Threshold Cryptosystems Based on Factoring

Jonathan Katz[*] Moti Yung[†]

Abstract

We consider distributed (threshold) cryptosystems over a composite modulus N in which the *factors* of N are shared among the participants as the secret key. This is a new idea for threshold cryptosystems based on a composite (i.e., different from the typical treatment of the much-studied RSA-based threshold systems where a "decryption exponent" is shared among the participants). The paradigm enables solutions to open problems in threshold cryptography and it also yields substantial efficiency improvements when generation of N is done in a distributed manner (i.e., without a trusted dealer). In particular, our approach yields two new threshold schemes:

1. *Threshold homomorphic encryption.* We present a scheme for threshold decryption of the homomorphic Goldwasser-Micali (GM) encryption scheme [31] which has numerous applications and is based on the *hardness of deciding quadratic residuosity.* This answers in the positive an open question in [18].
2. *Threshold cryptosystems based on factoring.* We describe a threshold version of a standardized (ISO/IEC 9796) variant of Rabin signatures [36, Section 11.3.4], the first threshold signature scheme whose security (in the random oracle model) can be reduced [2] to the assumption that *factoring is hard.* The extension to Rabin decryption (whose semantic security may be reduced to factoring) is clear.

Efficient extensions to achieve robustness and proactive security are all possible with our schemes.

1 Introduction

Threshold cryptosystems provide for increased security and availability of a particular cryptographic protocol by distributing the protocol among a number of participants. In a k-out-of-ℓ threshold scheme, the protocol is distributed in such a way that an adversary who corrupts at most $k-1$ participants (and learns all their local information) gains no advantage in determining the secret key of the system or in breaking the underlying cryptographic protocol. On the other hand, increased availability is achieved by ensuring that only k participants are required in order to carry out the computation and deliver the result. Going further, systems can be designed in a *robust* manner, such that even a malicious adversary who causes up to $k-1$ ($k \leq \ell/2$) players to deviate arbitrarily from the protocol cannot prevent the correct output from being computed. Threshold schemes can also be *proactivized*

[*] jkatz@cs.columbia.edu. Department of Computer Science, Columbia University. Part of this work was done while the author was at Telcordia Technologies.

[†] moti@cs.columbia.edu. CertCo, Inc.

to withstand the compromise of even *all* participants over the lifetime of the protocol, as long as only $k-1$ participants are corrupted during each time period; they may also be extended to handle *adaptive* adversaries who decide which participants to corrupt at any point during execution of the protocol.

A long line of research has focused on threshold cryptography, with particular emphasis on threshold signature schemes (in many cases, deriving a threshold decryption scheme from a related signature scheme is easy). The approach was initiated by [14, 15, 16], and the first provably secure schemes for RSA and DSS and other discrete-log-based signatures were given in [13, 27, 32]. Subsequent work focused on adding robustness to existing schemes [21, 28, 29] and on threshold decryption schemes with security against chosen-ciphertext attacks [44, 7, 17].

The above protocols are all proven secure with respect to a non-adaptive adversary who must choose which participants to corrupt before protocol execution begins (this is the type of adversary we consider here). Many recent works have dealt with stronger classes of adversaries, including adaptive adversaries [1, 5] who may corrupt participants at any time during the protocol based on its entire history. Proactive systems [38] consider adversaries who may corrupt up to $k-1$ participants during any single time period. We refer the reader elsewhere for exhaustive references (e.g., [25, 33]).

The previously-mentioned protocols assume a trusted dealer who distributes keys to the participants before the protocol begins. The dealer must be trusted to operate correctly (in some cases participants can verify, to some extent, correct dealing); in any case, the dealer must be minimally trusted not to reveal the secret key, and therefore represents a single point of failure for the entire system. Thus, it is often desirable to distribute the key-generation phase of the protocol among the participants. This was first accomplished for discrete-log-based cryptosystems in [29, 6] (building on [40]), and for RSA-based cryptosystems in [3] (for passive adversaries) and [24] (for the case of active adversaries).

There is still a need to design threshold systems for many important *specific* cryptosystems and applications (note that most previous research on threshold cryptosystems was restricted to RSA- and discrete-log-based schemes and efficiency improvements thereof). In particular, as pointed out elsewhere [26, 18, 11, 34, 10], *homomorphic*, semantically-secure, threshold cryptosystems are useful for achieving such goals as (robust) voting and efficient multi-party computation. We note that a threshold scheme for the (homomorphic) Paillier cryptosystem [39] has been given previously [18, 11]. Yet, for some applications, homomorphism over $\mathbb{Z}_2$ is required or sufficient [26, 34, 10, 35] and using the Paillier cryptosystem may not work or may be "overkill". Clearly, additional approaches yielding threshold homomorphic encryption are needed (and this was left as an explicit open question in [18] and as a derived challenge in [10]).

Finally, even if one is satisfied with the existing threshold systems (e.g., RSA), much research remains to be done to increase the efficiency of existing solutions.[1] An example is [42], which shows how to achieve k-out-of-ℓ threshold computation of RSA signatures using the simple protocol of [20] (which allowed only ℓ-out-of-ℓ threshold computation). Note that a k-out-of-ℓ solution had already been achieved by [13], yet the later scheme of [42] is conceptually simpler. Another example is the recent proposal of [43], which gives a different

[1] In fact, this is the entire motivation for the area of threshold cryptography since, in a theoretical sense, "solutions" already exist based on general multi-party computation [30].

method for k-out-of-ℓ sharing of RSA signatures and also describes non-interactive proofs of correctness which make the protocol robust. Again, even though a previous solution existed [21], the protocol of [43] is substantially more practical (for N a product of strong primes). A final example is [8], which suggests a way to improve the efficiency (and round-complexity) of an important step in the distributed key-generation protocols of [3, 24]. For threshold cryptography to become truly practical, further efforts to improve the efficiency of existing solutions are needed.

1.1 Our Contributions

Threshold homomorphic encryption. We present a threshold decryption scheme for Goldwasser-Micali (GM) encryption based on the quadratic residuosity assumption [31]. It is well-known that this encryption scheme is homomorphic over $\mathbb{Z}_2$. Semantically-secure threshold homomorphic encryption schemes have many important applications. As an example, efficient multi-party computation can be based on any efficient threshold homomorphic encryption scheme [26, 10]. The scheme can be used for distributed crypto-counting and tallying in the electronic voting scheme of [34].

A variant threshold decryption for a GM-like cryptosystem has been constructed recently (concurrent with the present work) using an alternate approach [10]. However, the scheme of [10] (which builds on [26]) requires the DDH assumption in $\mathbb{Z}_N^*$, whereas the security of our construction relies only on the quadratic residuosity assumption (eliminating this assumption is left open in [10]). In addition, our solution offers a more efficient and conceptually simpler method. Finally, our scheme has the added advantage of allowing for efficient distributed key generation[2] (without a trusted dealer).

Threshold cryptosystems based on factoring. We are not aware of any previous constructions of threshold cryptosystems whose security can be reduced to the assumption that factoring is hard. Here, we propose a novel and efficient distributed version of a Rabin signature scheme variant [36, Section 11.3.4] (see also [41]) as secure as factoring in the random oracle model [2]. Extending the scheme to yield threshold decryption of the Rabin encryption scheme (whose semantic security can be based on factoring) is immediate.

Efficiency improvements. The protocols we present are additional examples of efficient and *practical* threshold schemes. When a trusted dealer cannot be assumed (and key generation must be done in a distributed fashion), our threshold schemes are more efficient than previous solutions which do not require a trusted dealer [12, 19]. The threshold schemes presented here may be easily executed in a modular manner following a "streamlined" version of the distributed key-generation protocols of [3, 24] (all parameters required for the present schemes are in place upon completion of these key-generation protocols, and we do not require that N be a product of safe primes); we may use a "streamlined" version of these protocols because *we do not require computation of an inverse over a shared (secret) modulus*. We therefore avoid *altogether* the very step which a recent paper [8] improves!

Finally, we believe the methods outlined in this paper are interesting in their own right; in some sense, the distribution of the primes themselves is a new paradigm for threshold cryptography over composite moduli, and may prove useful in the design of future schemes.

[2] The scheme of [10] requires N to be a product of strong primes, a restriction we do not impose here.

2 Model and Definitions

2.1 The Model

Participants. The participants are ℓ servers $\{P_1, \ldots, P_\ell\}$ and a trusted dealer D. We note that, for some of our schemes, the trusted dealer is merely a notational convenience since a distributed algorithm can be run in place of the dealer. When this is the case, we will explicitly mention it. The dealer D generates a public key N for the underlying cryptosystem and distributes shares to each of the participants. After the dealing phase, the dealer does not take part in execution of the protocol. Following [27], we assume the participants are connected by a complete network of private channels. In addition, all players have access to an authenticated broadcast channel (i.e., the true sender of a message can always be correctly determined). These assumptions allow us to focus on high-level descriptions of the protocol; however, they may be instantiated using standard cryptographic techniques (in the proactive setting, care needs to be taken; see [38, 32]).

Time. When we discuss proactivation of our schemes, we view time as divided into disjoint periods which are determined by a common global clock. Each period consists of an initial refresh period during which shares of players are refreshed and/or reconstructed (in the case of participants controlled by the adversary in the previous time period). After the refresh period, participants may generate signatures on messages they are given as input.

The Adversary. Our k-out-of-ℓ schemes assume a non-adaptive adversary who may corrupt up to $k-1$ participants in each time period (in the proactive setting, a player is considered corrupted during a time period if he was corrupted during that time period or the preceeding refresh period). The adversary has access to all information available to the corrupted players, including their secret keys, messages they receive, and messages broadcast to all players. Additionally (in the case of threshold signatures), the adversary may submit signing requests to the system at any time. One may consider two types of adversaries: *passive* adversaries who follow the protocol faithfully yet monitor all information available to corrupted participants, and *active* adversaries who may cause participants to deviate arbitrarily from the protocol. We note that it is possible to modify our protocols to accommodate an adaptive adversary, but we defer a detailed discussion of this point from the present abstract.

2.2 Security

Formal definitions of security for threshold cryptosystems have appeared elsewhere [28]. We describe, informally, our requirements. First, we want the security of the threshold scheme to be equivalent to the security of the original scheme, even when an adversary has corrupted $k-1$ servers and obtained all their local information. To prove that this requirement is met, we reduce the security of the threshold scheme to that of the original scheme by showing how an adversary attacking the original scheme can simulate the view of (up to) $k-1$ servers in the threshold scheme. Following [28], we call such threshold protocols *simulatable*. An additional requirement we will consider is *robustness*: for any active adversary who causes at most $k-1$ ($k \leq \ell/2$) participants to deviate arbitrarily from the protocol, the correct result can always be computed by the remaining (uncorrupted) participants.

3 Threshold Homomorphic Encryption

We begin by describing our construction of a threshold decryption scheme for the well-known homomorphic encryption scheme [31] based on quadratic residues (henceforth, GM). The GM encryption scheme is as follows: the public key is a composite $N = pq$, where p and q are primes and $p = q = 3 \bmod 4$. The private key consists of the factorization of N. To encrypt bit $b \in \{0, 1\}$, choose a random element $r \in \mathbb{Z}_N$ and send $C = (-1)^b r^2 \bmod N$. Decryption of ciphertext C proceeds by determining whether C is a quadratic residue or not. First, calculate the Jacobi symbol $J = (\frac{C}{N})$. If $J \neq 1$, then the ciphertext is ill-formed (i.e., the encryption algorithm was not run honestly, or else the message was corrupted in transmission); therefore, simply output $\perp$. If $J = 1$, we may decide whether C is a quadratic residue by computing $b' = C^{(N-p-q+1)/4} \bmod N$; note that C is a quadratic residue iff $b' = 1$. At this point, the original plaintext can be recovered by computing $b = (1 - b')/2$. This scheme is semantically secure under the quadratic residuosity assumption [31].

3.1 The Basic ℓ-out-of-ℓ Solution

For simplicity, in this section we concentrate on describing a protocol (which we further build on) for basic threshold GM decryption (cf. Figure 1). The basic protocol assumes a trusted dealer and is an ℓ-out-of-ℓ solution. Thus, all ℓ participants are needed in order to decrypt a ciphertext; on the other hand, it remains infeasible for any adversary who corrupts $\ell - 1$ or fewer participants to decrypt a given ciphertext. We present this simple solution first for clarity of exposition. In the following section, we discuss extensions and modifications which allow for the more general k-out-of-ℓ threshold, provide robustness, and enable proactivation of the protocol. Additionally, we show how to remove the trusted dealer and perform the initial key generation and share distribution in a distributed manner.

Key Distribution. The dealer generates primes $p, q = 3 \bmod 4$ (where $|p| = |q| = n$) and sets $N = pq$. The public key is N, and the private key is computed as $d = (N-p-q+1)/4$. For all i, the dealer chooses $p_i, q_i \in_R (0, 2^{2n})$ such that $p_i = q_i = 0 \bmod 4$. Finally, the dealer sets $p_0 = p - \sum_{i=1}^{\ell} p_i$ and $q_0 = q - \sum_{i=1}^{\ell} q_i$. The dealer sends (p_i, q_i) to player i, and broadcasts (N, p_0, q_0).

Decryption. Decryption of a ciphertext C proceeds as follows: first, the Jacobi symbol $J = (\frac{C}{N})$ is computed (note that this can be computed in polynomial time even without knowledge of the factorization of N). If $J \neq 1$, all players simply output $\perp$. Otherwise, player i outputs $b_i = C^{(-p_i-q_i)/4} \bmod N$ (note that, by design, the exponents can all be computed over the integers). Players publicly compute $b_0 = C^{(N-p_0-q_0+1)/4} \bmod N$ (again, by design, this exponent may be computed over the integers). Deciding whether C is a quadratic residue may be done by computing $b' = \Pi_{i=0}^{\ell} b_i \bmod N$. The decrypted bit is simply $b = \frac{1-b'}{2}$. Security of the scheme is captured by the following theorem:

Dealing Phase

Input: Composite N and primes p, q ($|p| = |q| = n$) such that $N = pq$
with $p, q = 3 \bmod 4$

1. Choose $p_1, q_1, \ldots, p_\ell, q_\ell \in_R (0, 2^{2n})$ such that $p_i = q_i = 0 \bmod 4$, for all i
2. Set $p_0 = p - \sum_{i=1}^{\ell} p_i$ and $q_0 = q - \sum_{i=1}^{\ell}$
3. Send (p_i, q_i) to player i
4. Broadcast (N, p_0, q_0)

Decryption Phase

Input: Ciphertext C

1. All players compute $J = (\frac{C}{N})$ (this computation may be done publicly, so all players agree on the value)
2. If $J \neq 1$, all players output $\perp$ and stop
3. Otherwise ($J = 1$), player i broadcasts $b_i = C^{(-p_i - q_i)/4} \bmod N$
4. All players publicly compute $b_0 = C^{(N - p_0 - q_0 + 1)/4} \bmod N$
5. The decrypted bit b is computed as $b = \left(1 - \Pi_{i=0}^{\ell} b_i \bmod N\right)/2$

Figure 1: The ℓ-out-of-ℓ protocol for decryption

Theorem 1 *The protocol of Figure 1 is simulatable for any adversary who passively eavesdrops on at most $\ell - 1$ parties. This implies the semantic security of the encryption scheme for such an adversary, assuming the hardness of deciding quadratic residuosity.*

The proof is similar to the (more involved) proof of security for the Rabin signature scheme given below (cf. Theorem 4), and is therefore omitted.

3.2 Extended Protocols and Applications

Reducing the Threshold. It is a severe limitation to require ℓ active servers in order to decrypt. More preferable is a k-out-of-ℓ solution in which only k servers are required for decryption. A number of techniques exist for accomplishing this using the above protocol as a starting point; we sketch two such solutions here.

The first approach follows the suggestions of Rabin [42] for the case of threshold RSA. First, the dealer fixes a prime $P > 2^{2n}$ which is broadcast to all participants. Then, for each p_i (and also q_i), the dealer chooses a random $(k-1)$-degree polynomial $f_i(\cdot)$ over field $\mathbb{Z}_P$ such that $f_i(0) = p_i$. To player j, the dealer sends $f_i(j)$ for $1 \leq i \leq \ell$. This achieves a k-out-of-ℓ secret sharing of the $\{p_i\}$ (and also the $\{q_i\}$). Decryption proceeds as before, with each player i broadcasting its share b_i of the decryption. In addition, players prove correctness of their shares using one of the robustness techniques described below. If player i cannot prove correctness of his share (or, more generally, if player i fails to participate), the remaining players can publicly reconstruct (p_i, q_i) using the shares they have been given. The correct share b_i may then be computed publicly and included in the calculation of b. We note that, in case a trusted dealer is not available, each player may itself deal shares of (p_i, q_i) to the other players. If robustness is desired for this step, verifiable secret sharing (VSS) may be used. Details appear in [42].

A problem with this approach is that it may unfairly penalize servers which are temporarily off-line. In other words, if player i is momentarily disconnected and cannot participate in an execution of the signing protocol, his share is publicly reconstructed (and hence available to an adversary eavesdropping on the protocol). Note that it may be much easier in practice for an adversary to disconnect or prevent communication from players than for an adversary to corrupt players (even passively).

An alternative is to use ideas motivated by the sum-to-poly and poly-to-sum protocols introduced in [23]. Here, we begin with an ℓ-out-of-ℓ additive sharing as in Figure 1. Let $L = \ell!$. Player i chooses a random $(k-1)$-degree polynomial f_i over the *integers*, with coefficients chosen uniformly from $\{0, L, \ldots, L^3 2^{3n} k\}$, such that $f_i(0) = L^2 p_i$ (the process is repeated for q_i as well). Player i sends $f_i(j)$ to player j, for $1 \leq j \leq \ell$. Player i then sets his share to $p_i^* = \sum_{j=1}^{\ell} f_j(i)$ (the original shares $\{(p_i, q_i)\}$ may be erased). The p_i^* are now a k-out-of-ℓ polynomial sharing of p. To decrypt, a random set Λ of k players is chosen. Each player in this set computes the appropriate Lagrange interpolation coefficient $z_{i,\Lambda}$ and sets his (temporary) share to $\hat{p}_i = z_{i,\Lambda} \cdot p_i^*$. The $\hat{p}_i$ may be computed over the integers, due to the special form of the polynomials $\{f_i\}$ used to generate shares. Note that the $\hat{p}_i$ constitute a k-out-of-k *additive* sharing of p. The participants in Λ may now perform k-out-of-k decryption, using these additive shares, following the paradigm of Figure 1. We note that [23] additionally gives techniques to achieve robustness for the above approach.

The preceeding two approaches may be viewed as "generic" approaches which convert *any* ℓ-out-of-ℓ scheme to a k-out-of-ℓ scheme. Of course, the details must be verified (as we have done here), and each approach needs to be appropriately modified for the particular setting. The preceeding discussion, along with Theorem 1 and the results cited above, gives the following theorem:

Theorem 2 *The protocol of Figure 1 augmented with either of the approaches described above gives a k-out-of-ℓ protocol which is simulatable for any adversary who passively eavesdrops on at most $k-1$ parties.*

Robustness. We may distinguish two methods for adding robustness to the above protocol: methods which work when N is a product of strong primes[3], and methods which work for general N. Methods specialized for the former case are generally more efficient; on the other hand, when distributed key generation is required, methods which work in the general case *must* be used (since there are currently no known efficient, distributed protocols to generate N as a product of strong primes).

The work of [28] gives two methods for verifying correctness of the partial outputs b_i when N is a product of strong primes. One method, which is completely non-interactive, requires the dealer to distribute verification information to all players during the dealing phase; $V_{i,j}$ is sent to player i to enable his verification of player j. When executing the

[3] That is, $N = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$, where p, q, p', q' are all prime.

protocol, player i outputs s_i and also $b_{i,j}$ for all j; player j verifies the correctness of b_i using $V_{j,i}$ and $b_{i,j}$. This requires $O(\ell \cdot n)$ memory for each player, and also increases the communication of the protocol (per player) to $O(\ell \cdot n)$.

A second approach of [28] requires the dealer to choose a random element (of high order) $g \in \mathbb{Z}_N^*$ and broadcast g along with *witnesses* $w_i = g^{(-p_i - q_i)/4} \bmod N$, for all i. After player i broadcasts b_i, he engages in an interactive, zero-knowledge proof with all other players in which he proves that the discrete logarithms of w_i with respect to g and b_i with respect to C are equal. Unfortunately, this requires interaction; it does not seem possible to eliminate the interaction even using a random oracle. More recently, Shoup [43] (based on earlier work of [9]) describes a non-interactive, zero-knowledge proof (using a random oracle) of equality of discrete logarithms. Here, one must work over the subgroup of quadratic residues $Q_N \subset \mathbb{Z}_N^*$ and thus the dealer chooses $g \in Q_N$; furthermore, player i proves equality of the discrete logarithms of w_i with respect to g and b_i^2 with respect to C^2 (squaring is necessary to ensure that we are working in Q_N [43]).

The above approaches suffice for N a product of strong primes. For general N, however, we must use other techniques to achieve robustness[4]. One possibility is to use the cryptographic program-checking method of [21], which requires interaction between each pair of parties (this interaction can be reduced to only two rounds using a random oracle). Another approach extends the witness-based approach above. Using a random oracle, players may, as above, give an efficient, non-interactive, zero-knowledge proof [9] that $\log_g w_i = \log_C b_i$. A difficulty here is that soundness is only guaranteed if g is of high order; however, as shown in [24], a set (of super-logarithmic size) of random elements of $\mathbb{Z}_N^*$ generates a large-order subgroup of $\mathbb{Z}_N^*$ with all but negligible probability. Fixing such a set as part of the dealing phase and having players give a non-interactive proof with respect to each element in this set is thus sufficient to guarantee soundness. A third method for achieving robustness when special prime composites (which may be generated is a distributed manner) are used, is that of [19] as was pointed out to us by P.-A. Fouque. The above approaches to proving correctness of exponentiation modulo N allow proofs of correctness for the partial shares b_i broadcast by each player in the protocol. Theorems 1 and 2, together with the results cited above, yield the following theorem:

Theorem 3 *The protocol of Figure 1 augmented with any of the robustness techniques described above (appropriate for the modulus N) and any of the approaches for achieving a k-out-of-ℓ ($k \leq \ell/2$) scheme (as described in Theorem 2) results in a robust protocol which is simulatable for any adversary who actively controls at most $k-1$ parties.*

Removing the Trusted Dealer. The efficiency improvement of the current protocol is most clear when a trusted dealer is not assumed, and the public modulus must be generated in a distributed fashion. In this case, our scheme has two advantages: (1) moduli of a special form (i.e., N a product of strong primes) are not required, in contrast with some recent solutions (e.g., [43]); note that currently-known, distributed key-generation protocols [3, 24] cannot be used to generate such N. Furthermore, (2) an expensive step of the distributed

[4] Note that systems using general moduli N will generally do so because distributed key generation is required (and there are no currently-known, efficient protocols for distributed generation of N a product of strong primes). Although we still refer to a dealer, the robustness techniques described here can all be implemented quite easily following the robust, distributed, key-generation protocol of [24].

key-generation protocol can be skipped entirely. Specifically, computation of an inverse[5] over $\varphi(N)$ (recall that $\varphi(N)$ must remain hidden from the players) is not required for the current scheme.

The protocol of Figure 1 may be combined modularly with the distributed key-generation protocols of [3, 24]. Following execution of these key-generation protocols, all the players *already* have additive shares (p_i, q_i) of the factors of N. One point requiring care is that, as described above, the protocol requires all players to have $p_i = q_i = 0 \bmod 4$. To deal with this[6], simply have player i choose $p_i = q_i = 0 \bmod 4$. Additionally, the "public remainder" is set to $(p_0, q_0) = (0, 0)$. Decryption is then done as before.

Proactive Security. Proactive security may be added to our protocol using known techniques. For example, if the approach of [42] is used to achieve k-out-of-ℓ threshold, the generic proactivation techniques given there will work here as well. Similarly, if the approach of [23] is used, the proactivation techniques given there will also work for the present protocol. Due to space limitations, we refrain from a detailed description of these techniques.

Applications. Various protocols employ the GM scheme since it possesses a unique xor-homomorphic property. Let us review a few possible applications. The efficient multi-party computation scheme of [10] can be based on our threshold scheme, and the efficient voting scheme of [34] can now enjoy the important property of distributed tallying authorities. The private information retrieval (PIR) protocol in [35], which employs in a crucial fashion the GM-cryptosystem, is an example of a protocol into which our scheme may be integrated seamlessly to enable distribution of the information receiver. This may be useful in certain applications where the receiver is controlled by a quorum agreement which is managed internally.

Chosen Ciphertext Security. A generic method for making threshold cryptosystems secure against chosen-ciphertext attack was recently described [17], adapting the method of Naor-Yung [37] to efficient schemes using random-oracle-based proofs. What is required are two schemes (both based on threshold-GM above) and an honest-verifier ZK proof of knowledge that two encryptions are of the same plaintext. Such a scheme is presented in Appendix A; while the protocol given there is a bitwise scheme, it can be run in parallel (and is hence very efficient) in the random-oracle model.

4 Threshold Signatures Based on Factoring

Distributing the prime factors of the modulus among the participants offers (in some sense) a new paradigm for the construction of threshold systems over composite moduli. An example of the applicability of this technique is the following method for threshold signatures based on a variant of the Rabin signature scheme. The scheme is particularly interesting since it offers the first threshold signature scheme whose security can be based on factoring (in the random oracle model, when appropriate hashing is employed).

[5] Note that this is precisely the step that [8] show how to perform more efficiently than the original solution of [3]. Here, we avoid the step altogether!

[6] A similar approach was noted in [3], where they require $p = q = 3 \bmod 4$.

4.1 The Modified-Rabin Signature Scheme

The Rabin signature scheme works as follows (we use the modified-Rabin scheme where N is a Williams integer as presented in [36, Section 11.3.4]): a public key is generated by choosing two primes p, q of length n, such that $p = 3 \bmod 8$ and $q = 7 \bmod 8$. The public key is set to $N = pq$. The private key is $d = (N - p - q + 5)/8$.

Messages m to be signed are assumed to be appropriately encoded (i.e., a hash of the original message such that the results of [2] apply) and the resulting underlying message space is $\mathcal{M} = \{m : m = 6 \bmod 16\}$ (which the randomized hashing and signing method in [2] can incorporate). First, the Jacobi symbol $J = (\frac{m}{N})$ is computed. If $J = 1$, set $\tilde{m} = m$; if $J = -1$, set $\tilde{m} = m/2$ (note that there is only negligible probability that $J \neq 1, -1$). The signature is computed as $s = \tilde{m}^d \bmod N$.

To verify signature s on message m (where $m = 6 \bmod 16$), first compute $\tilde{m} = s^2 \bmod N$. Then, verify the following:

- If $\tilde{m} = 6 \bmod 8$, verify whether $m \overset{?}{=} \tilde{m}$
- If $\tilde{m} = 3 \bmod 8$, verify whether $m \overset{?}{=} 2\tilde{m}$
- If $\tilde{m} = 7 \bmod 8$, verify whether $m \overset{?}{=} N - \tilde{m}$
- If $\tilde{m} = 2 \bmod 8$, verify whether $m \overset{?}{=} 2(N - \tilde{m})$

See [36, Section 11.3.4] for proof of correctness and further discussion.

4.2 The Protocol

As above, we present the ℓ-out-of-ℓ solution here for simplicity (cf. Figure 2); extensions as discussed in Section 3.2 are applicable here as well.

Key Distribution. The dealer generates primes p, q (where $|p| = |q| = n$, $p = 3 \bmod 8$, and $q = 7 \bmod 8$) and sets $N = pq$. The public key of the protocol is N, and the private key (see Section 4.1) is $d = (N - p - q + 5)/8$. For $i = 1, \ldots, \ell$, the dealer then chooses $p'_i, q'_i \in_R (0, 2^{2n-3})$ and computes $p_i = 8p'_i$ and $q_i = 8q'_i$ (in this way, $p_i = q_i = 0 \bmod 8$). The dealer sets $p_0 = p - \sum_{i=1}^{\ell} p_i$ and $q_0 = q - \sum_{i=1}^{\ell} p_i$. Finally, the dealer sends (p_i, q_i) to player i, and broadcasts (p_0, q_0).

Signature Generation. We assume the message $m \in \mathcal{M}$ to be signed is already encoded in some appropriate agreed-upon manner (i.e., such that the results of [2] apply and the scheme is unforgeable in the random oracle model). First, the Jacobi symbol $J = (\frac{m}{N})$ is computed publicly (note that the Jacobi symbol can be computed in polynomial time even without knowledge of the factorization of N). If $J = 1$, define $\tilde{m} = m$; if $J = -1$, define $\tilde{m} = m/2$; this step may be done publicly as well.

The desired signature is $s = \tilde{m}^d = \tilde{m}^{(N-p-q+5)/8} \bmod N$. Player i broadcasts the value $s_i = \tilde{m}^{(-p_i-q_i)/8} \bmod N$ (note that, by construction of the shares, the exponent can be computed over the integers). Players publicly compute $s_0 = \tilde{m}^{(N-p_0-q_0+5)/8} \bmod N$ (again, by construction of (p_0, q_0) the exponent can be computed over the integers). Finally, the

Dealing Phase
Input: Composite N and primes p, q ($|p| = |q| = n$) such that $N = pq$
with $p = 3 \bmod 8$ and $q = 7 \bmod 8$

1. Choose $p_1, q_1, \ldots, p_\ell, q_\ell \in_R (0, 2^{2n})$ such that $p_i = q_i = 0 \bmod 8$, for all i
2. Set $p_0 = p - \sum_{i=1}^{\ell} p_i$ and $q_0 = q - \sum_{i=1}^{\ell} q_i$
3. Send (p_i, q_i) to player i
4. Broadcast (N, p_0, q_0)

Signature Generation Phase
Input: Message $m = 6 \bmod 16$ (appropriately encoded)

1. Player i computes $J = (\frac{m}{N})$ (this computation may be done publicly, so all players agree on the value)
2. If $J = 1$, set $\tilde{m} = m$; else set $\tilde{m} = m/2$
3. Player i broadcasts $s_i = \tilde{m}^{(-p_i - q_i)/8} \bmod N$
4. All players publicly compute $s_0 = \tilde{m}^{(N - p_0 - q_0 + 5)/8} \bmod N$
5. The signature s is computed as $s = \Pi_{i=0}^{\ell} s_i \bmod N$

Figure 2: The ℓ-out-of-ℓ protocol for Rabin signatures

signature is computed as $s = \Pi_{i=0}^{\ell} s_i \bmod N$. Verification of the signature is exactly the same as described in Section 4.1.

The security of the protocol is given by the following theorem:

Theorem 4 *The protocol of Figure 2 is simulatable for any adversary who passively eavesdrops on at most $\ell - 1$ parties. This implies the unforgeability (in the random oracle model) of the signature scheme for such an adversary, assuming the hardness of factoring.*

Proof A description of a simulator for the dealing phase and the signature generation phase appears in Figure 3. We assume (without loss of generality) that the adversary eavesdrops on players $1, \ldots, \ell - 1$. Simulatability of the dealing phase is evident from the following:

- The $\{p_i, q_i\}_{1 \leq i \leq \ell - 1}$ have the same distribution as in a real execution of the protocol.
- The distribution on (p_0, q_0), conditioned on the values of $\{p_i, q_i\}_{1 \leq i \leq \ell - 1}$ seen by the adversary, is statistically indistinguishable from the distribution on (p_0, q_0) in a real execution of the protocol. This is because, for *any* $p, p^* < 2^{n+1}$, the distributions $\{p - p_1\}_{p_1 \in_R (0, 2^{2n})}$ and $\{p^* - p_1\}_{p_1 \in_R (0, 2^{2n})}$ are statistically indistinguishable.

Simulatability of the signature generation phase derives from the following:

Simulation of Dealing Phase
Input: Composite N where $|N| = 2n$

1. Choose $p_1, q_1, \ldots, p_\ell, q_\ell \in_R (0, 2^{2n})$ such that $p_i = q_i = 0 \bmod 8$
2. Choose random p^*, q^* such that $|p^*| = |q^*| = n$, $p^* = 3 \bmod 8$, and $q^* = 7 \bmod 8$
3. Set $p_0 = p^* - \sum_{i=1}^{\ell} p_i$ and $q_0 = q^* - \sum_{i=1}^{\ell} q_i$
4. Send (p_i, q_i) to player i, for $1 \leq i \leq \ell - 1$
5. Broadcast (p_0, q_0)

Simulation of Player ℓ in Signature Generation Phase
Input: Message $m = 6 \bmod 16$ (appropriately encoded); signature s

1. Compute $J = (\frac{m}{N})$
2. If $J = 1$, set $\tilde{m} = m$; else set $\tilde{m} = m/2$
3. Compute $s_i = \tilde{m}^{(-p_i - q_i)/8} \bmod N$, for $1 \leq i \leq \ell - 1$
4. Compute $s_0 = \tilde{m}^{(N - p_0 - q_0 + 5)/8} \bmod N$
5. Broadcast $s_\ell = s / \left(\Pi_{i=0}^{\ell-1} s_i \right) \bmod N$

Figure 3: Simulator for ℓ-out-of-ℓ threshold Rabin signature scheme

- The distribution on s_ℓ is indistinguishable from its distribution in a real execution of the protocol. This is easily argued based on the indistinguishability of the $\{p_i, q_i\}_{0 \leq i \leq \ell - 1}$.

This concludes the proof of the theorem. ■

Efficient extensions to achieve optimal threshold, robustness, proactivation, and distributed key generation are all possible as outlined in Section 3.2. Also, the above method extends to give threshold decryption of the Rabin scheme, whose semantic security may be based on factoring.

References

[1] D. Beaver and S. Haber. Cryptographic Protocols Provably Secure Against Dynamic Adversaries. Eurocrypt '92.

[2] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures — How to Sign with RSA and Rabin. Eurocrypt '96.

[3] D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. Crypto '97.

[4] C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, eds., *Cryptography and Coding*, Clarendon Press, 1989.

[5] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively Secure Multi-Party Computation. STOC '96.

[6] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive Security for Threshold Cryptosystems. Crypto '99.

[7] R. Canetti and S. Goldwasser. An Efficient Threshold Public-Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. Eurocrypt '99.

[8] D. Catalano, R. Gennaro, and S. Halevi. Computing Inverses over a Shared Secret Modulus. Eurocrypt '00.

[9] D. Chaum and T. Pedersen. Wallet Databases and Observers. Crypto '92.

[10] R. Cramer, I. Damgård, and J.B. Nielson. Multiparty Computation from Threshold Homomorphic Encryption. Eurocrypt 2001.

[11] I. Damgård and M. Jurik. A Generalisation, A Simplification, and Some Applications of Paillier's Probabilistic Public-Key System. PKC'01.

[12] I. Damgård and M. Koprowski. Practical Threshold RSA Signatures without a Trusted Dealer. Eurocrypt 2001.

[13] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. STOC '94.

[14] Y. Desmedt. Society and Group-Oriented Cryptography: A New Concept. Crypto '87.

[15] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto '89.

[16] Y. Desmedt and Y. Frankel. Shared Generation of Authenticators and Signatures. Crypto '91.

[17] P.-A. Fouque, and D. Pointcheval, Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks, Asiacrypt'01.

[18] P.-A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. Financial Cryptography, 2000.

[19] P.-A. Fouque and J. Stern. Fully Distributed Threshold RSA under Standard Assumptions. Asiacrypt'01.

[20] Y. Frankel. A Practical Protocol for Large Group-Oriented Networks. Eurocrypt '89.

[21] Y. Frankel, P. Gemmell, and M. Yung. Witness-Based Cryptographic Program Checking and Robust Function Sharing. STOC '96.

[22] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. Crypto '97.

[23] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptography. FOCS '97.

[24] Y. Frankel, P. MacKenzie, and M. Yung. Robust Efficient Distributed RSA Key Generation. STOC '98.

[25] Y. Frankel, P. MacKenzie, and M. Yung. Adaptively Secure Threshold Cryptosystems. European Symposium on Algorithms '99.

[26] M. Franklin and S. Haber. Joint Encryption and Message-Efficient Secure Computation. J. of Cryptology 9(4): 217–232 (1996).

[27] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. Eurocrypt '96.

[28] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. J. of Cryptology 13(2): 273–300 (2000).

[29] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log-Based Cryptosystems. Eurocrypt '99.

[30] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game. STOC '87.

[31] S. Goldwasser and S. Micali. Probabilistic Encryption. JCSS 28(2): 270–299 (1984).

[32] A. Herzberg, M. Jakobsson, S, Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Systems. CCCS '97.

[33] S. Jarecki and A. Lysyanskaya, Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. Eurocrypt '00.

[34] J. Katz, S. Myers, and R. Ostrovsky. Cryptographic Counters and Applications to Electronic Voting. Eurocrypt 2001.

[35] E. Kushilevitz and R. Ostrovsky. Replication is not Needed: Single Database Computationally-Private Information Retrieval. FOCS 97.

[36] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1999.

[37] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. STOC 90.

[38] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC '91.

[39] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt '99.

[40] T. P. Pedersen. A Threshold Cryptosystem Without a Trusted Party. Eurocrypt '91.

[41] M. O. Rabin. Digital Signatures and Public Key Functions as Intractable as Factoring. Technical Memo TM-212, Lab. for Computer Science, MIT, 1979.

[42] T. Rabin. A Simplified Approach to Threshold and Proactive RSA. Crypto '98.

[43] V. Shoup. Practical Threshold Signatures. Eurocrypt '00.

[44] V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Against Chosen Ciphertext Attack. Eurocrypt '98.

A ZK Proof of Equality for GM-Ciphertexts

The following is the protocol:

Input: Two Blum-integer composites N_1, N_2 and the input pair which is a twin encryption of the same bit b:
$\{X_1 = -1^b x_1^2 \bmod N_1, X_2 = -1^b x_2^2 \bmod N_2\}$ where $x_j \in Z^*_{N_j}, j = 1, 2$.

Repeat k times:

1. Prover chooses a bit $c \in \{0, 1\}$ at random, and "twin encrypts" it at random: $\{V_1 = -1^c v_1^2 \bmod N_1, V_2 = -1^c v_2^2 \bmod N_2\}$ for random $v_j \in Z^*_{N_j}$.
2. Prover sends: V_1, V_2.
3. Verifier chooses a challenge bit $d \in \{0, 1\}$ at random, and sends d.
4. Prover respond to the challenge by sending: $\{m_1 = v_1 \cdot [x_1]^d \bmod N_1, m_2 = v_2 \cdot [x_2]^d \bmod N_2\}$
5. Verifier checks that: there exists a bit a such that both: $\{m_1^2 = -1^a \cdot V_1 \cdot [X_1]^d \bmod N_1, m_2^2 = -1^a \cdot V_2 \cdot [X_2]^d \bmod N_2\}$

Only if the checks done in all iterations are positive the verifier accepts.

Figure 4: Proof of Knowledge of twin GM-encryption

Note that the verifier assures that either the twin encryptions are of the same bit (when $d = 0$), or that the plaintext of the twin encryption input, when XORed with the plaintext of the twin encryption V_1, V_2, yields the same result, namely the bit a.

The above proof system which is complete and sound; furthermore, it is a proof of knowledge where the extractor attempts to rewind the prover and extract the bit b and the random bits used in the input twin encryption. Zero-knowledgeness of the protocol follows via standard simulation techniques; details are omitted here. To turn this to a non-interactive random-oracle-based proof of knowledge, the Fiat-Shamir technique is employed.