# Practical Threshold Signatures*

Victor Shoup
IBM Zurich Research Lab, Säumerstr. 4, 8803 Rüschlikon, Switzerland
sho@zurich.ibm.com

June 24, 1999

**Abstract**

We present an RSA threshold signature scheme. The scheme enjoys the following properties:

1. it is unforgeable and robust in the random oracle model, assuming the RSA problem is hard;
2. signature share generation and verification is completely non-interactive;
3. the size of an individual signature share is bounded by a constant times the size of the RSA modulus.

## 1 Introduction

A $k$ out of $l$ threshold signature scheme is a protocol that allows any subset of $k$ players out of $l$ to generate a signature, but that disallows the creation of a valid signature if fewer than $k$ players participate in the protocol. This *non-forgeability* property should hold even if some subset of less than $k$ players are corrupted and work together. For a threshold scheme to be useful when some players are corrupted, it should should also be *robust*, meaning that corrupted players should not be able to prevent uncorrupted players from generating signatures.

The notion of a threshold signature scheme has been extensively studied. However, all previously proposed schemes suffer from at least one of the following problems:

*This is a revision of IBM Research Report RZ 3121 (April 19, 1999).

1. the scheme has no rigorous security proof, even in the random oracle model;

2. signature share generation and/or verification is interactive, moreover requiring a synchronous communications network;

3. the size of an individual signature share blows up linearly in the number of players.

To correct this situation, we present a new threshold RSA signature scheme that enjoys the following properties:

1. it is unforgeable and robust in the random oracle model, assuming the RSA problem is hard;

2. signature share generation and verification is completely non-interactive;

3. the size of an individual signature share is bounded by a small constant times the size of the RSA modulus.

Moreover, the resulting signature is a standard RSA signature.

Our scheme is exceedingly simple, and it is truly amazing that such a scheme has apparently not been previously proposed and analyzed.

We also consider a more refined notion of a threshold signature scheme, where there is one threshold $t$ for the maximum number of corrupt players, and another threshold $k$ for the minimum quarum size. The fact that a particular message has been signed means that at least $k - t$ uncorrupted players have authorized the signature.

Previous investigations into threshold signature schemes have always assumed (explicitly or implicitly) that $k = t + 1$. We also investigate the more general setting where $k \geq t + 1$. This generalization is useful in situations where the uncorrupted parties do not necessarily agree on what they are signing, but one wants to be able to prove that a large number of them have authorized a signature. Indeed, threshold signatures with $k \approx 2l/3$ and $t \approx l/3$ can be exploited to reduce the sizes of the messages sent in Byzantine agreement protocols in an asynchronous network.

We work with a "static corruption model": the adversary must choose which players to corrupt at the very beginning the attack. This is in line with previous investigations into threshold signatures, which also (explicitly or implicitly) assume static corruptions.

Our basic scheme, Protocol 1, can be proven secure when $k = t + 1$ in the random oracle model under the RSA assumption.

We present another scheme, Protocol 2, for use in the more general setting $k \geq t+1$. Protocol 2 can be be proven secure—again, in the random oracle model—when $k = t + 1$ under the RSA assumption, and when $k > t + 1$ under an additional assumption, namely, an appropriate variant of the Decision Diffie-Hellman assumption.

As already mentioned, our proofs of security are valid in the so-called "random oracle model," where cryptographic hash functions are replaced by a random oracle. This model was used informally by Fiat and Shamir [9], and later was rigorously formalized and more fully exploited in Bellare and Rogaway [1], and thereafter used in numerous papers.

For Protocol 1, we only need random oracles for robustness, if we assume that ordinary RSA signatures are secure. In fact, Gennaro *et al.* [13] present a non-interactive share verification scheme that can be analyzed without resorting to random oracles. One could use their verification scheme in place of the one we suggest, thus avoiding random oracles in the analysis, but this would have certain practical drawbacks, requiring a special relationship between the sender and recipient of a share of a signature.

The analysis of Protocol 2 makes use of the random oracle model in a more fundamental way. Since this seemed inevitable, we took several liberties in the design of Protocol 2, so that it is actually a bit simpler and more efficient than Protocol 1. Thus, even if $k = t + 1$, Protocol 2 may be an attractive practical alternative to Protocol 1.

We view a proof of security in the random oracle model as a heuristic argument that provides strong evidence that a system cannot be broken. All things being equal, a proof of security in the random oracle model is not as good as a proof of security in the "real world," but is much better than no proof at all. Anyway, it does not seem unreasonable to use the random oracle model, since that is the only way we know of to justify the security of ordinary RSA signatures.

### Previous Work

Desmedt [4] introduces the more general notion of threshold signatures. Desmedt and Frankel [5] present a non-robust threshold ElGamal scheme [7] based on "secret sharing," [19] i.e., polynomial interpolation over a finite field. Their scheme has small share size, but requires (synchronous) interaction. Harn [15] presents a robust threshold ElGamal scheme with small share size, but again requires interaction. It seems that the security of both of the above schemes can be rigorously analyzed in a satisfactory way, although neither paper does this. Gennaro *et al.* [14] present a robust threshold DSS scheme with small share size that again requires interaction; they also give a rigorous security analysis.

All of the above-mentioned schemes are interactive. Indeed, any threshold signature scheme based on discrete logarithms appears doomed to be interactive, since all such signature schemes are randomized, and so the signers have to generate random values jointly, which apparently requires interaction.

In [5], Desmedt and Frankel also briefly address the problem of designing a threshold RSA [18] signature scheme, noting that there are some technical obstructions to doing this arising from the fact that polynomial interpolation over the coefficient ring $\mathbf{Z}_{\phi(n)}$, where $n$ is the RSA modulus and $\phi$ the Euler totient function, is somewhat awkward. Later, Desmedt and Frankel [6] return again to the problem of threshold RSA, and present a non-robust threshold RSA scheme that is non-interactive and with small share size, but with no security analysis. Frankel and Desmedt [10] present results extending those in [6], giving a proof of security for a non-robust threshold RSA scheme with small share size, but which requires interaction. Later, De Santis *et al.* [3] present a variation (also non-robust) on the scheme in [10] that trades interaction for large share size (growing linearly in the number of players). Both [10] and [3] avoid the problems of polynomial interpolation over $\mathbf{Z}_{\phi(n)}$ by working instead with over $\mathbf{Z}_{\phi(n)}[X]/(\Phi_q(X))$, where where $\Phi_q(X)$ is the $q$th cyclotomic polynomial (taken mod $\phi(n)$), and $q$ is a prime greater than $l$. This is convenient, as standard secret sharing techniques can then be directly applied, but it leads to a much more complicated schemes that also require either interaction or large share sizes.

Gennaro *et al.* [13] give a few general techniques that allow one to make RSA threshold systems robust.

Later, Frankel *et al.* [12, 11] and Rabin [17] propose and rigorously analyze robust threshold RSA schemes that have small share size, but require interaction. These papers take a different approach to the "interpolation over $\mathbf{Z}_{\phi(n)}$ problem," sidestepping it by introducing an extra layer of "secret sharing" and *much* more interaction and complexity. These schemes have other features as well, namely they provide a type of security known as "pro-active security," a topic we do not address here at all.

As we shall see, the "interpolation over $\mathbf{Z}_{\phi(n)}$ problem" is not really a problem at all—it is entirely trivial to work around the minor technical difficulties to obtain an extremely simple and provably secure threshold RSA scheme. We do not even need a random oracle if we do not require robustness and we are willing to assume that the RSA signature scheme is itself secure.

### Organization

In §2 we describe our system model and security requirements for threshold signatures. In §3 we describe Protocol 1. In §4 we analyze Protocol 1 in the case $k = t + 1$. In §5 we present Protocol 2, and analyze it in the more general case $k \geq t + 1$.

## 2 System model and security requirements

**The participants.** We have a set of $l$ players, indexed $1, \ldots, l$, a *trusted dealer*, and an *adversary*. There is also a *signature verification algorithm*, a *share verification algorithm*, and a *share combining algorithm*.

There are two parameters:

$t$—the number of corrupted players;

$k$—the number of signature shares needed to obtain a signature.

The only requirements are that $k \geq t + 1$ and $l - t \geq k$.

**The action.** At the beginning of the game, the adversary selects a subset of $t$ players to *corrupt*.

In the *dealing phase*, the dealer generates a public key $PK$ along with secret key shares $SK_1, \ldots, SK_l$, and verification keys $VK$, $VK_1, \ldots, VK_l$. The adversary obtains the secret key shares of the corrupted players, along with the public key and verification keys.

After the dealing phase, the adversary submits signing requests to the uncorrupted players for messages of his choice. Upon such a request, a player outputs a *signature share* for the given message.

**Robustness and combining shares.** The signature verification algorithm takes a input a message and a signature, along with the public key, and determines if the signature is valid. The signature share verification algorithm takes as input a message, a signature share on that message from a player $i$, along with $PK$, $VK$, and $VK_i$, and determines if the signature share is valid. The share combining algorithm takes as input a message and $k$ valid signature shares on the message, along with the public key and (perhaps) the verification keys, and outputs a valid signature on the message.

**Non-forgeability.** We say that the adversary *forges a signature* if at the end of the game he outputs a valid signature on a message that was not submitted as a signing request to at least $k-t$ uncorrupted players. We say that the threshold signature scheme is *non-forgeable* if it is computationally infeasible for the adversary to forge a signature.

**Discussion.** Notice that our model explicitly requires that the generation and verification of signature shares is completely non-interactive.

Also notice that we have two independent parameters $t$ and $k$. As mentioned in the introduction, previous investigations into threshold signatures have only dealt with the case $k = t+1$. In this case, the non-forgeability requirement simply says that a signature is forged if no uncorrupted player was asked to sign it. As we shall see, achieving non-forgeability when $k > t+1$ is harder to do than when $k = t+1$. For simplicity, we shall start with the case $k = t+1$.

## 3 Protocol 1: a very simple RSA threshold scheme

We now describe Protocol 1, which will be analyzed in the next section when $k = t+1$.

**The dealer.** The dealer chooses at random two large primes of equal length (512 bit, say) $p$ and $q$, where $p = 2p' + 1$, $q = 2q' + 1$, with $p'$, $q'$ themselves prime. The RSA modulus is $n = pq$. Let $m = p'q'$. The dealer also chooses the RSA public exponent $e$ as a prime $e > l$.

The public key is $PK = (n, e)$.

Next, the dealer computes $d \in \mathbf{Z}$ such that $de \equiv 1 \bmod m$. The dealer sets $a_0 = d$ and chooses $a_i$ at random from $\{0, \ldots, m-1\}$ for $1 \le i \le k-1$. The numbers $a_0, \ldots, a_{k-1}$ define the polynomial $f(X) = \sum_{i=0}^{k-1} a_i X^i \in \mathbf{Z}[X]$.

Let $L(n)$ be the bit-length of $n$. For $1 \le i \le l$, the dealer computes

$$s_i = f(i) \bmod m. \tag{1}$$

This number $s_i$ is the secret key share $SK_i$ of player $i$.

We denote by $Q_n$ the subgroup of squares in $\mathbf{Z}_n^*$.

Next, the dealer chooses a random $v \in Q_n$, and for $1 \le i \le l$ computes $v_i = v^{s_i} \in Q_n$. These elements define the verification keys: $VK = v$, and $VK_i = v_i$.

**Some preliminary observations.** Note that $\mathbf{Z}_n^* \simeq \mathbf{Z}_m \times \mathbf{Z}_2 \times \mathbf{Z}_2$. If we let $J_n$ denote the subgroup of elements $x \in \mathbf{Z}_n^*$ with Jacobi symbol $(x|n) = 1$, then we have $Q_n \subset J_n \subset \mathbf{Z}_n^*$; moreover, $Q_n$ is cyclic of order $m$ and $J_n$ is cyclic of order $2m$. Also, $-1 \in J_n \backslash Q_n$.

Generally speaking, we shall ensure that all group computations are done in $Q_n$, and corresponding exponent arithmetic in $\mathbf{Z}_m$. This is convenient, since $m = p'q'$ has no small prime factors.

Since the dealer chooses $v \in Q_n$ at random, we may assume that $v$ generates $Q_n$, since this happens with all but negligible probability. Because of this, the values $v_i$ completely determine the values $s_i \bmod m$.

For any subset of $k$ points in $\{0, \ldots, l\}$, the value of $f(X)$ modulo $m$ at these points uniquely determines the coefficients of $f(X)$ modulo $m$, and hence the value of $f(X)$ modulo $m$ at any other point modulo in $\{0, \ldots, l\}$. This follows from the fact the corresponding Vandermonde matrix is invertible modulo $m$, since its determinant is relatively prime to $m$.

From this, it follows that for any subset of $k-1$ points in $\{1, \ldots, l\}$, the distributions of the value of $f(X)$ modulo $m$ at these points are uniform and mutually independent.

Let $\Delta = l!$. For any subset $S$ of $k$ points in $\{0, \ldots, l\}$, and for any $i \in \{0, \ldots, l\} \backslash S$, and $j \in S$, we can define

$$\lambda_{i,j}^S = \Delta \frac{\prod_{j' \in S \backslash \{j\}} (i - j')}{\prod_{j' \in S \backslash \{j\}} (j - j')} \in \mathbf{Z}. \tag{2}$$

These values are derived from the standard Lagrange interpolation formula. They are clearly integers, since the denominator divides $j!(l-j)!$ which in turn divides $l!$. It is also clear that these values are easy to compute. From the Lagrange interpolation formula, we have:

$$\Delta \cdot f(i) \equiv \sum_{j \in S} \lambda_{i,j}^S f(j) \bmod m. \tag{3}$$

**Valid signatures.** We next describe what a valid signature looks like. We need a hash function $H$ mapping messages to elements of $\mathbf{Z}_n^*$. If $x = H(m)$, then a valid signature on $m$ is $y \in \mathbf{Z}_n^*$ such that $y^e = x$. This is just a classical RSA signature.

**Generating a signature share.** We now describe how a signature share on a message $m$ is generated. Let $x = H(m)$. The signature share of player $i$ consists of

$$x_i = x^{2\Delta s_i} \in Q_n, \tag{4}$$

along with a "proof of correctness."

The proof of correctness is basically just a proof that the discrete logarithm of $x_i^2$ to the base

$$\tilde{x} = x^{4\Delta} \tag{5}$$

is the same as the discrete logarithm of $v_i$ to the base $v$. For this, we can easily adapt a well-known interactive protocol, due to Chaum and Pedersen [2]. We "collapse" the protocol, making it non-interactive, by using a hash function to create the challenge—this is where the random oracle model will be needed. We also have to deal with the fact that we are working in a group $Q_n$ whose order is not known. But this is trivially dealt with by just working with sufficiently large integers.

Now the details. Let $H'$ be a hash function, whose output is an $L_1$-bit integer, where $L_1$ is a secondary security parameter ($L_1 = 128$, say). To construct the proof of correctness, player $i$ chooses a random number $r \in \{0, \ldots, 2^{L(n)+2L_1} - 1\}$, computes

$$v' = v^r, \; x' = \tilde{x}^r, \; c = H'(v, \tilde{x}, v_i, x_i^2, v', x'), \; z = s_i c + r.$$

The proof of correctness is $(z, c)$.

To verify this proof of correctness, one checks that

$$c = H'(v, \tilde{x}, v_i, x_i^2, v^z v_i^{-c}, \tilde{x}^z x_i^{-2c}).$$

The reason for working with $x_i^2$ and not $x_i$ is that although $x_i$ is supposed to be a square, this is not easily verified. This way, we are sure to be working in $Q_n$, where we need to be working to ensure soundness.

**Combining shares.** We next describe how signature shares are combined. Suppose we have valid shares from a set $S$ of players, where $S = \{i_1, \ldots, i_k\} \subset \{1, \ldots, l\}$.

Let $x = H(m) \in \mathbf{Z}_n^*$, and assume that $x_{i_j}^2 = x^{4\Delta s_{i_j}}$. Then to combine shares, we compute

$$w = x_{i_1}^{2\lambda_{0,i_1}^S} \cdots x_{i_k}^{2\lambda_{0,i_k}^S},$$

where the $\lambda$'s are the integers defined in (2). From (3), we have $w^e = x^{e'}$, where

$$e' = 4\Delta^2. \tag{6}$$

Since $\gcd(e', e) = 1$, it is easy to compute $y$ such that $y^e = x$, using a standard algorithm: $y = w^a x^b$ where $a$ and $b$ are integers such that $e'a + eb = 1$, which can be obtained from the extended Euclidean algorithm on $e'$ and $e$.

# 4 Security analysis of Protocol 1

**Theorem 1** *For $k = t + 1$, in the random oracle model for $H'$, Protocol 1 is a secure threshold signature scheme (robust and non-forgeable) assuming the the standard RSA signature scheme is secure.*

We show how to simulate the adversary's view, given access to an RSA signing oracle which we use only when the adversary asks for a signature share from an uncorrupted player.

Let $i_1, \ldots, i_{k-1}$ be the set of corrupted players. Recall $s_i \equiv f(i) \bmod m$ for all $1 \leq i \leq l$, and $d \equiv f(0) \bmod m$.

To simulate the adversary's view, we simply choose the $s_{i_j}$ belonging to the set of corrupted players at random from the set $\{0, \ldots, \lfloor n/4 \rfloor - 1\}$. We have already argued that the the corrupted players' secret key shares are random numbers in the set $\{0, \ldots, m-1\}$. We have

$$n/4 - m = (p' + q')/2 + 1/4 = O(n^{1/2}),$$

and from this a simple calculation shows that the statistical distance between the uniform distribution on $\{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and the uniform distribution on $\{0, \ldots, m-1\}$ is $O(n^{-1/2})$.

Once these $s_{i_j}$ values are chosen, the values $s_i$ for the uncorrupted players are also completely determined modulo $m$, but cannot be easily computed. However, given $x, y \in \mathbf{Z}_n^*$ with $y^e = x$, we can easily compute $x_i = x^{2\Delta s_i}$ for an uncorrupted player $i$ as

$$x_i = y^{2(\lambda^S_{i,0} + e(\lambda^S_{i,i_1} s_{i_1} + \cdots + \lambda^S_{i,i_{k-1}} s_{i_{k-1}}))},$$

where $S = \{0, i_1, \ldots, i_{k-1}\}$. This follows from (3).

Using this technique, we can generate the values $v, v_1, \ldots, v_l$, and also generate any share $x_i$ of a signature, given the standard RSA signature.

This argument shows why we defined the share $x_i$ to be $x^{2\Delta s_i}$, instead of, say, $x^{2 s_i}$. This same idea was used by Feldman [8] in the context of the different but related problem of verifiable secret sharing.

With regard to the "proofs of correctness," one can invoke the random oracle model for the hash function $H'$ to get soundness and statistical zero-knowledge. This is quite straightforward, but we sketch the details.

First, consider soundness. We want to show that the adversary cannot construct, except with negligible probability, a proof of correctness for an incorrect share. Let $x$ and $x_i$ be given, along with a valid proof of correctness $(z, c)$. We have $c = H'(v, \tilde{x}, v_i, x_i^2, v', x')$, where

$$\tilde{x} = x^{4\Delta}, \ v' = v^z v_i^{-c}, \ x' = \tilde{x}^z x_i^{-2c}.$$

Now, $\tilde{x}, v_i, x_i^2, v', x'$ are all easily seen to lie in $Q_n$, and we are assuming that $v$ generates $Q_n$. So we have

$$\tilde{x} = v^{\alpha}, \ v_i = v^{s_i}, \ x_i^2 = v^{\beta}, v' = v^{\gamma}, \ x' = v^{\delta},$$

for some integers $\alpha, \beta, \gamma, \delta$. Moreover,

$$z - cs_i \equiv \gamma \bmod m \quad \text{and} \quad z\alpha - c\beta \equiv \delta \bmod m.$$

Multiplying the first equation by $\alpha$ and subtracting the second, we have

$$c(\beta - s_i\alpha) \equiv \alpha\gamma - \delta \bmod m. \tag{7}$$

Now, a share is correct if and only if

$$\beta \equiv s_i\alpha \bmod m. \tag{8}$$

If (8) fails to hold, then it must fail to hold mod $p'$ or mod $q'$, and so (7) uniquely determines $c$ modulo one of these primes. But in the random oracle model, the distribution of $c$ is uniform and independent of the inputs to the hash function, and so this even happens with negligible probability.

Second, consider zero-knowledge simulatability. We can construct a simulator that simulates the adversary's view without knowing the value $s_i$. This view includes the values of the random oracle at those points where the adversary has queried the oracle, so the simulator is in complete charge of the random oracle. Whenever the adversary makes a query to the random oracle, if the oracle has not been previously defined at the given point, the simulator defines it to be a random value, and in any case returns the value to the adversary. When an uncorrupted player is supposed to generate a proof of correctness for a given $x$, $x_i$, the simulator chooses $c \in \{0, \ldots, 2^{L_1} - 1\}$ and $z \in \{0, \ldots, 2^{L(n)+2L_1} - 1\}$ at random, and for given values $x$ and $x_i$, defines the value of the random oracle at $(v, \tilde{x}, v_i, x_i^2, v^z v_i^{-c}, \tilde{x}^z x_i^{-2c})$ to be $c$. With all but negligible probability, the simulator has not defined the random oracle at this point before, and so it is free to do so now. The proof is just $(z, c)$. It is straightforward to verify that the distribution produced by this simulator is statistically close to perfect.

From soundness, we get the robustness of the threshold signature scheme. From zero-knowledge, and the above arguments, we get the non-forgeability of the threshold signature scheme, assuming that the standard RSA signature scheme is secure, i.e., existentially non-forgeable against adaptive chosen message attack. This last assumption can be further justified (see [1]): in the random oracle model for $H$, this assumption follows from the RSA assumption—given random $x \in \mathbf{Z}_n^*$, it is hard to compute $y$ such that $y^e = x$.

## 5 Protocol 2: a modification and security analysis when $k \geq t+1$

We now present Protocol 2 and analyze its security when $k \geq t+1$. In our analysis of Protocol 2, we need to make use of the random oracle model in a fundamental way. As such, we fully exploit the random oracle model to get a scheme that is a bit simpler and more efficient that Protocol 1.

Protocol 2 is obtained by modifying Protocol 1 as follows.

Instead of computing the secret key share $s_i$ as in (1), the dealer computes it as

$$s_i = f(i)\Delta^{-1} \bmod m.$$

We add to the verification key $VK$ an element $u \in \mathbf{Z}_n^*$ with Jacobi symbol $(u|n) = -1$. Note that the Jacobi symbol can be efficiently computed, and such a $u$ can be found just by random sampling.

We then modify the share generation algorithm as follows. Let $\hat{x} = H(m)$. We set

$$x = \begin{cases} \hat{x} & \text{if } (\hat{x}|n) = 1; \\ \hat{x}u^e & \text{if } (\hat{x}|n) = -1. \end{cases}$$

This forces the Jacobi symbol of $x$ to be 1. The share generation, verification, and combination algorithms then run as before, using this new value of $x$, except that we make the following simplifications: we redefine $x_i$, $\tilde{x}$, and $e'$ (defined in (4), (5), and (6)) as

$$x_i = x^{2s_i}, \ \tilde{x} = x^4, \ e' = 4.$$

Thus, we eliminate the somewhat "artificial" appearances of $\Delta$ in the share generation and combination algorithms.

The original share combination algorithm produces $y$ such that $y^e = x$. If $x = \hat{x}u^e$, then we can divide $y$ by $u$, obtaining an $e$th root of $H(m)$, so we still obtain a standard RSA signature.

That completes the description of Protocol 2.

To analyze the security of Protocol 2, we will need to work in the random oracle model for $H$. The intractability assumptions we will need to make are then as follows:

- The RSA assumption—it is hard to compute $y$ such that $y^e = x$, given random $x \in \mathbf{Z}_n^*$;
- The Decision Diffie-Hellman (DDH) assumption—given random $g, h \in Q_n$, along with $g^a$ and $h^b$ it is hard to decide if $a \equiv b \bmod m$.

We make our DDH assumption a bit more precise. For $h \in Q_n$, $a, b \in \mathbf{Z}_m$, and $c \in \{0, 1\}$, define

$$F(h, a, b, c) = \begin{cases} h^a & \text{if } c = 0; \\ h^b & \text{if } c = 1. \end{cases}$$

The DDH assumption states that for random $g \in Q_n$, and random $h, a, b, c$ as above, it is hard to compute—with negligible error probability—$c$ given $g, h, g^a, F(h, a, b, c)$.

Note that this is an average-case complexity assumption. It is equivalent to a worst-case complexity assumption, by a standard "random self reduction" argument [21, 16], provided the inputs are restricted in the following way: $g$ and $h$ should generate $Q_n$, and $\gcd(a - b, m) \notin \{p', q'\}$.

Note that the DDH is a reasonable assumption here, since the group $Q_n$ has no small prime factors [20].

By a standard "hybrid" argument (see [16]), the above DDH assumption is equivalent to the following: the distributions

$$(g, g^{a_1}, \ldots, g^{a_s};\ h, h^{a_1}, \ldots, h^{a_s})$$

and

$$(g, g^{a_1}, \ldots, g^{a_s};\ h, h^{b_1}, \ldots, h^{b_s})$$

are computationally indistinguishable. Here $s$ is any (small) number, $g$ and $h$ are random elements of $Q_n$, and the $a_i$'s and $b_i$'s are random numbers modulo $m$.

**Theorem 2** *In the random oracle model for $H$ and $H'$, under the RSA and DDH assumptions Protocol 2 is a secure threshold signature scheme (robust and non-forgeable) for $k \geq t+1$; moreover, when $k = t+1$, the same holds under the RSA assumption alone.*

The proof of the robustness property goes through as before. We focus here in the proof of non-forgeability.

The random oracle model for $H$ will allow the simulator to choose the outputs of $H$ as it wishes, so long as these outputs have the right distribution.

We now describe a series of simulators.

**The first simulator.** The simulator chooses the shares for the corrupted players $s_{i_1}, \ldots, s_{i_t}$ as random numbers chosen from $\{0, \ldots, \lfloor n/4 \rfloor - 1\}$, just as it did in the previous section.

Let $g, g_{i_{t+1}}, \ldots, g_{i_{k-1}}$ be random elements in $Q_n$. Here, $i_{t+1}, \ldots, i_{k-1}$ are arbitrary indices of uncorrupted players. We assume that all of these group elements are generators for $Q_n$, which is the case with all but negligible probability. The values $g, g_{i_{t+1}}, \ldots, g_{i_{k-1}}$ implicitly define $s_{i_{t+1}}, \ldots, s_{i_{k-1}}$ modulo $m$ by the equation $g_{i_j} = g^{s_{i_j}}$.

We next show how to sample from the distribution

$$\hat{x}, x_1, \ldots, x_l.$$

We choose $r \in \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ at random, and $b_1, b_2 \in \{0, 1\}$ at random. We set $\hat{x} = (g^r)^{\Delta^2 e} u^{-b_1 e} (-1)^{b_2}$, thus defining the corresponding value $x$ to be $(g^r)^{\Delta^2 e} (-1)^{b_2}$. For one of the uncorrupted players $i_j \in \{i_{t+1}, \ldots, i_{k-1}\}$, we have $x_{s_j} = (g_{i_j}^r)^{2\Delta^2 e}$. For other uncorrupted players $i$, we can compute $x_i$ as

$$x_i = (g^r)^{2(\lambda^S_{i,0} + \Delta e(\lambda^S_{i,i_1} s_{i_1} + \cdots + \lambda^S_{i,i_t} s_{i_t}))} (g^r_{i_{t+1}})^{2\Delta e \lambda^S_{i,i_{t+1}}} \cdot \cdots \cdot (g^r_{i_{k-1}})^{2\Delta e \lambda^S_{i,i_{k-1}}},$$

where $S = \{0, i_1, \ldots, i_{k-1}\}$. Again, this follows from (3).

We can generate values in this way so that $\hat{x}$ is the output of the random oracle $H$. We can also generate the verification keys $v, v_1, \ldots, v_l$ in basically the same way.

This simulator generates $\hat{x}$ in this way for *every* random oracle query, so we will not be able to break the RSA problem with this simulator (this is only the first step).

It is easy to see that this simulation is statistically close to perfect. The one thing to notice is that $\hat{x}$ is nearly uniformly distributed in $\mathbf{Z}_n^*$. The proof of this utilizes the fact that every element in $\mathbf{Z}_n^*$ can be expressed uniquely as $g^a u^{-eb_1} (-1)^{b_2}$, for $a \in \{0, \ldots, m-1\}$, and $b_1, b_2 \in \{0, 1\}$.

**The second simulator.** This simulator is the same as the first, except as follows. Let $g, g_{i_{t+1}}, \ldots, g_{i_{k-1}}$ and $h, h_{i_{t+1}}, \ldots, h_{i_{k-1}}$ be random elements in $Q_n$. This simulator "guesses" which message will be forged by the adversary; that is, we can assume that the forged message is an input to the random oracle, and the simulator just guesses one of these queries is the "target" message.

Everything is the same as before, except that when generating $\hat{x}, x_1, \ldots, x_l$ for the target message, the simulator performs the same calculations using the values $h, h_{i_{t+1}}, \ldots, h_{i_{k-1}}$ instead of $g^r, g^r_{i_{t+1}}, \ldots, g^r_{i_{k-1}}$ in the calculation.

This simulation is no longer statistically indistinguishable from from the real game, but this is where we use the DDH assumption. On this assumption, with non-negligible probability, the adversary will still forge a message, and that message will be the selected target.

Notice that the "correctness proofs" of the shares can be still be simulated using the random oracle model for $H'$ just as before—the fact that the statement being "proved" is false is interesting, but irrelevant.

**The third simulator.** This simulator is the same as the first, except as follows. Let $z$ be a random element in $\mathbf{Z}_n^*$. For the target message hash value, the simulator sets $\hat{x} = z$. Also, whenever the adversary asks for a signature share $x_i$ on the target message from any uncorrupted player, the adversary simply outputs a random quadratic residue. The "correctness proofs" can still be simulated, just as before. If the adversary ever asks for more than $k - t - 1$ signature shares on the target message, the simulator simply halts and reports an error.

It is easy to see that the distribution of this simulation is identical to that of the second simulation, provided the adversary does not ask for too many shares of the target message. Indeed, because of the way the second simulator constructs the signature shares $x_i$ from the uncorrupted players on the target message, any subset of $k - t - 1$ of them is uniformly distributed in $Q_n$, and independent of all other variables in the adversary's view. So with non-negligible probability, the adversary will forge a signature on the target message, which means, in particular, the he does not ask for too many shares. Moreover, if he forges this signature, then he has computed an $e$th root of $z$ in $\mathbf{Z}_n^*$, thus contradicting the RSA assumption.

To complete the proof of the theorem, we simply note that when $k = t+1$, the DDH is not needed at all in the above arguments.

## Acknowledgements

Thanks to Rosario Gennaro for suggesting improvements to a previous version of the paper.

## References

[1] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[2] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology–Crypto '92*, pages 89–105, 1992.

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

[4] Y. Desmedt. Society and group oriented cryptography: a new concept. In *Advances in Cryptology–Crypto '87*, pages 120–127, 1987.

[5] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology–Crypto '89*, pages 307–315, 1989.

[6] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In *Advances in Cryptology–Crypto '91*, pages 457–569, 1991.

[7] T. El Gamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[8] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.

[9] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology–Crypto '86, Springer LNCS 263*, pages 186–194, 1987.

[10] Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Technical Report TR-92-04-02, Univ. of Wisconsin–Milwaukee, 1992.

[11] Y. Frankel, P. Gemmall, P. MacKenzie, and M. Yung. Optimal-resilience proactive public-key cryptosystems. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[12] Y. Frankel, P. Gemmall, P. MacKenzie, and M. Yung. Proactive RSA. In *Advances in Cryptology–Crypto '97*, 1997.

[13] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *Advances in Cryptology–Crypto '96*, pages 157–172, 1996.

[14] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS. In *Advances in Cryptology–Eurocrypt '96*, pages 354–371, 1996.

[15] L. Harn. Group-oriented $(t, n)$ threshold digitial signature scheme and digital multisignature. *IEE Proc.-Comput. Digit. Tech.*, 141(5):307–313, 1994.

[16] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[17] T. Rabin. A simplified approach to threshold and proactive RSA. In *Advances in Cryptology–Crypto '98*, 1998.

[18] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, pages 120–126, 1978.

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

[20] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, 1997.

[21] M. Stadler. Publicly verifiable secrete sharing. In *Advances in Cryptology–Eurocrypt '96*, pages 190–199, 1996.