J. Cryptology (2000) 13: 273–300
DOI: 10.1007/s001459910011

Journal of
CRYPTOLOGY

Robust and Efficient Sharing of RSA Functions[*]

Rosario Gennaro and Tal Rabin
IBM T.J. Watson Research Center,
PO Box 704, Yorktown Heights, NY 10598, U.S.A.
{rosario,talr}@watson.ibm.com

Stanislav Jarecki
Laboratory of Computer Science, Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139, U.S.A.
stasio@theory.lcs.mit.edu

Hugo Krawczyk
Department of Electrical Engineering, Technion,
Haifa 32000, Israel
hugo@ee.technion.ac.il
and
IBM T.J. Watson Research Center,
PO Box 704, Yorktown Heights, NY 10598, U.S.A.

Communicated by Oded Goldreich

Received 21 March 1997 and revised 28 September 1999

Abstract. We present two efficient protocols which implement robust threshold RSA signature schemes, where the power to sign is shared by N players such that any subset of $T+1$ or more signers can collaborate to produce a valid RSA signature on any given message, but no subset of T or less corrupted players can forge a signature. Our protocols are robust in the sense that the correct signature is computed even if up to T players behave in an arbitrarily malicious way during the signature protocol. This, in particular, includes the cases of players who refuse to participate or who introduce erroneous values into the computation. Our robust protocols achieve optimal resiliency as they can tolerate up to $(N-1)/2$ faults, and their efficiency is comparable with the efficiency of the underlying threshold RSA signature scheme. Our protocols require RSA moduli which are the product of two safe primes, and that the underlying (centralized) RSA signature scheme is unforgeable. Our techniques also apply to the secure sharing of the RSA decryption function.

We show that adding robustness to the existing threshold RSA schemes reduces to solving the problem of how to verify an RSA signature without a public verification

[*] A preliminary version of the paper appeared in *Crypto* ’96 [GJKR1].

exponent. Our technical contribution focuses on solving this problem. Once a solution to this problem exists it can be applied to any existing threshold RSA signature scheme in order to achieve a *robust* threshold safe prime RSA signature scheme.

Key words. RSA signatures, Threshold RSA, Threshold cryptography.

1. Introduction

The idea of distributed signature schemes is to depart from the single-signer approach in which one person is the sole holder of a secret key, and to allow a group of people to "hold" the key in such a manner that they can, as a group, produce signatures, yet no person on his own can generate a signature. The signature which is generated by the group is the same as if it were generated by a single signer. This ensures the important property that the verifier of such a signature does not need to be aware of the method (centralized or distributed) used to generate the signature.

We say that a distributed signature scheme is a (T, N)-*threshold signature scheme*, if given a message m, and a group of N players, where each one holds a part of the secret key, any subset of $T + 1$ or more players can generate the signature on m. We say that the scheme is *secure or unforgeable* if no coalition of T (or fewer) players can produce a valid signature on a new message, even after the system has produced signatures for many messages. Furthermore, a (T, N)-threshold signature scheme is *robust* (or *fault-tolerant*) if it can correctly compute signatures, even in the presence of up to T arbitrarily malicious players.

Note that a simple secret sharing of the secret key and its reconstruction by a player at signing time would not satisfy our requirement, as it allows future signatures to be generated by a single player (i.e., such reconstruction would create a single point of failure).

Robust threshold signature schemes have very important applications, and we exemplify this briefly. The secret key of a global certification authority will be an attractive target for attacks. If the key is held in one site, then once this site is broken into, the key is exposed. Yet, if the key is distributed among N sites, using a (T, N)-threshold signature scheme, then one needs to break into $T + 1$ sites in order to learn the key or forge signatures. In addition, once a site is broken into, it might exhibit arbitrary performance, yet the system should still be able to generate signatures. Hence, we increase the security and availability of a system by distributing the secret key. Furthermore, if the key is held on a single site, then signatures cannot be generated if this site crashes. Note that the trivial solution of key replication solves the availability problem, yet it requires more sites to hold the full key, hence adding to the vulnerability of the system. A robust threshold signature protocol can, in particular, tolerate up to T such crashes (and even arbitrary malicious actions), thus increasing the availability of the signature operation without decreasing its security.

Threshold signatures are part of the general approach known as *threshold cryptography* introduced through the works of Desmedt [Des1], Boyd [Bo], Croft and Harris [CH], and Desmedt and Frankel [DF1]. For an early survey on threshold cryptography see [Des2]. Solutions for the case of the RSA signature scheme are especially important because of its widespread use (a de facto standard). In addition, since in the RSA cryptosystem the signing algorithm coincides with the decryption algorithm, solutions to shared RSA signatures usually lead to shared RSA decryption procedures which have various applications, e.g., key escrow (see [Mi]) and secure distributed storage (see [GGJR]). Desmedt and Frankel initiated the study of threshold RSA [DF2], which was followed by De Santis, et al. [DDFY]. These papers provide solutions for the problem of threshold RSA, however, they lack the robustness property.

The basic paradigm followed by known threshold signature schemes is as follows. Each player P_i has a share d_i corresponding to the signature key d. Given a message m each of the players P_i participating in the signature generation computes a "partial signature" which depends on the message m, the secret local key d_i, and, possibly, other public information. Partial signatures are then combined into the required full signature on m. This combining operation does not require secrecy and thus can be executed in a public form or by any (not necessarily trusted) single party. In particular, partial signatures do not require secrecy protection. On the other hand, a single incorrect partial signature can lead to a wrong final signature on m. Therefore, in order to add the robustness property to such a threshold signature scheme it suffices to solve the problem of verifying the correctness of a single partial signature.

It turns out that in all known threshold RSA signature schemes (e.g., [DF2], [DDFY], and [Ra2]), checking a partial signature reduces to checking an RSA signature produced using the secret d_i held by the signing player P_i. In these partial signatures d_i is in fact a secret RSA exponent, and the partial signature has the form $m^{d_i} \bmod n$ where n is the public RSA modulus. What makes this verification problem nontrivial is that the corresponding verification exponent e_i is unknown. In fact, this exponent must be hidden even from P_i since the knowledge of d_i and e_i together allows him to factor n and in turn to forge signatures. Hence, the problem our paper focuses on is: how to verify an RSA signature for which the modulus is known but the verification exponent is not.

Technically, the main contribution of our paper is in presenting and analyzing two efficient solutions for the problem of verifying a partial signature. Once this problem is solved the solution can be incorporated into any of the exiting threshold schemes [DF2], [DDFY], [Ra2] in order to achieve a robust scheme. In this paper we exemplify this by applying our current solution to the threshold RSA result of [Ra2].

The first solution is achieved by employing extensions which we developed of the *Information Checking Protocol* of Rabin and Ben-Or [Ra1], [RB]. The second solution draws from the *undeniable signature* work initiated by Chaum and van Antwerpen [CA]. This later protocol is of independent interest and our solution has been already used in other applications such as in [GKR] to achieve RSA-based undeniable signatures, and [FGMY], and [Ra2] in the context of proactive RSA. Our solutions require RSA composites of a special form, namely, the product of two safe primes (see Section 3).

We then combine this solution with an efficient threshold RSA scheme (e.g., [Ra2]) in order to achieve an efficient solution for robust (T, N)-threshold RSA signatures, for any threshold value T, $N \geq 2T + 1$.

In a recent and independent work, Frankel, et al. [FGY] have extended the notion of *result-checking* introduced by Blum and Kannan [BK], to the setting of *witness-based cryptographic checking*. Among the main motivations for that work is the generation of a robust threshold RSA signature scheme. The result of [FGY] works for any form of the RSA composite. While [FGY] provides a more general theoretical framework, our techniques, specifically designed for safe prime RSA, result in much more efficient and practical solutions. In particular, our basic protocols involve just a small constant number of modular exponentiations while in [FGY] a very large number of such costly exponentiations is required (at least $O(k)$ exponentiations for a probability of error of 2^{-k}).

Our scheme eliminates single points of failure from the system once the key is shared. We assume a dealer which generates the RSA keys and shares the secret key among the players, erasing afterward all the secret information it holds. The dealer is still a single point of failure in the system, but its presence is now limited to a very short period of time (the initialization of the system). In a recent result Boneh and Franklin [BF] showed a way to generate the RSA keys in a distributed fashion. Yet their solution cannot be used in our setting as it does not provide a method for verifying the properties of the composite modulus that we require. We note that, in principle, one could use generic results for secure multiparty computation [GMW1], [BGW], [CCD] to generate, distribute, and verify properties of an RSA key, but those are obviously too impractical for actual use.

2. Overview of Our Results

We briefly describe the basic construction underlying the existing threshold RSA schemes. Given $n = pq$, where p and q are large primes and $\varphi(n) = (p-1)(q-1)$, the public RSA key is a pair (n, e), where $\gcd(e, \varphi(n)) = 1$, the secret key is a number d, s.t. $ed \equiv 1 \bmod \varphi(n)$, and the signature on a message m is $S_m = m^d \bmod n$. We note that such a scheme applied directly to the signed message m is not secure against forgery; hereafter, we consider m to be an encoding—e.g., using a hash function—of the original message to be signed such that the resultant signature can be assumed to be secure against forging.

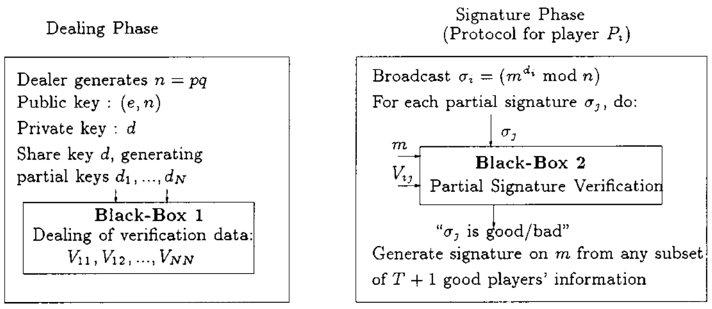
A distributed (T, N)-threshold signature scheme has two phases. In the first one, called the Dealing Phase, the secret key d is shared among the N players, such that each player P_i has a "share" d_i of d. These shares are created in such a manner that in the second phase, which is called the Signature Phase, when a message m is given, any subset S of $T + 1$ players can generate "partial signatures" $\sigma_i = m^{d_i} \bmod n$, $i \in S$, which are then combined to generate the signature of m. For illustration and concreteness, we provide in Section 3.3 details of one such threshold RSA scheme from [Ra2].

All of the existing RSA threshold schemes *require*, in order for the generated signature to be correct, that all partial signatures σ_i be correct as well. Consequently, these schemes cannot tolerate faults. If we are to confront failures, we must be able to detect which of the partial signatures provided by the players are improper. Once the incorrect partial signatures are sieved out, the computation can be carried out the same way as in the case where there are no faults.

In this paper we concentrate on providing solutions for the problem of detecting an improper partial signature. In the known threshold RSA solutions (in particular, [DF2], [DDFY], and [Ra2]), the shares $d_1, \ldots, d_N$ of the secret key d can be viewed themselves as RSA secret keys. Yet, for each of these keys the corresponding public exponent, $e_i = d_i^{-1} \bmod \varphi(n)$, is unknown. Furthermore, the public exponent e_i *must not* be known even to the signer P_i as it would expose a multiple of $\varphi(n)$ and hence would allow P_i to factor n [DeL] (and thus produce full signatures by itself without collaboration of a threshold of players). Hence, we are faced with the problem of checking a partial signature for which we do not have a public exponent. Under the circumstances where the public exponent is not known, there must be some other information that allows the verification of such partial signatures.

To achieve this goal, additional *verification data*, denoted $V_{11}, V_{12}, \ldots, V_{NN}$, is generated in the Dealing Phase, where V_{ij} is a piece of information used by player P_i during the Signature Phase to check the partial signature σ_j provided by another player P_j.

The following is a schema that represents in a generic way the phases and components of a robust threshold RSA signature scheme:


Dealing Phase
Dealer generates $n = pq$
Public key : (e, n)
Private key : d
Share key d, generating partial keys $d_1, \ldots, d_N$
Black-Box 1
Dealing of verification data: $V_{11}, V_{12}, \ldots, V_{NN}$
Signature Phase (Protocol for player P_i)
Broadcast $\sigma_i = (m^{d_i} \bmod n)$
For each partial signature σ_j, do:
σ_j
m
V_{ij}
Black-Box 2
Partial Signature Verification
"σ_j is good/bad"
Generate signature on m from any subset of $T + 1$ good players' information

The protocols presented in this paper correspond to the "black-boxes" shown in the schema. Using these protocols we can prove that there exist robust and unforgeable threshold RSA signature schemes (see Theorem 3).

We observe that the addition of the verification protocols to an otherwise nonrobust threshold signature scheme has an obvious performance cost. However, under normal circumstances when the system is not under active attack one can avoid this extra overhead: each player generates a signature from the appropriate number of broadcasted partial signatures, and then checks the obtained combined signature using the known public exponent e. If the result is a proper signature, then there is no need for verification of the partial signatures. Only if the combined signature fails should the verification protocols that we provide be triggered.

The tradeoffs between different properties of a fault-tolerant threshold RSA depend on the particular application of these techniques. For example, some applications may require minimal communication between the players during the Signature Phase. Others might need that partial signatures will be "publicly verifiable," namely, that any person, even outside the group of N players, can verify every partial signature. We have devised two different protocols for verification of partial signatures to address these different requirements. The choice of which protocol to use depends on the needs of a specific application.

The first protocol has a *noninteractive* Signature Phase, and requires each player to hold local secret verification data. The second protocol requires *interaction* in order to verify partial signatures, yet all the verification data is public (in particular, even parties not present at the dealing phase can later act as verifiers of partial signatures). In the Dealing Phase of the later protocol a publicly known *sample* message and its corresponding partial signatures are generated. In the Signature Phase of this protocol anyone can verify a player's partial signature by interacting with him, and using the sample signature as the basis for the proof. Both solutions are efficient computationwise (a verification involves only a few RSA exponentiations). Surprisingly, our noninteractive verification is somewhat more efficient in computation than the interactive one. Communicationwise the noninteractive solution is clearly optimal, while the interactive solution requires the exchange of four messages between every two players. Both protocols enjoy the fundamental property that they do not leak any information that can be used by the players (even malicious ones) to forge signatures.

3. Preliminaries

3.1. *Computation and Adversary Model*

We assume that our computation model is composed of a set of N *players* $\{P_1, \ldots, P_N\}$. They are connected by a complete network of private (i.e., untappable) point-to-point channels. In addition, they have access to a dedicated broadcast channel; by dedicated we mean that if P_i broadcasts a message, it will be recognized by the other players as coming from P_i.

These assumptions (privacy of the communication channels and dedication of the broadcast channel) allow us to focus on a high-level description of the protocols. However, it is worth noticing that these abstractions can be substituted with cryptographic techniques for privacy and authentication.

In addition to the N signing players, there exists a special entity, the dealer D, who generates (or is given) the secret signature key and is responsible for generating and distributing the key shares and verification data $V_{11}, V_{12}, \ldots, V_{NN}$ among the players. This distribution occurs before the system starts producing signatures and we call this the Dealing Phase. Only during this phase is the dealer active. It has no role during signature generation. (In practical implementations one could destroy all data held by the dealer at the end of the Dealing Phase.)

Our solutions deal with a special form of RSA, which is the following: the composite n will be generated as a product of two primes p, q of the form $p = 2p' + 1$ and $q = 2q' + 1$ where both p', q' are primes. The size of p', q' is related to a global security parameter.

The Adversary. We assume a computationally bounded adversary, $\mathcal{A}$, who cannot break the underlying basic signature scheme. This adversary can corrupt up to T of the $N \geq 2T + 1$ players in the network. We consider a *malicious* adversary who learns all the information held by the corrupted players and hears the broadcasted messages. He may cause corrupted players to behave in any (possibly malicious) way. Yet, the adversary can never corrupt the dealer D[1] or read its memory. Nor can he tap the communication line between two uncorrupted players.

[1] In Section 7 we relax this assumption by showing how the dealing actions of the dealer can be verified by the other players.

3.2. *Definitions of Threshold Signature Schemes*

The following definitions of secure threshold signature schemes are from [GJKR2]. Related definitions can be found in [CMI]. We start by recalling the definition of signature schemes in the traditional nondistributed setting.

Signature Scheme. A signature scheme $\mathcal{S}$ is a triple of efficient randomized algorithms (Key-Gen, Sig, Ver). Key-Gen is the *key generator* algorithm: on input a random string, it outputs a pair (VK, SK), such that VK is the *public key* and SK is the *secret key* of the signature scheme. Sig is the *signing* algorithm: on input a message m and the secret key SK, it outputs *sig*, a signature of the message m. Ver is the *verification* algorithm. On input a message m, the public key VK, and a string *sig*, it checks whether *sig* is a proper signature of m.

The notion of security for signature schemes was formally defined by Goldwasser, et al. [GMR1] using various security flavors. The following definition captures the strongest of these notions: existential unforgeability against adaptively chosen message attack.

Definition 1. We say that a signature scheme $\mathcal{S} =$ (Key-Gen,Sig,Ver) is *unforgeable* if no adversary who is given the public key VK generated by Key-Gen, and the signatures of k adaptively chosen messages $m_1, \ldots, m_k$, can produce the signature on a new message m with nonnegligible probability.

Threshold Signature Schemes. Let $\mathcal{S} =$ (Key-Gen, Sig, Ver) be a signature scheme. A (T, N)-threshold signature scheme $\mathcal{TS}$ for $\mathcal{S}$ is a pair of protocols (Thresh-Key-Gen, Thresh-Sig) for the set of players $\{P_1, \ldots, P_N\}$ (and for a designated dealer in the case of Thresh-Key-Gen).

Thresh-Key-Gen is a key generation protocol carried out by a designated dealer to generate a pair (VK, SK) of public/private keys with the same distribution as Key-Gen. At the end of the protocol the private output of player P_i is a value SK_i (related to the private key SK). The public output of the protocol contains the public key VK.

Thresh-Sig is the distributed signature protocol. The private input of P_i is the value SK_i. The public inputs for all players consist of a message m and the public key VK. The output of the protocol is the value $sig = \mathsf{Sig}(m, SK)$. (The verification algorithm is, therefore, the same as in the regular signature scheme $\mathcal{S}$.)

Secure Threshold Signature Schemes. Our definition of security includes both *unforgeability* and *robustness* as defined below.

Definition 2. We say that a (T, N)-threshold signature scheme $\mathcal{TS} =$ (Thresh-Key-Gen, Thresh-Sig) is *unforgeable* if no malicious adversary who corrupts at most T players can produce the signature on any new (i.e., previously unsigned) message m, given the view of the protocol Thresh-Key-Gen and of the protocol Thresh-Sig on input messages $m_1, \ldots, m_k$ which the adversary adaptively chose.

This is the analogous to the notion of existential unforgeability under chosen message attack presented in Definition 1. Notice that now the adversary does not just see the signatures of k messages adaptively chosen, but also the internal state of the corrupted players and the public output of the protocols. Following [GMR1] one can also define weaker notions of unforgeability.

In order to prove unforgeability we use the concept of *simulatable adversary view* [GMR2], [MR]. Intuitively, this means that the adversary who sees all the information of the corrupted players and the signature of m, could generate by itself all the other public information produced by the protocol Thresh-Sig. In other words, the run of the protocol provides no useful information to the adversary other than the final signature on m.

Definition 3. A threshold signature scheme $\mathcal{TS} = (\text{Thresh-Key-Gen}, \text{Thresh-Sig})$ is *simulatable* if the following properties hold for any efficient (polynomial-time) adversary against $\mathcal{TS}$:

1. The protocol Thresh-Key-Gen is simulatable. That is, there exists an efficient (polynomial-time) simulator SIM_1 that, on input the public key VK generated by an execution of Thresh-Key-Gen, can simulate the view of the adversary on that execution.
2. The protocol Thresh-Sig is simulatable. That is, there exists an efficient simulator SIM_2 that, on input the public input of Thresh-Sig (in particular VK and m), the transcript of the execution of SIM_1 (in particular the private information held by the adversary), and the signature sig of m, SIM_2 can simulate the view of the adversary on an execution of Thresh-Sig that generates s as an output.

This is actually a stronger property than what we need. Indeed it would be enough for us to say that the executions of the protocols Thresh-Key-Gen and Thresh-Sig give the adversary no advantage in forging signatures for the scheme $\mathcal{S}$. In other words, we could allow the adversary to gain knowledge provided that such knowledge is useless for forging. However, our stronger definition subsumes this specific goal and provides a proof of security for any of the "flavors" of signature security as listed in [GMR1] (see Proposition 1 below). Another important advantage of proving simulatability of the protocol is that this property allows for securely composing the threshold protocol into other application protocols.[2]

Proposition 1. *If $\mathcal{S}$ is an unforgeable signature scheme* (*Definition* 1) *and $\mathcal{TS}$ is a simulatable threshold signature protocol* (*Definition* 3) *for $\mathcal{S}$, then $\mathcal{TS}$ is unforgeable* (*Definition* 2).

Proof. We prove this for the case of an existential unforgeability against adaptive chosen message attack. The proof is similar for weaker notions.

[2] The simulators used in this paper all admit *auxiliary input* and thus can be *sequentially* composed [Go].

Assume that $\mathcal{TS}$ is forgeable, i.e., upon executing k threshold signature generations on messages $m_1, \ldots, m_k$ and outputting the appropriate signatures, an adversary who corrupted at most T players can produce a signature on a new (previously unsigned) message m. Denote this adversary by $\mathcal{A}$. Let us design a forger $\mathcal{F}$ for the signature scheme $\mathcal{S}$. Forger $\mathcal{F}$ has access to an oracle that provides signatures under $\mathcal{S}$ using a given public key *VK*. In the construction of $\mathcal{F}$ we make use of the adversary $\mathcal{A}$ and the simulators SIM_1 and SIM_2 of the adversary $\mathcal{A}$ that are guaranteed to exist by Definition 3.

First, $\mathcal{F}$ runs SIM_1 with public key *VK* to obtain the view of the adversary $\mathcal{A}$ on an execution of the key generation process that results on output of the public key *VK*. In particular, this adversary's view contains the private information of the corrupted players.

Then $\mathcal{F}$ runs SIM_2 with adversary $\mathcal{A}$ as follows. For each message m_i queried by $\mathcal{A}$, $\mathcal{F}$ queries the $\mathcal{S}$ signature oracle to obtain the signature on m_i, and uses this signature as input to SIM_2.

In this way the whole view of $\mathcal{A}$ from key generation to the distributed computation of signatures on the messages $m_1, \ldots, m_k$ chosen by $\mathcal{A}$ is generated by $\mathcal{F}$ with an indistinguishable distribution from the real execution. If $\mathcal{A}$ succeeds in forging a signature on a new message m, then so does $\mathcal{F}$. However, the latter contradicts the security of the signature scheme $\mathcal{S}$. $\square$

As we remarked earlier, Definitions 1 and 2 can be relaxed to allow weaker notions of unforgeability [GMR1]. Proposition 1 holds also for these weaker notions. That is, if $\mathcal{TS}$ is simulatable, then $\mathcal{TS}$ inherits the same level of unforgeability that $\mathcal{S}$ has.

We now define the other important property of threshold schemes, namely, robustness. This property ensures that the protocol will compute a correct output even in the presence of malicious faults.

Definition 4. A threshold signature scheme $\mathcal{TS} =$ (Thresh-Key-Gen, Thresh-Sig) is T-*robust* if even in the presence of an adversary who corrupts T players, both Thresh-Key-Gen and Thresh-Sig are guaranteed to complete successfully.

Remark (optimal resiliency). A consequence of our definition of security, that requires both unforgeability and robustness (as long as T or less parties are corrupted), is that the total number N of players in the system must be at least $2T + 1$. To see this, assume by means of contradiction that there is a secure (T, N)-threshold signature scheme where up to T players may be maliciously faulty and $N = 2T$. Consider the following adversary strategy: crash the faulty T players during the signature computation protocol. Since the protocol is robust this means that it will successfully complete anyway. However, this means that the remaining T players can compute the signature on their own which is in direct contradiction with the fact that the scheme is unforgeable for coalitions of T or less players. Notice that this argument applies to our setting with a trusted dealer; indeed, there is nothing a trusted dealer can do to avoid the above counterexample. An important property of our schemes is that they achieve $N = 2T + 1$ and thus they are optimally resilient.

Input: Secret key d, composite n,
number of players N, and threshold T.

Secret Key Distribution—executed by the dealer.

1. Choose and hand P_i value $d_i \in_R [0..\varphi(n)-1]$ for $1 \le i \le N-1$, set $d_N = d - \sum_{i=1}^{N-1} d_i \bmod \varphi(n)$.
2. Share the value d_i using any T-out-of-N Verifiable Secret Sharing (VSS) protocol.

Signature Generation
Input: Message m.

1. Player P_i publishes partial signature $\sigma_i = m^{d_i} \bmod n$.
2. If player P_i fails to give a partial signature all players reconstruct d_i using the Reconstruction Phase of the VSS protocol.
3. Set $S_m = \prod_{i=1}^{N} \sigma_i \bmod n$.

Fig. 1. Secret key distribution and signature generation [Ra2].

3.3. *Threshold Sharing of RSA functions*

As said before, our contribution is in adding the robustness property to existing RSA threshold signature schemes. Our techniques apply to several such schemes. For the sake of illustration and concreteness we briefly describe here a particular solution to which our techniques add robustness. We choose to present the solution of [Ra2] due to its extreme simplicity. This solution is presented in Fig. 1 and is explained next. We stress however that our techniques are applicable to several other nonrobust threshold RSA schemes like the ones in [DF2] and [DDFY].

Given the secret key d and p, q the dealer chooses random values $d_1, \ldots, d_{N-1} \in_R [0..\varphi(n)-1]$, he then sets $d_N = d - \sum_{i=1}^{N-1} d_i \bmod \varphi(n)$. For each value d_i he further shares this value using any T-out-of-N Verifiable Secret Sharing (VSS) protocol (e.g, [BGW] and [Pe] where the sharing is computed in a prime field Z_P with $P > n$). We stress that this is a one-time operation.

On input a message m the protocol is carried out by having each player compute the partial signature $\sigma_i = m^{d_i} \bmod n$ and publish it. The signature is $S_m = \prod_{i=1}^{N} m^{d_i} \bmod n$. This works if all players are honest, however, a malicious player may contribute an incorrect partial signature. Our techniques allow for testing each partial signature for correctness. If a player fails to compute the correct partial signature, then his share d_i is reconstructed using the VSS Reconstruction Phase, and the partial signature is recomputed.

Remark. In order to simplify the description of our robustness technique, what we described above is a simplified version of the protocol in [Ra2]. In particular, [Ra2] presents ways to avoid revealing in the clear the shares of players who do not contribute a correct partial signature (this can be desirable in case the player is not really corrupted, but simply disconnected or lagging). Our techniques can be directly applied to add robustness to the complete solution of [Ra2] as well.

3.4. *A Technical Issue*

The methods which we describe in this paper for verifying a single RSA signature pose a technical difficulty. A partial signature σ_i will be accepted as correct if either $\sigma_i = m^{d_i} \bmod n$ or $\sigma_i = -m^{d_i} \bmod n$. As it turns out this is not a serious problem. A wrong sign can only influence the sign of the final signature, but then the correct sign can be decided using the public verification exponent e on the produced signature.

3.5. *Notation*

For a positive integer k we denote $[k] \overset{\text{def}}{=} \{1, \ldots, k\}$. The public modulus is denoted by n. We assume $n = pq$, and $p = 2p' + 1$, $q = 2q' + 1$, where $p < q$ and p, q, p', q' are all prime numbers. Z_n^* denotes the multiplicative group of integers modulo n, and $\varphi(n) = (p-1)(q-1)$ is the order of this group. For an element $w \in Z_n^*$ we denote by $ord(w)$ the order of w in Z_n^* and by $ind(w)$ the index of w in this group (it holds that $ind(w) = \varphi(n)/ord(w)$). The subgroup generated by an element $w \in Z_n^*$ is denoted by $\langle w \rangle$. The number $d \in [\varphi(n)]$ denotes the (private) signature exponent. For any message $m \in Z_n^*$ we denote by S_m the corresponding signature on m, namely, $S_m = m^d \bmod n$.

4. Noninteractive Robust Threshold RSA

Here we present our noninteractive solution to the robustness problem of threshold RSA signatures. Section 4.2 contains the protocol for dealing verification information during the Dealing Phase, while in Section 4.3 we describe the protocol for verification of partial signatures during the Signature Phase. (See the schema in Section 2 for the role of these components in the full threshold signature protocol.) Our solution is based on the Information Checking Protocol (ICP) from [Ra1] and [RB]. The original ICP technique is intended for *one-time* verification of information provided by an untrusted party. In our case we extend this technique to verification of *multiple* partial signatures; in particular, we extend ICP to work over the integers rather than over a prime field as originally designed.

We first give a very rough sketch of the noninteractive solution. Consider two players P and V, P generates a signature which V wants to verify. The verification will be achieved by having both P and V hold auxiliary values. The prover P holds values d (the secret key) and y. The verifier V holds b and c, such that $y = bd + c$. The values d, y, b, and c are dealt to the corresponding parties during the Dealing Phase (and kept secret by the parties). Given a message m, the prover generates the partial signature $m^d \bmod n$, and the additional information $m^y \bmod n$. P gives these values to V, who verifies the partial signature by checking whether $(m^d)^b m^c = m^y \bmod n$.

An important technical aspect of this solution is that the equality $y = bd + c$ needs to hold over the integers. The more natural approach of generating this equation modulo $\varphi(n)$, would enable P and V to combine their information and compute a multiple of $\varphi(n)$ which, in turn, would allow for the efficient factorization of n [DeL]. In the next subsection we present an extension over the integers of the original ICP.

ICP-Gen-Integers

Input: RSA composite n, secret value $d \in [\varphi(n)]$ known to D, security parameters $0 \leq k_1, k_2, k_3 = k_1 + k_2 + \log n$.

1. Choose $b \in [2^{k_1}]$ and $c \in [2^{k_3}]$ with uniform distribution.
2. Compute $y = c + bd$ over the integers.
3. Secretly transmit d and y to the prover P.
4. Secretly transmit b and c to the verifier V.

Fig. 2. The ICP generation protocol.

4.1. *Extensions of Information Checking*

The following protocol is carried out by three players: a dealer D, who is nonfaulty, and two additional players: prover P and verifier V, who can be either faulty or not. In Fig. 2 we present the ICP generation protocol over the integers, carried out by the dealer. The protocol gets as input the RSA composite, the secret d, and two security parameters k_1 and k_2. The computation carried out in the protocol is a function of the security parameters.

For the following we denote $\mathcal{Y} \stackrel{\text{def}}{=} [2^{k_1+\log n}, \ldots, 2^{k_3}]$.

Lemma 1. *Given values $d \in [\varphi(n)]$ and $y \in \mathcal{Y}$, for every possible value of b there is exactly one possible value for $c \in [2^{k_3}]$ such that $y = bd + c$.*

Proof. Since the computation is over the integers, there is exactly one value of c for each b, d, and y. Furthermore, if $b \in [2^{k_1}]$ and $y \in [2^{k_1+\log n}, \ldots, 2^{k_3}]$, then value $c = y - bd$ is contained in $[2^{k_3}]$ because

$$1 \leq 2^{k_1+\log n} - 2^{k_1}\varphi(n) = y_{min} - b_{max}d_{max} \leq c \leq y_{max} - b_{min}d_{min} = 2^{k_3} - 1,$$

which proves the lemma. □

Lemma 2. *$Pr(y \notin \mathcal{Y}) \leq 1/2^{k_2}$.*

Proof. The number of options to choose different pairs of b and c is $2^{k_1}2^{k_3}$. The range $\mathcal{Y}$ is of size $2^{k_3} - 2^{k_1+\log n}$. From Lemma 1 it follows that each value y in this range can be generated by 2^{k_1} pairs (b, c). Consequently, as b, c are chosen with uniform distribution, the probability of y falling outside of this range is $1 - (2^{k_3} - 2^{k_1+\log n})2^{k_1}/2^{k_3}2^{k_1} = 1/2^{k_2}$. □

In a generic (one-time) application of ICP the variables y, d and b, c are used by players P and V in an *authentication protocol* in the following way: When the prover P wants to prove to the verifier V that he holds the value d which he received from D, he sends d and y to V. Upon receiving values $\hat{d}, \hat{y}$ from P, the verifier *accepts* $\hat{d}$ (i.e., concludes that $\hat{d} = d$) only if $\hat{y} = b\hat{d} + c$.

In the following lemma we prove the properties of the ICP over the integers with respect to the generation protocol (Fig. 2) and the above described authentication protocol.

Lemma 3 (ICP over the Integers).

Completeness. *If P and V follow the protocol, then V always accepts d in the authentication protocol.*

Soundness. *The probability that P generates $\hat{d}, \hat{y}$ such that $c + b\hat{d} = \hat{y}$ when in fact $\hat{d} \neq d$ is at most $1/2^{k_1} + 1/2^{k_2}$. We denote this occurrence as OC_1.*

Zero-knowledge. *Given b, c the verifier learns no additional information on the value d, i.e., for any value d_0, $Prob[d = d_0 | b, c] = Prob[d = d_0]$.*

Proof. *Completeness.* Immediate.

Soundness. Notice that $\Pr(OC_1) \leq \Pr(OC_1 \mid y \in \mathcal{Y}) + \Pr(y \notin \mathcal{Y})$. From Lemma 1 it follows that if $y \in \mathcal{Y}$, then P will be able to generate values $\hat{d}, \hat{y}$ (where $\hat{d} \neq d$) which satisfy the equation $\hat{y} = b\hat{d} + c$ with probability at most $1/2^{k_1}$. Lemma 2 gives us the probability that y falls out of this range. Combining these probabilities we prove our lemma.

Zero-knowledge. Values b and c are uniformly distributed and randomly chosen independently from d, and hence reveal no information on its value. □

4.2. *Generation of Verification Data* (*Dealing Phase*)

In order to generate the data for verification of partial signatures within the context of the Dealing Phase, the dealer simply runs ICP-Gen-Integers (Fig. 2) for every pair of players P_i and P_j. All these invocations have as input the same RSA composite n and security parameters k_1, k_2. For the invocation where P_i is the prover P, and P_j is the verifier V, the secret key input is d_i, namely, P_i's secret partial key.

It will be seen later that a single pair of values b, c suffices for V to verify multiple different partial signatures of P. This results in an efficient protocol for the dealer, as the number of invocations to the ICP-Gen-Integers protocol during the Dealing Phase depends only on the number of players but not on the number of signatures that the system will need to generate.

As a result of the complete Dealing Phase, player P_i holds the following values:

1. His share d_i.
2. Auxiliary authentication values $y_{i,1}, \ldots, y_{i,N}$, where $y_{i,j} \in \mathbb{Z}$ is used to prove his partial signature to P_j.
3. Verification data $V_{1,i}, \ldots, V_{N,i}$, where $V_{j,i} = (b_{j,i}, c_{j,i})$, $b_{j,i} \in [2^{k_1}]$, and $c_{j,i} \in [2^{k_3}]$. For each j, the pair $V_{j,i}$ is used to verify the correctness of P_j's partial signatures.

4.3. *Partial Signatures Verification* (*Signature Phase*)

We show the protocol for verification of a partial signature where there are two players, P and V, each holding the data which they received in ICP-Gen-Integers. The protocol appears in Fig. 3. In the context of the Signature Phase this protocol will be carried

Noninteractive Verification

Input: Player V: $b \in [2^{k_1}]$, $c \in [2^{k_3}]$.
Player P: $d \in [\varphi(n)]$, $y = bd + c$.
Both players: message $m \in Z_n^*$, RSA composite n.

1. P broadcasts the signature $\hat{S}_m = m^d \bmod n$ and the auxiliary value $Y = m^y \bmod n$.
2. V checks if $\hat{S}_m^b m^c = Y$. If yes, he concludes that $\hat{S}_m = \pm m^d \bmod n$ and accepts it.

Fig. 3. The Noninteractive Verification protocol.

out by every pair of players. After executing all these invocations of the Noninteractive Verification protocol, player P_i will take a subset of $T + 1$ partial signatures which he has accepted, and will generate the signature on m.

Theorem 1 (Noninteractive Verification). *Assume that a cheating prover P^* cannot break RSA with modulus n (in particular, he does not know and cannot compute the factorization of n). Let $n = pq$, where $p < q$, $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are all prime numbers.*

Completeness. *If P and V follow the protocol, then V always accepts the partial signature.*

Soundness. *A cheating prover P^* can convince V to accept $\hat{S}_m \neq \pm m^d \bmod n$, with probability at most $1/p' + 1/2^{k_1} + 1/2^{k_2}$.*

Zero-knowledge. *V does not learn any information beyond the signature $S_m = m^d \bmod n$, i.e., given S_m, b, c there is a polynomial time procedure to compute Y.*

Proof. *Completeness.* Immediate.

Soundness. First we examine the case where $y \in \mathcal{Y} = [2^{k_1 + \log n}, \ldots, 2^{k_3}]$. Notice that the verifier uses a deterministic procedure to accept or reject the published pair S, Y. Therefore the probability stated in the theorem is taken over the coin tosses of the dealer in ICP-Gen-Integers which are consistent with the view of the prover (i.e., the value y). In order for P to convince V to accept $\hat{S}_m, Y$, it must hold that $Y = \hat{S}_m^b m^c \bmod n$. We know from the ICP-Gen-Integers protocol that $y = bd + c$, and hence $m^y = (m^d)^b m^c \bmod n$. By dividing these two equations we get that

$$Y m^{-y} = (\hat{S}_m m^{-d})^b \bmod n. \tag{1}$$

This means that Ym^{-y} must be in the subgroup $\langle \hat{S}_m m^{-d} \rangle$. Let k be the minimal value such that $Ym^{-y} = (\hat{S}_m m^{-d})^k \bmod n$. Consequently, (1) is satisfied only if $b = k \bmod ord(\hat{S}_m m^{-d})$. Since b is chosen at random with uniform distribution from $[2^{k_1}]$, the probability that a pair $(\hat{S}_m, Y)$ satisfies 1 is

$$\Pr(\, b = k \bmod ord(\hat{S}_m m^{-d})\,) \le \frac{\lceil 2^{k_1}/ord(\hat{S}_m m^{-d})\rceil}{2^{k_1}} \le \frac{1}{ord(\hat{S}_m m^{-d})} + \frac{1}{2^{k_1}}.$$

Because of the special form of n, there are only four elements of Z_n^* whose order is smaller than p', namely, the four roots of unity. If $\hat{S}_m m^{-d} = \pm 1 \bmod n$ then $\hat{S}_m = \pm m^d \bmod n$. If the prover could find $\hat{S}_m$ such that $\hat{S}_m m^{-d}$ is a nontrivial root of unity, then he could factor n which we assume to be infeasible. For all other choices of $\hat{S}_m$, order $ord(\hat{S}_m m^{-d}) \geq p'$. This completes the proof for the case where $y \in \mathcal{Y}$. However, from Lemma 2 we know that the probability that $y \notin \mathcal{Y}$ is at most $1/2^{k_2}$, by combining these two probabilities we get the desired probability.

Zero-knowledge. Immediate since, knowing b, c, and $\hat{S}_m = m^d \bmod n$, the verifier can compute $Y = m^y = \hat{S}_m^b m^c \bmod n$. (We comment that since this is a noninteractive procedure, the only role of V is in the verification of the information sent by the prover; thus we do not need to consider—as is typical in the interactive case—arbitrary behaviors of V.) □

The prover can compute the signature one time (for all players), and generate the additional needed values for each player individually. To give a numerical example of the amount of computation that the two parties carry out, assume that the desired probability of error for the protocol is set to 2^{-80}. In this case both k_1 and k_2 can be set to 81. Then the prover does one full exponentiation for the computation of the signature, also for each player he computes another exponentiation and $k_1 + k_2$ (i.e., 162) multiplications. The verifier on his part computes a single full exponentiation and $2k_1 + k_2$ (i.e., 243) multiplications.

Note that a single pair (b, c) is sufficient for V to verify the multiple signature of P, as for P to compute these values is equivalent to breaking RSA.

Remark. The values b, c are used by V as secret exponents in modular exponentiations. Notice that we choose those exponents relatively short (i.e., k_1 and k_2 bits, respectively). It is known that there are some attacks on RSA using short secret exponents (e.g., [Wi]), but they do not apply in this case for the following reasons:

- those attacks use the knowledge of a public exponent which is not present here,
- more importantly the results of the exponentiations $\hat{S}_m^b$ and m^c are *never* revealed by V, as they are used for internal checks only.

4.4. *Performance Analysis*

Computation. Under normal operation of the system, i.e., when no faults occur, each player needs to compute a partial signature which is a single exponentiation of an exponent which is the length of the RSA modulus.

When faults occur the players need to engage in Noninteractive Verification. Each such verification requires P to carry out an additional exponentiation of an exponent of length k_3. The verifier needs to compute two exponentiations with exponents of length at most k_3.

When a player fails the noninteractive verification test, which means that it is faulty, the private share of this player is reconstructed. This is done using the VSS Reconstruction Phase which is a computation which requires only a polynomial number in N of multiplications.

Memory. The size of secret memory required for each player is $(N+1)\log n + 3Nk_3$.

Communication. Signature generation requires one round of communication. If Noninteractive Verification is invoked this requires an additional round of communication, and the VSS Reconstruction Phase requires another additional round.

5. Interactive Robust Threshold RSA

Like in the noninteractive protocol the two components of the interactive protocol are the protocol for dealing verification information during the Dealing Phase (5.1), and the protocol for verification of partial signatures during the Signature Phase (5.2). (See the schema in Section 2.)

The basic idea underlying the interactive solution is that RSA signatures can be verified even if the verification exponent is unknown to the verifier, provided that the later is given the correct RSA signature on a (suitably chosen and) known message w. That is, there will be a fixed and public *sample* message w for which the partial signatures $w^{d_i} \bmod n$, $i = 1, 2, \ldots, N$, corresponding to the N players are all known. The choice of w and generation of sample signatures are all done during the dealing phase.

5.1. *Generation of Verification Data* (*Dealing Phase*)

The verification information dealt during the initialization protocol (and used during the Signature Phase to verify partial signatures) consists of a random public *sample* message w and and its corresponding sample partial signatures $w^{d_i} \bmod n$, for each one of the partial keys d_i held by the players. The sample signatures are broadcast to all players, and can even be made public (Fig. 4).

In terms of our generic schema in Section 2, the verification data is $V_{i,j} = w_j = w^{d_j} \bmod n$, for all i, j. Notice that unlike in the noninteractive protocol, here the $V_{i,j}$'s are public. Furthermore, note that the commitments to the partial key also generate a commitment to the signing key d, i.e., the value $w^d \bmod n$ is also made public.

5.2. *Verification of Partial Signatures* (*Signature Phase*)

During the Signature Generation Phase, a shared signature is computed using the partial signatures. Here, we provide a protocol by which each partial signature can be verified

Sample Partial-Signature Generation

Input:	Public:	RSA modulus n.
	Dealer $\mathcal{D}$:	key-shares $d_i \in [\varphi(n)]$, for $i = 1, 2, \ldots, N$.

1. $\mathcal{D}$ chooses a random value w of high order in Z_n^* and broadcasts values $w_i = w^{d_i} \bmod n$ for $i = 1, 2, \ldots, N$.

Fig. 4. The sample signature generation protocol.

by the other participants by interacting with the signer. This interactive protocol uses challenges to the signer based on the message m whose signature is being verified and on the sample message w; the challenge is verified using the alleged partial signature of m and the known signature on w. A successfully answered challenge by the signer proves the correctness of the signature on m. Indeed, we show that if the signature on m is incorrect the signer has only negligible probability to convince the verifier to accept its validity. In addition, we prove this protocol to be *zero-knowledge* [GMR2], thus even corrupted verifiers do not learn any information from the execution of this protocol that may help in forging signatures.

Our solution is based on a protocol due to Chaum and van Antwerpen [CA], and further developed in [Ch] and [BCDP], designed to prove in zero-knowledge the equality of the discrete logarithms of two elements over a prime field Z_p relative to two different bases. The protocol and the proof presented in the above papers do not work over Z_n for composite n as required here, in particular, since they strongly rely on the existence of a generator for the multiplicative group Z_p^*. We show a variant of the protocol that we prove to solve our problem over Z_n.

In the Signature Phase, each player P_i checks the partial signatures produced by each other player P_j. For clarity of presentation, we concentrate on two players only, the prover (who is the signer) P and the verifier V. The player P has his secret signature key $d \in [\varphi(n)]$ and both players have access to a publicly known sample message w and its partial signature (under P's key) $w^d \bmod n$. For any $x \in Z_n^*$ we denote by S_x the corresponding signature of P on x under the key d, namely, $S_x = x^d \bmod n$. By $\hat{S}_x$ we denote the "alleged" signature on x, i.e., a string claimed (but not yet verified) to be the signature of x.

Figure 5 presents the basic Interactive Verification protocol. This description corresponds to an interactive proof between P and V. The function *commit* is a commitment function [Go], and adding this step to the protocol is done to achieve the zero-knowledge condition (these are standard techniques, see [GMW2], [BCC], and [Go]). Indeed, the

Interactive Verification

Input: Prover: secret $d \in [\varphi(n)]$.
Common: RSA composite n, sample message $w \in Z_n^*$, signature S_w, message $m \in Z_n^*$, claimed $\hat{S}_m$.

1. V chooses $i, j \in_R [n]$ and computes $Q \overset{\text{def}}{=} m^i w^j \bmod n$.
   $V \longrightarrow P$: Q.
2. P computes $A \overset{\text{def}}{=} Q^d \bmod n$ and $commit(A)$.
   $P \longrightarrow V$: $commit(A)$.
3. $V \longrightarrow P$: i, j.
4. P verifies that $Q = m^i w^j \bmod n$.
   If equality holds, then P decommits to A.
5. V verifies that $A = \hat{S}_m^i {S_w}^j \bmod n$.
   If equality holds, then V accepts $\hat{S}_m$ as the signature on m, otherwise it rejects.

Fig. 5. The Interactive Verification protocol.

zero-knowledge condition is achieved through the properties of the commitment function, namely, (I) $commit(x)$ reveals no information on x, and (II) P cannot find x' such that $commit(x) = commit(x')$.

Commitment functions can be implemented in many ways, in particular using a semantically secure encryption. In our context, one could use commitment functions based on the RSA function (e.g., [GM] and [BR]) where the private key is not known to V (and, possibly, not even known to P), and the public key is known to both parties. Commitment to A is done by P by (probabilistically) encrypting this value, while decommitment is done by having P reveal both A and the string r used for the probabilistic encryption. This implementation of a commitment function is very efficient as it involves no long exponentiations.

In the following theorem we state the security properties of the above Interactive Verification protocol.

Theorem 2. *Let $n = pq$, where $p < q$, $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are all prime numbers. Assuming that factoring n is hard (i.e., no cheating prover knows or can compute the factorization of n) then the Interactive Verification protocol in Fig.* 5 *is a zero-knowledge proof system for the language $EDL_n = \{(m, S_m, w, S_w) \in (Z_n^*)^4$ s.t. there exists d: $S_m = \pm m^d \bmod n$ and $S_w = w^d \bmod n\}$.*

We therefore need to prove the following three properties:

Completeness. If P and V follow the protocol, then V always accepts the signature.
Soundness. No cheating prover P^* can convince V to accept $\hat{S}_m \neq \pm m^d \bmod n$, except for a negligible probability $O(1)/p'$.
Zero-knowledge (*informal*). Any (possibly cheating) verifier V^* interacting with prover P does not learn any information beyond the signature $S_m = m^d \bmod n$.

The proof of completeness of the protocol is immediate and the zero-knowledge property is argued above. Here we prove the soundness property. The following is the core claim behind the proof of soundness.

Lemma 4. *The prover's cheating probability in the Interactive Verification protocol is at most $ind(w)/ord(\hat{S}_m/m^d) + 2((n - \varphi(n))/n)$.*

Proof. The prover's probability to cheat, i.e., to convince V to accept $\hat{S}_m$, is maximized by choosing A that passes V's test (in Step 3) with maximal probability (relative to the values i, j chosen by V). Since the prover chooses A after having seen the "challenge" Q from V (and based on its knowledge of $\hat{S}_m$, m, w, d, and n), then the proof of soundness needs to capture that some information on i, j (at least from the information-theoretic point of view) is available to the prover when selecting A.

In the actual protocol, V chooses i, j randomly from the set $[n]$; for simplicity of analysis we assume that these values are chosen from $[\varphi(n)]$, and account for the event that either i or j fall outside of this range in the prover's probability to cheat. The probability of such an event (i.e., that i or $j \notin [\varphi(n)]$) is at most $2((n - \varphi(n))/n)$, and it appears in the lemma's bound. Therefore we assume $i, j \in_R [\varphi(n)]$.

We define $I(Q) = \{i \in [\varphi(n)]\colon \exists j,\ Q = m^i w^j \bmod n\}$. We write $\hat{S}_m = \alpha m^d$, for $\alpha \in Z_n^*$. In Step 3 the verifier will check whether $A = \hat{S}_m^i S_w^j = \alpha^i m^{di} w^{dj} = \alpha^i Q^d$. As the value α has been set then for any A the number of i's which satisfy that $\alpha^i = AQ^{-d}$ is at most $\varphi(n)/ord(\alpha)$. Given Q, V's choice of i is uniformly distributed over $I(Q)$, as for each $i \in I(Q)$ there is the same number ($ind(w)$) of values j which satisfy the equation $Q = m^i w^j \bmod n$. Thus, the probability of P to succeed is at most $(\varphi(n)/ord(\alpha))/|I(Q)|$.

Now we show that, $\forall Q$, $|I(Q)| = 0$ or $|I(Q)| \geq ord(w)$. For values $i \in I(Q)$ and Δ such that $m^\Delta \in \langle w \rangle$, it holds that $i + \Delta \in I(Q)$ (because there exist j, j' such that $Q = m^i w^j$ and $m^\Delta = w^{j'}$ from which it follows that $Q = m^{i+\Delta} w^{j-j'}$). Therefore, we get that $I(Q) = \{i_0 + \Delta\colon m^\Delta \in \langle w \rangle \text{and } \Delta < \varphi(n)\}$ where i_0 is the minimal element in $I(Q)$. Thus, if $I(Q)$ is nonempty, then its size equals the size of the set $D = \{\Delta < \varphi(n)\colon m^\Delta \in \langle w \rangle\}$. We proceed to bound the size of D. Using standard arguments it is easy to show that if δ is the minimal nonzero element of D, then the elements of D are exactly the multiples of δ (smaller than $\varphi(n)$). Thus, $|D| = \varphi(n)/\delta$. We end the proof by showing $\delta \leq ind(w)$. Let $i_1 < i_2 \leq \delta$. The cosets $m^{i_1}\langle w \rangle$ and $m^{i_2}\langle w \rangle$ are disjoint (a common element would imply that $w^{i_2 - i_1} \in \langle w \rangle$ in contradiction to the minimality of δ). Thus, $m^1\langle w \rangle, m^2\langle w \rangle, \ldots, m^\delta\langle w \rangle$ are δ disjoint cosets in Z_n^* each of size $ord(w)$. We have $\delta \leq |Z_n^*|/ord(w) = \varphi(n)/ord(w) = ind(w)$. In sum, $|I(Q)| = |D| = \varphi(n)/\delta \geq \varphi(n)/ind(w) = ord(w)$. Combining all the above we get our probability $\varphi(n)/ord(\alpha)ord(w) = ind(w)/ord(\alpha)$. $\square$

We stress that the above lemma holds also for a computationally unbounded cheating prover, and that the bound in the lemma is tight for such a prover (up to the term $2((n - \varphi(n))/n)$). Next, we show how to apply the lemma to prove the soundness of the protocol in the case that n is chosen with the particular form stated in Theorem 2, and the (cheating) prover cannot break RSA.

Proof of Theorem 2 (Soundness). The bound in Lemma 4 is given in terms of the order of elements in the group Z_n^*. Thus, in order to establish the exact bound for the above special form of n we need to study the order of elements in this particular group. There is one element of order 1 in Z_n^* (the unit element), three of order 2 (-1 and two other nontrivial roots of 1), $4p' + 4q' - 8$ elements of order ranging between p' and $2q'$, and the rest have all order which is at least $p'q'$. (This can be argued based on the special form of p and q and the order of elements modulo these primes, and then using the Chinese Remainder Theorem.) In particular, the order of w, which is chosen at random, is at least $p'q'$, with probability $1 - 4(p' + q')/\varphi(n) \geq 1 - 2/p'$, and then $ind(w)$ which equals $\varphi(n)/ord(w)$ is at most 4 (notice that $\varphi(n) = 4p'q'$).

As in the noninteractive protocol (see soundness part of Theorem 1), a successful cheating of P^* happens when it convinces V to accept a value $\hat{S}_m = \alpha m^d \bmod n$, for $\alpha \neq \pm 1 \bmod n$. Notice that the prover (who knows d) can compute such an α from m and $\hat{S}_m$. This excludes the possibility that α would be one of the nontrivial square roots of 1, since knowledge of such an element would allow the prover to factor n. Therefore, $\alpha = \hat{S}_m/m^d$ must be of order at least p'. Finally, the expression $n - \varphi(n)/n$ is at most $1/p'$ in this case. The corollary then follows by replacing these values in the bound expression $ind(w)/ord(\hat{S}_m/m^d) + 2((n - \varphi(n)/n)$ in Lemma 4. $\square$

5.3. *Performance Analysis*

Computation. Under normal operation of the system, i.e., when no faults occur, each player needs to compute a partial signature which is a single exponentiation of an exponent which is the length of the RSA modulus.

When faults occur the players need to engage in Interactive Verification. Each such verification requires P to carry out three exponentiations of exponents of the size of the RSA modulus, and a single commitment which is an efficient computation. The verifier needs to compute four exponentiations.

When a player fails the interactive verification test, which means that it is faulty, the private share of this player is reconstructed. This is done using the VSS Reconstruction Phase which is a computation which requires only a polynomial number in N of multiplications.

Memory. The size of the memory required for each player is $(N + 1) \log n$.

Communication. Signature generation requires one round of communication. If Interactive Verification is invoked this requires four additional rounds of communication, and the VSS Reconstruction Phase requires another additional round.

6. A Secure Threshold RSA Scheme

In this section we prove that the combination of the [Ra2] threshold RSA scheme (see Fig. 1 in Section 3.3) with the robustness techniques developed in Sections 4 and 5 yields two secure solutions to threshold RSA signature schemes. Similar results can be shown by combining our techniques with other threshold RSA solutions [DF2], [DDFY], [Ra2].

Theorem 3. *Assuming factoring is hard there exist efficient, simulatable, and robust* (T, N)*-threshold RSA signature schemes for any value* T, $N \geq 2T + 1$, *where the RSA composite is the product of two safe primes* (*i.e.*, $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$, *and* p, p', q, q' *are all prime numbers*).

Remark (simulatability versus unforgeability). The theorem proves that the two constructions are *simulatable*. Using Proposition 1 it follows that the resultant threshold schemes are "as unforgeable as" the underlying signature scheme. Note that the hypothesis of the theorem simply assumes that factoring is hard. This is indeed sufficient to prove the simulatability and robustness of the schemes. However, in order to deduce unforgeability of the distributed schemes, one has to assume that the underlying signature scheme is unforgeable (which is not known to rely only on the hardness of factoring). Typically, such a scheme uses an encoding of the message (e.g., using a cryptographic hash function) to which the RSA decryption function is applied.

Overview. Fixing an adversary $\mathcal{A}$, we present simulators such that the view of the adversary on the execution of the protocol and its view on the execution of the simulator

Input: The number of players N and threshold T, the public key n, e, the message m, and its signature $S_m = m^d \bmod n$.

$$\mathcal{SIM} - 1$$

Secret Key Distribution

1. Choose shares $\hat{d}_1, \ldots, \hat{d}_{N-1} \in_R [1..n]$. Send $\hat{d}_1, \ldots, \hat{d}_T$ to players $P_1, \ldots, P_T$, respectively.
2. For each $\hat{d}_i$, $1 \le i \le N-1$, run the ICP-Gen-Integers protocol.
3. For the missing share $\hat{d}_N$ send to player P_j two random values $\hat{b}_{Nj} \in [2^{k_1}]$ and $\hat{c}_{Nj} \in [2^{k_3}]$.
4. Using a T-out-of-N VSS protocol (as in [BGW] and [Pe]) share among $P_1, \ldots, P_N$ the value $\hat{d}_i$ for $1 \le i \le N-1$ in a prime field Z_P where $P > n$.

Signature Generation

1. Compute $\hat{\sigma}_i \overset{\triangle}{=} m^{\hat{d}_i} \bmod n$ for $1 \le i \le N-1$.
2. Set $\hat{\sigma}_N \overset{\triangle}{=} S_m / \prod_{i=1}^{N-1} m^{\hat{d}_i}$.
3. Verify partial signatures:
   (a) For $1 \le i \le T$ carry out Noninteractive Verification (Fig. 3) as the verifier. Reconstruct the shares $\hat{d}_i$ of the players P_i $(1 \le i \le T)$ that fail the verification step (this reconstruction uses the fact that $\hat{d}_i$ was shared among all players during the secret key distribution phase).
   (b) For $T+1 \le i \le N-1$ carry out Noninteractive Verification (Fig. 3) as the prover.
   (c) For $i = N$ carry out Noninteractive Verification (Fig. 3) as the prover in the following way. Output $\hat{Y}_{Nj} = \hat{\sigma}_N^{b_{Nj}} m^{c_{Nj}}$

Fig. 6. Simulator for the noninteractive case.

are indistinguishable. For simplicity, and without loss of generality, we assume that the adversary corrupts players $P_1, \ldots, P_T$.

6.1. *Proof of the Noninteractive Scheme*

The robustness of the protocol is immediate, since Theorem 1 implies that the probability that an incorrect partial signature is accepted is $T \cdot (1/p' + 1/2^{k_1} + 1/2^{k_2})$ which can be made negligible.

The simulator is shown in Fig. 6. We analyze the information viewed by the adversary which is generated by an execution of the protocol and the simulator.

Secret Key Distribution.

1. $\hat{d}_i$ is a uniformly distributed value in $[1..n]$ and d_i is a uniformly distributed value in $[0..\varphi(n)-1]$ which are statistically indistinguishable distributions.
2. The verification data dealt during the simulation through ICP-Gen-Integers is distributed exactly the same as in the real execution (including for the share $\hat{d}_N$).
3. For each of the VSS (as the [BGW] VSS scheme is information theoretically secure) the view of the adversary is simply T uniformly distributed values in Z_P, and this is what the simulator generates.

Signature Generation.

1. The indistinguishability of the distribution of σ_i and $\hat{\sigma}_i$ follows from the argument on the distribution of d_i and $\hat{d}_i$ for $T+1 \leq i \leq N$.
2. The simulated Noninteractive Verification protocol for $1 \leq i \leq N$ is identical to a real execution in the protocol.

6.2. *Proof of the Interactive Scheme*

The robustness of the protocol is immediate, since Theorem 2 implies that the probability that an incorrect partial signature is accepted is $O(T)/p'$ which is negligible.

The simulator is shown in Fig. 7. We analyze the information viewed by the adversary which is generated by an execution of the protocol and the simulator.

Secret Key Distribution.

1. Clearly, $\hat{w}$ and $\hat{S}_w$ have the same exact distribution as in the real execution (a random message and its signature).
2. $\hat{d}_i$ is a uniformly distributed value in $[1..n]$ and d_i is a uniformly distributed value in $[0..\varphi(n)-1]$ which are statistically indistinguishable distributions.
3. The values $\hat{w}_1, \ldots, \hat{w}_{N-1}$ are generated in the same manner as $w_1, \ldots, w_{N-1}$.

Input: The number of players N and threshold T, the public key n, e, the message m, and its signature $S_m = m^d \bmod n$.

$$\mathcal{SIM}-2$$

Secret Key Distribution

1. Choose a random $\hat{S}_w \in Z_n^*$ and set $\hat{w} = \hat{S}_w^e \bmod n$.
2. Choose shares $\hat{d}_1, \ldots, \hat{d}_{N-1} \in_R [1..n]$. Give $\hat{d}_i$ to player P_i.
3. Compute $\hat{w}_1 \stackrel{\triangle}{=} \hat{w}^{\hat{d}_1}, \ldots, \hat{w}_{N-1} \stackrel{\triangle}{=} \hat{w}^{\hat{d}_{N-1}} \bmod n$. Set $\hat{w}_N \stackrel{\triangle}{=} \hat{S}_w / \prod_{i=1}^{N-1} \hat{w}_i \bmod n$.
4. Using the T-out-of-N VSS protocol (as in [BGW] and [Pe]) share among $P_1, \ldots, P_N$ the value $\hat{d}_i$ for $1 \leq i \leq N-1$ in a prime field Z_P where $P > n$. For $1 \leq i \leq T$ set P_i's share of $\hat{d}_N$ uniformly at random to a value in Z_P.

Signature Generation

1. Compute $\hat{\sigma}_i \stackrel{\triangle}{=} m^{\hat{d}_i} \bmod n$ for $T+1 \leq i \leq N-1$.
2. Set $\hat{\sigma}_N \stackrel{\triangle}{=} S_m / \prod_{i=1}^{N-1} \hat{\sigma}_i \bmod n$.
3. For $1 \leq i \leq T$ carry out Interactive Verification (Fig. 5) as the verifier. Reconstruct $\hat{d}_i$ for the players P_i who fail (this reconstruction uses the fact that $\hat{d}_i$ was shared among all players during the secret key distribution phase).
4. For $T+1 \leq i \leq N-1$ carry out Interactive Verification (Fig. 5) as the prover.
5. For $\hat{\sigma}_N, \hat{w}_N$ simulate Interactive Verification as follows:
   (a) Get the question Q and choose a random value r and compute *commit*(r).
   (b) Get the values i, j, compute $A_N = \hat{\sigma}_N^i \hat{w}_N^j \bmod n$. Rewind the simulator one step back and give *commit*(A_N).

Fig. 7. Simulator for the interactive case.

It is easily argued that the distribution of $\hat{w}_i$ where $\hat{d}_i \in [1..n]$ is statistically indistinguishable from the distribution of w_i where $d_i \in [0..\varphi(n) - 1]$.

4. The values $\hat{d}_1, \ldots, \hat{d}_{N-1}$ are statistically indistinguishable from $d_1, \ldots, d_{N-1}$, hence their sum mod $\varphi(n)$ is statistically indistinguishable from the sum of the d_i's, hence $\hat{d}_N$ mod $\varphi(n)$ is statistically indistinguishable from d_N mod $\varphi(n)$, thus the value $\hat{w}_N$ is statistically indistinguishable from w_N.
5. The VSS for the values $\hat{d}_i$, $1 \leq i \leq N - 1$, is identical to the execution in the real protocol.
6. The simulator does not know $\hat{d}_N$ and thus cannot share it using VSS, yet as the [BGW] VSS scheme is information theoretically secure the view of the adversary is simply T uniformly distributed values in Z_P, and this is what the simulator generates.

Signature Generation.

1. The indistinguishability of the distribution of σ_i mod n and $\hat{\sigma}_i$ mod n follows from the argument on the distribution of d_i and $\hat{d}_i$ for $1 \leq i \leq N$.
2. The Interactive Verification protocol for $1 \leq i \leq N-1$ is identical to the execution in the protocol.
3. Since the Interactive Verification protocol is zero-knowledge its simulation is indistinguishable from the real execution.

7. Verifying the Dealing Phase

Our basic security model assumes that the Dealing Phase is trusted, namely, that the adversary is not active during this phase. This is necessary due to the fact that during this phase the secret signature key is generated and held in a single place by the dealer. Recent work [BF] has presented progress toward distributed solutions for the generation of private RSA keys. At this point, this work may not be combined with our solutions due to the special form of the RSA modulus that we assumed for the analysis of our protocols. However, in many practical applications one can assume that the one-time generation of a private signature key can be done in a safe environment (e.g., using a protected computing device that is erased or destroyed after the dealing is done). In other applications it is in the interest of the dealer to keep its private key secret (e.g., in the case of distribution of the decryption capability for the sake of key backup or key escrow). This form of trust in the secrecy of the key does necessarily imply that the dealing process was performed correctly (e.g., a party escrowing a key may be interested in dealing the key incorrectly so it cannot be reconstructed later). Thus, it is important to ensure that once the Dealing Phase is over, the players have a guarantee that the system will work properly. In particular, that the information dealt to them is correct and consistent.

In this section we show how to verify that the dealer performs correctly the sharing of the keys and of the verification data in the interactive protocol of Section 5. This reduces significantly the trust required from the dealer at system initialization. This contribution of our work, i.e., specifying the steps to verify the correct dealing of the signature key, is important in addition to, and independently of, the aspect of verification of partial signatures.

We first present the main steps of the Dealing Phase including the measures to verify the actions of the dealer and other players. We then elaborate on some of these steps.

1. The dealer $\mathcal{D}$ chooses primes p and q, it sets the modulus $n = pq$, and chooses the private and public exponents d and e, respectively. These parameters are chosen as for any RSA system. We additionally impose on p and q that they be of special form, namely, $p = 2p' + 1$, $q = 2q' + 1$, where p', q' are prime numbers. $\mathcal{D}$ broadcasts the values of n and e, which will be used as the *public key* (or verification key) for the system's combined RSA signatures.
2. $\mathcal{D}$ shares the private exponent d as specified by the underlying threshold signature scheme (see Section 3.3). $\mathcal{D}$ sends to each player P_i its share d_i in a private but "openable way." By "openable way" we mean that the sender and/or recipient of the information can later show, if necessary, to other parties what the actual value transmitted was. This can be implemented by sending the value d_i from $\mathcal{D}$ to P_i encrypted under a public key of P_i (and later exposing the encrypted value to all players).
3. The N players collectively choose a random value w in Z_n^* (using standard techniques for collective coin tossing).
4. $\mathcal{D}$ broadcasts the values $w_i = w^{d_i} \bmod n$, $i = 1, \ldots, N$.
5. Each P_i checks that the value w_i broadcasted by $\mathcal{D}$ in the previous step corresponds to the values w and d_i as P_i holds. If not, P_i "opens" the value d_i sent from $\mathcal{D}$ in Step 2, and broadcasts it as an accusation against $\mathcal{D}$. If P_i's accusation is verified by a majority of other players, then $\mathcal{D}$ is disqualified. Otherwise it is P_i who is disqualified.
6. Each player verifies the correct dealing of d and the correct signatures w_i, for $i = 1, \ldots, N$, by performing the verification procedure explained below. A player that finds an inconsistency in this procedure votes for the disqualification of the dealer (votes are broadcast to all players). If a majority of such votes are collected, the dealer is disqualified. Otherwise, the players (if any) that vote for disqualification are disqualified.
7. The dealer $\mathcal{D}$ destroys all the secret information it generated, including p, q, d, and the shares d_i.

Disqualification of players is decided by majority vote, namely, we require more than $N/2$ players to agree before a player is disqualified. Notice that if the dealer is disqualified in the above protocol a new Dealing Phase (with a different dealer, rebooted machine, etc.) needs to be initiated. Disqualified players need not necessarily be replaced. The Dealing Phase can be successfully completed as long as there is a majority of good players (see Lemma 5 below). However, for simplicity, we assume that disqualified players are replaced (and a new Dealing Phase started). As a consequence, we assume that the Signature Generation Phase is started after the Dealing Phase is completed successfully with no player disqualified.

We elaborate on Step 6 of the above protocol.

VERIFICATION OF SHARES AND SAMPLE SIGNATURES. The following is a procedure by which each player can verify the correct dealing of d into the shares $d_1, \dots, d_N$, and the correct value of the sample signatures $w_i = w^{d_i} \bmod n$. (In what follows we omit the mod n notation.) Verifying the correctness of the shares means that the sum of the d_i's is d. The values w_i are correct if they correspond to the partial signatures w^{d_i} for the verified shares d_i. Furthermore, the players need to establish that the VSS back-up [Ra2] of all values d_i is in fact the sharing of the value committed to by w_i.

We get that a correct sharing is verified by the equation

$$d = \sum_{i=1}^{n} d_i. \tag{A.1}$$

In our case the explicit values of all the shares d_i are not available to each player, therefore the checking of correct dealing is done using the equivalent of the above equation "in the exponent," namely, each player verifies that (remember that $w_i = w^{d_i}$)[3]

$$w = \left(\prod_{i=1}^{n} w_i\right)^e. \tag{A.2}$$

In order to be able to claim that the verification of (A.2) implies the correctness of (A.1) we need to solve two problems. The first is the fact that there may be a value w_i which is not a power of w, i.e., there is no value t for which $w_i = w^t$. The second problem is that even if the values w_i are all exponents of w, the equality in (A.2) only implies that the equality in (A.1) holds modulo $ord(w)$, which may be a problem if w is an element of low order.

To verify that the values w_i are indeed exponents of w we use the following subprotocol [CEG]. For each $i \in [N]$ the dealer $\mathcal{D}$ chooses a value $r \in_R [\varphi(n)]$ and broadcasts $w' = w^r$. The players collectively choose a random bit b. If $b = 0$, $\mathcal{D}$ broadcasts the value r, otherwise it broadcasts the value $d_i + r \bmod \varphi(n)$. In the first case, each player can check whether $w^r = w'$, and in the second, whether $w^{(r+d_i)} = w'w_i$. If $w_i \notin \langle w \rangle$, then the probability that $\mathcal{D}$ passes this test is $1/2$. By repeating this procedure k times the probability that the dealer can cheat goes down to 2^{-k}. This protocol can be done noninteractively if one is willing to assume the existence of ideal hashing functions (random oracle).

As for the problem that the equality (A.1) is verified only modulo $ord(w)$, we point out that because of the assumed form of p and q (i.e., $(p-1)/2$ and $(q-1)/2$ being prime numbers), the order of a random element w is equal to $\varphi(n)/2$ or $\varphi(n)/4$ with overwhelming probability. In the former case, the order of w is a multiple of the order of all other elements in Z_n^* and then (A.2) implies (A.1). In the case $ord(w) = \varphi(n)/4$, the element $-w$ is of order $\varphi(n)/2$. Therefore, one solution to the above problem is to repeat the described process for both w and $-w$. If the above verification procedure

[3] This is a specific verification which relates to the specific form in which the secret key d is shared in [Ra2]. If the sharing of d is done via a polynomial (this may occur in other threshold schemes), then the fact that all the committed values interpolate into a single polynomial of degree T can be verified in the exponent as well.

is completed successfully for both values, then only the value of w is carried to the Signature Generation Phase.[4]

In order to verify that the VSS sharing of the values d_i correspond to the desired value committed to by w_i we change the VSS sharing slightly. Instead of sharing the values over a prime field we use a sharing using Feldman's VSS over the integers. For details see [Ra2] and [FGMY]. The Feldman secret sharing is designed is such a manner that in addition to the sharing of a value, say d_i, it also exposes the value $w_i = w^{d_i}$. As the value w_i is exactly our witness value we immediately have the guarantee that the shared value is the value committed to by the witness.

VERIFICATION OF THE PRIME FACTORS. We need to check that the dealer chooses the modulus n of the right form, i.e., $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$. Recently, Gennaro et al. [GMR3] have presented a zero-knowledge proof to verify that a composite is of a slightly different form, where p, q are of the form $p = 2p_1^{\alpha} + 1$ and $q = 2q_1^{\beta} + 1$. Applying their techniques in our setting equates the dealers' probability of cheating with the probability of factoring his composite. See [GMR3] for details.

We summarize the properties of the Dealing Phase in the following lemma.

Lemma 5. *Assume that the composite n is chosen as specified and let T, $N \geq 2T + 1$. If there are at most T cheating players during the above Dealing Phase, and the dealer is not disqualified, then the good players end that phase with correct partial signatures on w for every nondisqualified player, and the corresponding shares d_i sum to the correct exponent d, as chosen by $\mathcal{D}$. Moreover, if the dealer is honest nothing is learned by any of the players that can help a coalition of less than T players to forge a signature.*

8. Conclusions and Further Applications

We presented two protocols for verifying partial signatures. The first protocol is a noninteractive one, the second is interactive, yet provides the ability to have public verification of the partial signatures. Both protocols are low on computation and communication, thus, achieving an efficient, robust, threshold-RSA signature scheme.

Our techniques are closely related to the notion of undeniable signatures [CA]. Recently, Gennaro et al. [GKR] achieved RSA-based undeniable signatures which build on the Interactive Verification protocol presented and analyzed here (previous undeniable signature schemes were based on discrete logarithm-based systems). Undeniable signatures are characterized by the fact that public information is not sufficient in order to verify the signature but interaction with the signer is required for such verification. The techniques based in our work can be further applied to separate between the signing and verification processes in the sense that a signer could delegate the ability to verify signatures to a third party while the latter cannot forge signatures.

The techniques developed here for shared RSA signature generation apply to shared

[4] Another option is to test only w. If $ord(w) = \varphi(n)/2$, then no cheating for $\mathcal{D}$ is possible. If $ord(w) = \varphi(n)/4 = p'q'$, then the only possible cheating by $\mathcal{D}$ is to deal instead of the right exponent d, the exponent $d' = d + p'q'$ (or $d' = d + 3p'q'$) which satisfies all equations for w but not for values w' of order $2p'q'$. However, even in this case the equations are satisfied up to their sign (since in this case $w'^{d'} = -w'^{d}$), and as stated in Section 3.4 getting a right signature except for the wrong sign is acceptable in our setting.

RSA decryption as well. Using these mechanisms, a user (acting as the dealer) shares its private decryption key with a set of agents, such that the cooperation of at least a threshold of these agents is required in order to decrypt messages intended for that user; no coalition of less than T agents can decrypt such messages or learn about the user's decryption key. This has natural applications to key escrow systems. In such an application the verifiability of the dealer's actions is particularly important since the user (acting as a dealer) may have a strong interest to share the wrong key in order to prevent the eventual decryption by the agents of messages intended for him/her.

References

[BCC] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. System Sci.*, 37(2):156–189, 1988.

[BCDP] J. Boyar, D. Chaum, I. Damgård, and T. Pedersen. Convertible undeniable signatures. In A. J. Menezes and S. Vanstone, editors, *Advances in Cryptology — Crypto* '90, pages 189-205. Springer-Verlag, Berlin, 1990. Lecture Notes in Computer Science No. 537.

[BF] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In B. Kaliski, editor, *Advances in Cryptology — Crypto* '97, pages 425-439. Springer-Verlag, Berlin, 1997. Notes in Computer Science No. 1294.

[BGW] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proc.* 20*th Annual Symp. on the Theory of Computing*, pages 1–10. ACM, New York, 1988.

[BK] M. Blum and S. Kannan. Program correctness checking and the design of programs that check their work. In *Proc.* 21*st Annual Symp. on the Theory of Computing*, pages 86–97. ACM, New York, 1989.

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

[BR] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. De Santis, editor, *Advances in Cryptology — Eurocrypt* '94, pages 92-111. Springer-Verlag, Berlin, 1994. Lecture Notes in Computer Science No. 950.

[CA] D. Chaum and H. Van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology — Crypto* '89, pages 212-217. Springer-Verlag, Berlin, 1989. Lecture Notes in Computer Science No. 435.

[CCD] D. Chaum, C. Crepeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proc.* 20*th Annual Symp. on the Theory of Computing*, pages 11–19. ACM, New York, 1988.

[CEG] D. Chaum, J.-H. Evertse, and J. van der Graaf. An improved protocol for demonstrating possession of a discrete logarithm and some generalizations. In D. Chaum, editor, *Advances in Cryptology — Eurocrypt* '87, pages 127-141. Springer-Verlag, Berlin, 1987. Lecture Notes in Computer Science No. 304.

[Ch] D. Chaum. Zero-knowledge undeniable signatures. In I. Damgård, editor, *Advances in Cryptology — Eurocrypt* '90, pages 458-464. Springer-Verlag, Berlin, 1990. Lecture Notes in Computer Science No. 473.

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

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

[DDFY] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proc.* 26*th Annual Symp. on the Theory of Computing*, pages 522–533. ACM, New York, 1994.

[DeL] J. M. DeLaurentis. A further weakness in the common modulus protocol for RSA cryptosytems. *Cryptologia*, (8):253–259, 1984.

[Des1] Y. Desmedt. Society and group oriented cryptography: a new concept. In C. Pomerance, editor, *Advances in Cryptology — Crypto* '87, pages 120-127. Springer-Verlag, Berlin, 1987. Lecture Notes in Computer Science No. 293.

[Des2] Y. G. Desmedt. Threshold cryptography. *European Trans. Telecommun.*, 5(4):449–457, July 1994.

[DF1] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology — Crypto* '89, pages 307-315. Springer-Verlag, Berlin, 1989. Lecture Notes in Computer Science No. 435.

[DF2] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto* '91, pages 457-469. Springer-Verlag, Berlin, 1991. Lecture Notes in Computer Science No. 576.

[FGMY] Y. Frankel, P. Gemmell, P. Mackenzie, and M. Yung. Optimal resilience proactive public-key cryptosystems. In *Proc.* 38*th Annual Symp. on Foundations of Computer Science*, pages 384–393. IEEE, New York, 1997.

[FGY] Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking and robust function sharing. In *Proc.* 28*th Annual Symp. on the Theory of Computing*, pages 499–508. ACM, New York,1996.

[GGJR] J. Garay, R. Gennaro, C. Jutla, and T. Rabin. Secure distributed storage and retrieval. In M. Mavronicolas and P. Tsigas, editors, *Proc.* 11*th International Workshop, WDAG* '97, pages 275-289. Springer-Verlag, Berlin, 1997. Lecture Notes in Computer Science No. 1320.

[GJKR1] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In N. Koblitz, editor, *Advances in Cryptology — Crypto* '96, pages 157-172. Springer-Verlag, Berlin, 1996. Lecture Notes in Computer Science No. 1109.

[GJKR2] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. Maurer, editor, *Advances in Cryptology — Eurocrypt* '96, pages 354-371. Springer-Verlag, Berlin, 1996. Lecture Notes in Computer Science No. 1070.

[GKR] R. Gennaro, H. Krawczyk, and T. Rabin. RSA-based undeniable signatures. In B. Kaliski, editor, *Advances in Cryptology—Crypto* '97, pages 132-149. Springer-Verlag, Berlin, 1997. Lecture Notes in Computer Science No. 1294.

[GM] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. System Sci.*, 28(2):270–299, April 1984.

[GMR1] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988.

[GMR2] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM. J. Comput.*, 18(1):186–208, February 1989.

[GMR3] R. Gennaro, D. Micciancio, and T. Rabin. An efficient noninteractive statistical zero-knowledge proof system for quasi-safe prime products. To appear in the 1998 ACM Conference on Computer and Communication Security.

[GMW1] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc.* 19*th Annual Symp. on the Theory of Computing*, pages 218–229. ACM, New York, 1987.

[GMW2] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. Assoc. Comput. Mach.*, 38(1):691–729, 1991.

[Go] O. Goldreich. *Foundation of Cryptography—Fragments of a Book*. Electronic Colloquium on Computational Complexity, February 1995. Available online from http://www.eccc.uni-trier.de/eccc/.

[Mi] S. Micali. Fair public-key cryptosystems. In E. Brickell, editor, *Advances in Cryptology — Crypto* '92, pages 113-138. Springer-Verlag, Berlin, 1992. Lecture Notes in Computer Science No. 740.

[MR] S. Micali and P. Rogaway. Secure computation. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto* '91, pages 392-404. Springer-Verlag, Berlin, 1991. Lecture Notes in Computer Science No. 576.

[Pe] T. Pedersen. Noninteractive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto* '91, pages 129-140. Springer-Verlag, Berlin, 1991. Lecture Notes in Computer Science No. 576.

[Ra1] T. Rabin. Robust sharing of secrets when the dealer is honest or faulty. *J. Assoc. Comput. Mach.*, 41(6):1089–1109, 1994.

[Ra2] T. Rabin. A simplified approach to threshold and proactive RSA. In *Proc. Crypto* '98. Springer-Verlag, Berlin, 1998. Lecture Notes in Computer Science No. 1462.

[RB] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc.* 21*st Annual Symp. on the Theory of Computing*, pages 73–85. ACM, New York, 1989.

[Wi] M. J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Trans. Inform. Theory*, 36(3):553–558, May 1990.