΣRecover the signature will be decrypted.

The soundness of the protocol is also based on the correctness of TEG-Decrypt. Equation (4) is a necessary and sufficient condition for (A_S, B_S) to decrypt to the signature. So either all the proxies reject if equation (4) is not satisfied or they will all accept, but then the signature will be decrypted successfully at the end of RSA-ΣRecover .

We are left with proving the security of the scheme. Assume w.l.o.g. that the adversary $\mathcal{A}$ corrupts[7] the first t proxies $P_1, \ldots, P_t$. Here is a simulator $\mathcal{S}$ for RSA-ΣShare-2 (rounds are numbered as in the protocol): recall that $\mathcal{S}$ works on input m, N_B, v_B but *not* the signature S:

1. $\mathcal{S}$ just sends m, N_B, v_B and a randomly chosen $\hat{r}$ to the proxies.
2. $\mathcal{S}$ runs TEG-Key-Gen for the good players, where the basis is set to be $\hat{G} = m^{\hat{r}} \bmod N_B$. In particular at the end, the values $\hat{\alpha}_{0i} = \hat{G}^{\hat{x}_i}$ are public. Notice that at the end $\mathcal{S}$ knows the shares $\hat{x}_i$ of the secret key of *all* the players.

[7]The case in which Alice is corrupted is easily simulated, $\mathcal{S}$ just follows the protocol for the honest proxies.

3. $\mathcal{S}$ encrypts the value 1 (since it does not know the real signature S) by choosing a random $\hat{K} \in_R Z_N$ and broadcasting $\hat{A}_S = \hat{G}^{\hat{K}} \bmod N$ and $\hat{B}_S = Y^{\hat{K}} \bmod N$.

4. At this point the proxies run TEG-Decrypt on $(A^* = \hat{A}_S^{v_B}, B^* = \hat{B}_S^{v_B})$. In order to get m as the result $\mathcal{S}$ will "cheat". Recall the details of TEG-Decrypt. Each proxy will broadcast $A_i^* = (A^*)^{\hat{x}_i}$ and prove in ZK that it is correct with respect to $\hat{G}^{\hat{x}_i}$ generated during TEG-Key-Gen. Notice that $\mathcal{S}$ knows the values A_i^* for $i = 1, \ldots, t$. For the proxies P_i, $i = t+1, \ldots, n-1$ the simulator will broadcast $A_i^* = (A^*)^{\hat{x}_i}$ and successfully prove in ZK their correctness. For P_n however the simulator will broadcast a value that will make the decryption equal to m: such value is $A_n^* = \frac{B^*}{m \cdot \prod_{i<n} A_i^*}$. For P_n, the simulator will run the simulation of the ZK proof of correctness.

Why is the adversary's view from the simulation indistinguishable from the real one? There are two differences:
(a) in the real execution A_S, B_S is an encryption of S, while in the simulated one $\hat{A}_S, \hat{B}_S$ is an encryption of 1.
(b) in the real execution the following values $(A = G^K, \alpha_{0n} = G^{x_n}, A_n = A^{x_n} = G^{Kx_n})$ constitute a Diffie-Hellman triplet. In the simulated execution $(\hat{A} = \hat{G}^{\hat{K}}, \hat{\alpha}_{0n} = \hat{G}^{\hat{x}_n}, A_n^*)$ is not necessarily a DH triplet.
To distinguish between the real view and the simulated view would imply to distinguish either one of these two case. But this contradicts the DDH (for case (a) the reduction is based again on the fact that $1, S$ are in the group generated by G). ■

4.4 Comments, Variants and Optimizations

RABIN'S SIGNATURES. Both the protocols RSA-VΣS described above can easily be adapted to Rabin's signatures by simply putting $v_B = 2$.

NON-INTERACTIVE RECONSTRUCTION. Reconstructing the secret key has the advantage of making the recover protocol non-interactive. It is worth pointing out that even in RSA-VΣS-2 the recover stage can be implemented by reconstructing the secret key.

NO-FAULT MODE. Under normal operations, it may be very likely that there are no faults in the system. Let us look at the behavior of our protocol in the case all players are honest. There will be no complaints in Feldman-Z_N-VSS. If we choose to expose the secret key in RSA-ΣRecover-2, it can be done without running n recover phases of Feldman-Z_N-VSS since the players will first reveal their correct x_i. All runs of TEG-Decrypt will successfully complete without invoking any share-backup. This is very important for step 4 of RSA-ΣShare and for RSA-ΣRecover-2. Moreover, in the specific case of RSA-ΣRecover-2 we can also postpone the ZK proofs of correctness of partial decryptions to *after* the alleged signature has been reconstructed. Indeed the proxies can verify (using v_B) if they recovered the valid signature S. If so they are done. Otherwise, it means that some of the proxies are contributing incorrect partial decryptions and one must enforce the ZK proofs. Thus with this approach, under normal operation of the system (i.e. when there are no faults), reconstruction using TEG-Decrypt is also non-interactive and requires no ZK proofs. The efficiency gains under normal operation of the system are quite substantial (see below).

Saving the Key Generation. In both protocols the generation of the ElGamal key (either by Alice or jointly by the proxies) can Steps 1 and 2 can be moved to a preprocessing stage to make the actual VΣS protocol faster.

Amortizing the Key Generation In protocol RSA-VΣS-2 the key generation may also be amortized over several VΣS when the signatures being shared are all under the same modulus N_B. Consider the scenario of an electronic auctions in which all the bids must be done by verifiably sharing a check drawn from the same bank. In this case the proxies will receive several signatures all under the same public key N_B, v_B and the key generation protocol TEG-Key-Gen can be done once for all the sharings. This means that all the signatures are encrypted under the same key. In turn this implies that if only a single signature has to be reconstructed then the reconstruction *must* use the decryption protocol instead of recovering the key.

One word of caution however is needed in this case. Since now the basis G is common to all the sharings the security of the scheme is based on the semantic security of ElGamal over a composite when the message space is all of $Z^*_{N_B}$.

Efficiency. The protocols are very efficient especially for Alice. We summarize the computational requirements of the protocol in Figure 7 reporting upper bounds on the number of modular exponentiations (in $Z^*_{N_B}$) required by each player. The numbers in parenthesis represent the numbers under normal operation of the system (i.e. no-fault mode).

The storage requirements for the proxies are as follows. A share of the signature S is basically the ciphertext A_S, B_S plus the shares of the secret key X. In RSA-VΣS-1 this totals to $\approx 5 \log N_B$) which is optimal. In RSA-VΣS-2 the share size is $\approx 3n \log N_B$. This is not optimal because of the multiplicative factor of n due to the share-backup approach in TEG-Key-Gen.

RSA-ΣShare-1	Preprocessing	Actual Sharing
Alice	$2n+3$	9
Proxy	$2n+4$ (4)	10

RSA-ΣShare-2	Preprocessing	Actual Sharing
Alice	none	3
Proxy	$5n^2$ ($9n$)	$3n^2$ ($5n$)

RSA-ΣRecover-1	Exposing secret key
Proxy	$3n$ noninter.

RSA-ΣRecover-2	Running TEG-Decrypt	Exposing secret key
Proxy	$3n^2$ inter. (3, noninter.)	$3n^2$ noninter. ($3n$ noninter.)

Figure 7: Efficiency numbers (n = number of proxies)

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 it down via SHA-1, i.e. $m = SHA$-$1(M)$. 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 [20]. We follow the same provably secure approach used in [20] to verifiably share an ElGamal signature. The main difference between a (plain) ElGamal signature and DSS[8] 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 [20] 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 [20] 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 $r^* = G^\sigma \bmod p$. 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 (this was also noticed in [23] and in [43]). The full protocol appears in Figure 8.

A REMARK ABOUT FAULT-TOLERANCE. By using Feldman's VSS with the extra complaint phase we achieve fault-tolerance t for $n > 2t$. In [20] the bound $n > 3t$ was claimed because a simplified version of Feldman's VSS without complaints was used. It is easy to see that all the discrete-log based protocols in [20] achieve the bound $n > 2t$ if Feldman with complaints is used.

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

Proof Completeness is obvious since the protocol inherits the property of successful reconstruction from Feldman's VSS.

Soundness is also quite clear. If the proxies accept, then the value σ shared satisfy the verification equation of DSS with message m and random coins r, i.e. $r = (g^m y^r)^\sigma \bmod p \bmod q$. So the value reconstructed in DSS-ΣRecover must be a correct signature.

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

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.

Figure 8: A protocol for DSS VΣS

Security relies once again on simulation. Assume w.l.o.g. that the adversary corrupts the first t proxies. The simulator $\mathcal{S}$ on input p, q, g, y, m works as follows. It chooses a random $k \in Z_q$ and computes $r^* = g^k \bmod p$ and $r = r \bmod q$. It simulates Feldman's VSS with $G = g^m y^r$ as basis and r^* as the public commitment to the secret being shared. I.e. it generates t values $\sigma_1, \ldots, \sigma_t$ at random in Z_q. Define by $f(z) = \sum_{i=0}^{t} a_i z^i$ the t-degree polynomial in Z_q such that $f(0) = DLog_G\, r^*$ and $f(i) = \sigma_i$. We cannot reconstruct the coefficients of $f(z)$ unless we solve discrete log, but we can compute the values $\alpha_i = G^{a_i} \bmod p$ via "interpolation in the exponent" (in particular $\alpha_0 = r^*$). The simulated view of the adversary is defined as the t shares $\sigma_1, \ldots, \sigma_t$ and the public values $\alpha_0, \ldots, \alpha_t$. It is easy to see that the simulated view has the same distribution of the real one. ■

6 Other Applications

6.1 A new RSA-based undeniable signature

A NEW UNDENIABLE SIGNATURE SCHEME. Undeniable signatures were introduced in [5]. Informally an undeniable signature needs the cooperation of the signer to be verified. This may be useful in scenarios in which the signer wants to have some control on the distribution of authenticated data (e.g. software, where the signer wants to collect a licensing fee from users). An undeniable signature comprises three components: a signing algorithm, a confirmation protocol that allows the signer to confirm legitimate signatures and a disavowal protocol that allows the signer to prove that an alleged signature is false. An undeniable signature is called convertible if the signer can publish a short piece of information to convert all undeniable signatures in publicly verifiable ones. In [10] a general paradigm was proposed to implement convertible undeniable signatures. The signer chooses a traditional signature scheme S with public verification key VK and secret signing key SK. He also chooses an asymmetric encryption scheme E, publishes the encryption key EK and keeps the decryption key DK private. To undeniably sign a message m, the signer publishes $C_S = E_{EK}(S_{SK}(m))$ i.e. an encryption of the signature. Then one adds ZK proofs to prove or disprove that the ciphertext contains a valid signature. To convert the signatures it is sufficient to publish DK. In [10] efficient protocols were presented for ElGamal-like signatures but not for RSA ones.

Our techniques yield an efficient RSA-based undeniable scheme that follows the [10] paradigm. The signer uses RSA to sign and the ElGamal over the same composite to encrypt the signature. The confirmation protocol is EGRSA-ZK-Proof shown in Figure 4. The disavowal protocol is described in Figure 9. It needs to be repeated ℓ times to achieve probability of error $2^{-\ell}$, however this is OK for the denial protocol since it will be invoked rarely. Notice that all the prover needs to know in order to participate in either the confirmation or the denial protocol is the decryption key X (which means that the signer can delegate both functions to someone else, and still reserve to himself the power to sign.)

The protocol in Figure 9 is only honest-verifier ZK since a cheating verifier could learn information about a ciphertext (A_R, B_R) which she did not generate. To make it ZK against any verifier standard techniques may be used: the prover would not reveal the answer $\bar{b}$ directly but he would rather commit to it. Then the Verifier would show that she constructed (A_R, B_R) properly. At that point the Prover would open the commitment to $\bar{b}$.

Lemma 3 *The protocol* EGRSA-ZK-Proof *is a zero-knowledge proof that (A_S, B_S) is not an ElGamal encryption under the key (N_B, G, Y) of the signature S of m under the key (N_B, v_B).*

Proof Completeness: since (A_S, B_S) is *not* an encryption of the signature of m, it means that C_{1-b} is not an encryption of the signature of mr. So the Prover will be able to guess b correctly all the time.

Soundness: if (A_S, B_S) is an encryption of the signature of m, both C_b and C_{1-b} will be encryptions of signatures of the first component. Thus the Prover will be able to guess b only with probability $\frac{1}{2}$.

Zero-Knowledge: The simulator chooses a bit $\hat{b}$ uniformly at random. He generates $\hat{R} \in_R Z^*_{N_B}$ and $\hat{J} \in_R Z_{N_B}$ uniformly at random. He sets $\hat{r} = \hat{R}^{v_B} \bmod N_B$, $\hat{A}_R = G^{\hat{J}} \bmod N_B$ and $\hat{B}_R = Y^{\hat{J}} \cdot \hat{R} \bmod N_B$. Finally he sets $\hat{C}_b = (\hat{r}, \hat{A}_R, \hat{B}_R)$ and $\hat{C}_{1-b} = (\hat{r}\hat{m}, \hat{A}_R A_S, \hat{B}_R B_S)$. The transcript of the simulation is $\hat{C}_b, \hat{C}_{1-b}, \hat{b}$ which clearly has the exact same distribution as in the real case. ■

In [26] the first RSA-based undeniable signature was presented. Our scheme is less efficient than [26] and also relies on more assumption (e.g. the security of the ElGamal encryption).

6.2 Binding RSA

In [42] 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

EGRSA-ZK-Denial

Secret input for the Prover: The decryption key X.
Input for all: The message m, the signer's RSA public key N_B, v_B, the encryption public key $G, Y = G^X$, the alleged encrypted signature $(A_S = G^K, B_S = Y^K \cdot S)$.

1. The Verifier 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 Verifier encrypts R under the ElGamal key of the proxies, i.e. she chooses $J \in_R Z^*_{N_B}$ and sets $A_R = G^J \bmod N_B$ and $B_R = Y^J \cdot R \bmod N_B$. She flips a random bit b and sends $C_b = (r, A_R, B_R)$ and $C_{1-b} = (rm, A_R A_S, B_R B_S)$ to the prover.
2. The Prover decrypts the last two components of both C_0 and C_1 and tests which one contains a correct signature of the first component. Let $\bar{b}$ the answer. The Prover sends to the Verifier $\hat{b}$. The Verifier accepts if $\bar{b} = b$.

Figure 9: A ZK Proof that an ElGamal encryption does not contain an RSA signature

public key of a third party and prove in (non-interactive) ZK that the two ciphertexts contain the same message. In [42] a scheme for binding ElGamal was presented but the problem of binding RSA left open. The main motivation of [42] was to establish a fraud-detectable key-recovery system: the third party would be a key recovery agent and because the proof is publicly verifiable fraudulent messages would be detected. In [37] however an attack is presented in which two cooperating players can produce seemingly binding ciphertexts that would not be decrypted by the key recovery authority. If at least one of the parties is honest, [37] notices that much more efficient solutions can be devised by having the receiver check that the key recovery agent can indeed decrypt. This seems to deprive this approach of its main application area.

However it is possible that in other models the binding approach may still be interesting. Consider for example a model in which the receiver wants his ISP to check that incoming encrypted messages are lawfully constructed and does not want to spend valuable resources by doing the check himself. In general binding cryptosystems could still be useful in other areas because of their interesting structure.

Our RSA VΣS scheme can be used to construct binding RSA public key encryptions. Bob's public key for encryption is (N_b, e_B) and the third party uses an ElGamal key $(N_B, G, Y = G^X)$. If Alice wants to send a message m to Bob using a binding cryptosystem she would encrypt m twice constructing $A = G^K$, $B = Y^K \cdot m$ and $C = m^{e_B} \bmod N$ and prove in ZK that the message contained in C is the same as the message contained in (A, B). She can use the proof in Figure 4 for this purpose (where C is the message and m is its secret signature). The proof can be made non-interactive in the random oracle model, by computing c via a suitable hash of the theorem statement $(N_b, e_B, G, Y, A, B, C)$ and of the first message of the proof (r, A_R, B_R).

7 Conclusion and an Open Problem.

We presented new, efficient and provably secure VΣS protocols for RSA, Rabin and DSS signature schemes. This problem was open since the introduction of the concept of VΣS in [20] as their RSA/Rabin protocols were subsequently broken and their DSS protocol was not provably secure. Various applications of our results have also been presented.

We leave the reader with an open problem: our RSA VΣS scheme is provably secure under an assumption which is not known to be weaker than the RSA assumption itself. It would be nice to have an RSA VΣS scheme which is provably secure just under the RSA assumption.

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. Bellare and P. Rogaway. The exact security of digital signatures. How to sign with RSA and Rabin. EUROCRYPT'96, pp.399–416, Lecture Notes in Computer Science vol.1070

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

[4] M. Burmester. Homomorphism of secret sharing schemes: a tool for verifiable signature sharing. EUROCRYPT'96, pp.96–106. Lecture Notes in Computer Science vol.1070.

[5] David Chaum and Hans Van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 212–217. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[6] D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. In *Proceeding 20th Annual Symposium on the Theory of Computing*, pages 11–19. ACM, 1988.

[7] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proc. 26th IEEE Symp. on Foundations of Comp. Science*, pages 383–395, Portland, 1985. IEEE.

[8] D. Coppersmith, M. Franklin, J. Patarin and M. Reiter. Low-exponent RSA with related messages. EUROCRYPT'96, pp.1–9. Lecture Notes in Computer Science vol.1070.

[9] R. Cramer, R. Gennaro and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. European Transactions of Telecommunications, Vol.8, No.5, September 1997.

[10] I. Damgard and T. Pedersen. New convertible undeniable signature schemes. In *Eurocrypt'96*, pages 372–386. Lecture Notes in Computer Science No. 1070.

[11] 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, Santa Fe, 1994. IEEE.

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

[13] 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.

[14] 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.

[15] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In *Proceeding 28th Annual Symposium on the Foundations of Computer Science*, pages 427–437. IEEE, 1987.

[16] Y. Frankel. A practical protocol for large group oriented networks. EUROCRYPT'89, pp.56–61, Lecture Notes in Computer Science vol.434

[17] Y. Frankel, P. Gemmell, and M. Yung. Witness-based Cryptographic Program Checking and Robust Function Sharing. In *Proc. 28th Annual Symp. on the Theory of Computing*, pages 499–508. ACM, 1996.

[18] Y. Frankel, P. Gemmell, P. Mackenzie, and M. Yung. Proactive RSA. In B. Kaliski, editor, *Advances in Cryptology — Crypto '97*, pages 440–454, Berlin, 1997. Springer-Verlag. Lecture Notes in Computer Science No. 1294.

[19] Y. Frankel, P. Gemmell, P. Mackenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. In FOCS'97, pp.384–393.

[20] M. Franklin and M. Reiter. Verifiable Signature Sharing. EUROCRYPT'95, pp.50–63. Lecture Notes in Computer Science vol.921.

[21] M. Franklin and M. Reiter. The design and implementation of a secure auction service. 1995 IEEE Symposium on Security and Privacy.

[22] J. Garay, R. Gennaro, C. Jutla and T. Rabin. Secure Distributed Storage and Retrieval. 1997 Workshop on Distributed Algorithms (WDAG'97), Lecture Notes in Computer Science vol.1320.

[23] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In Ueli Maurer, editor, *Advances in Cryptology — Eurocrypt '96*, pages 354–371, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science No. 1070.

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

[25] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. How to securely generate a key in a distributed fashion. Manuscript.

[26] R. Gennaro, H. Krawczyk and T. Rabin. RSA-Based Undeniable Signatures. CRYPTO'97, Lecture Notes in Computer Science vol.1294.

[27] 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.

[28] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but the Validity of the Assertion, and a Methodology of Cryptographic Protocol Design. In *Proceeding 27th Annual Symposium on the Foundations of Computer Science*, pages 174–187. ACM, 1986.

[29] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *19th ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[30] S. Goldwasser and S. Micali. Probabilistic Encryption. JCSS, 28(2), pp.270–299, April 1984.

[31] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM. J. Computing*, 18(1):186–208, February 1989.

[32] H. Krawczyk. Secret sharing made short. In D. Stinson, editor, *Advances in Cryptology — Crypto '93*, pages 136–146, Berlin, 1993. Springer-Verlag. Lecture Notes in Computer Science No. 773.

[33] K.S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology.* vol.1, pp.95–105, 1988.

[34] S. Micali. Fair Public-Key Cryptosystems. CRYPTO'92, pp.113–138.

[35] T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Berlin, 1991. Springer-Verlag.

[36] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Advances in Cryptology—CRYPTO'91*, pp.129-140, volume 576 of *Lecture Notes in Computer Science.*

[37] B. Pfitzmann and M. Waidner. How to break fault-detectable key recovery. Operating Systems Review 32/1 (1998) 23-28.

[38] T. Rabin. A Simplified Approach to Threshold and Proactive RSA. To appear in CRYPTO'98.

[39] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[40] Z. Shmuely. Composite Diffie-Hellman public-key generating systems are hard to break. Technion Dept. of Computer Science Technical Report no.356, 1985.

[41] M. Stadler. Publicly Verifiable Secret Sharing. EUROCRYPT'96, pp.190–199, Lecture Notes in Computer Science, vol.1070

[42] E. Verheul and H. van Tilborg. Binding ElGamal. Proceedings of EUROCRYPT'97, Lecture Notes in Computer Science vol. 1233.

[43] Y. Zheng. Digital SignCryption. CRYPTO'97, Lecture Notes in Computer Science vol.1294.

A Threshold Cryptosystems

A public key scheme E is defined by three algorithms:

Key-Gen key generation: a randomized algorithm that takes a security parameter as input and returns a pair (EK, DK) where EK is the public encryption key and DK is the secret decryption key.

Encrypt takes as input a message M and the public key EK and returns a ciphertext C also denoted as $E_{EK}(M)$.

Decrypt takes as input a ciphertext $C = E_{EK}(M)$ and the private decryption key DK and returns M.

A *threshold cryptosystems* 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 and as secret input his share DK_i and returns as public output a message M

For any adversary $\mathcal{A}$ that corrupts at most t players the following conditions must be met:

correct key generation T-Key-Gen generates keys with the same a distribution which is computationally indistinguishable from Key-Gen.

correct decryption on input $C = E_{EK}(M)$, T-Decrypt returns as output M

simulatability Recall that we define the view $\mathcal{V}$ of the adversary $\mathcal{A}$ during a protocol as the set of messages sent and received by the bad players during a run of the protocol. Consider an execution of T-Key-Gen that generated Y as an output followed by executions of T-Decrypt on input C_i and output M_i. Let $\mathcal{V}$ be the view of the adversary during that protocol. Then there exists an algorithm $\mathcal{S}$ called the *simulator* which on input Y, C_i, M_i and black-box access to $\mathcal{A}$ produces output strings with a distribution which is computationally indistinguishable from $\mathcal{V}$.