Threshold Cryptosystems Based on Factoring

Jonathan Katz[1,3] and Moti Yung[2]

[1] Department of Computer Science, University of Maryland (College Park)
jkatz@cs.umd.edu
[2] Department of Computer Science, Columbia University
moti@cs.columbia.edu
[3] Work done while at Columbia University and Telcordia Technologies

Abstract. We consider threshold cryptosystems over a composite modulus N where the *factors* of N are shared among the participants as the secret key. This is a new paradigm for threshold cryptosystems based on a composite modulus, differing from the typical treatment of RSA-based systems where a "decryption exponent" is shared among the participants. Our approach yields solutions to some open problems in threshold cryptography; in particular, we obtain the following:

1. *Threshold homomorphic encryption.* A number of applications (e.g., electronic voting or efficient multi-party computation) require threshold homomorphic encryption schemes. We present a protocol for threshold decryption of the homomorphic Goldwasser-Micali encryption scheme [34], answering an open question of [21].
2. *Threshold cryptosystems as secure as factoring.* We describe a threshold version of a variant of the signature standards ISO 9796-2 and PKCS#1 v1.5 (cf. [39, Section 11.3.4]), thus giving the first threshold signature scheme whose security (in the random oracle model) is equivalent to the hardness of factoring [12]. Our techniques may be adapted to distribute the Rabin encryption scheme [44] whose semantic security may be reduced to the hardness of factoring.
3. *Efficient threshold schemes without a trusted dealer.* Because our schemes only require sharing of N — which furthermore need not be a product of strong primes — our schemes are very efficient (compared to previous schemes) when a trusted dealer is not assumed and key generation is done in a distributed manner.

Extensions to achieve robustness and proactivation are also 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) still cannot determine the secret key of the system or break the underlying cryptographic protocol. On the other hand, increased availability is achieved by ensuring that only k participants are needed 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* 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 whom 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 [17–19], and the first provably secure schemes for RSA- and discrete-logarithm-based signature schemes were given in [16, 30, 35]. Subsequent work focused on adding robustness to existing schemes [24, 31, 32] and on threshold decryption schemes with security against chosen-ciphertext attacks [47, 9, 20].

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 [2, 7] and proactive [41] adversaries. We refer the reader elsewhere for more comprehensive surveys of the existing literature (e.g., [28, 36]).

The protocols mentioned above assume a dealer who distributes keys to the participants before the protocol begins. 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 among the participants. This was first accomplished for discrete-logarithm-based cryptosystems in [32, 8] (building on [43]), and for RSA-based cryptosystems in [5] (for passive adversaries) and [27] (for active adversaries).

There is still a need to design threshold schemes for many *specific* cryptosystems (most previous research on threshold cryptography was restricted to RSA- and discrete-logarithm-based schemes). First, note that for threshold cryptography to become truly practical, it remains important to improve the efficiency and conceptual simplicity of existing solutions.[1] Furthermore, as pointed out many times previously [29, 21, 14, 37, 13], threshold *homomorphic* encryption schemes are useful for achieving such goals as electronic voting and efficient multi-party computation. Threshold schemes have been given previously [43, 21, 14] for the El Gamal (which is homomorphic under group multiplication) and Paillier [42] (which is homomorphic under addition) cryptosystems. Yet, for some applications, homomorphism over, e.g., $\mathbb{Z}_2$ is required or sufficient [29, 38, 37, 13] and hence other homomorphic schemes may not work or may be "overkill" for the problem at hand. Clearly, additional approaches yielding threshold homomor-

[1] This is the motivation for the study of threshold cryptography since, in a theoretical sense, "solutions" already exist using generic multi-party computation [33].

phic encryption are needed (and this was left as an explicit open question in [21]; see also [13]).

1.1 Our Contributions

Threshold homomorphic encryption. We show how to achieve threshold decryption for the Goldwasser-Micali (GM) encryption scheme [34], whose security is based on the hardness of deciding quadratic residuosity. The GM encryption scheme is homomorphic over $\mathbb{Z}_2$. As mentioned above, (semantically-secure) threshold homomorphic encryption schemes have many important applications; for example, efficient multi-party computation can be based on any (efficient) scheme of this type [29, 13]. Threshold GM encryption can also be used for distributed tallying in electronic voting [37].

Concurrent with the present work, a variant threshold GM-like cryptosystem has been constructed [13] using an alternate approach. However, this scheme (which builds on [29]) requires the DDH assumption in $\mathbb{Z}_N^*$, whereas the security of our construction relies only on the quadratic residuosity assumption. Indeed, eliminating this assumption is left as an open question in [13]. We believe our solution also offers a more efficient and conceptually simpler method. Finally, our scheme has the added advantage of allowing for efficient distributed key generation when a trusted dealer is not assumed; this is not possible in [13] because they require N to be a product of safe primes.[2]

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 the Rabin-Williams signature scheme [39, Section 11.3.4] (see also [44]), variants of which have been standardized. Security of this scheme has recently been shown [12] to be equivalent to the hardness of factoring in the random oracle model (see also earlier work of [3]). Our techniques may be adapted for other cryptosystems whose security is based on factoring, such as the Rabin encryption scheme [44].

Efficiency improvements. The protocols we present are all efficient and *practical* threshold schemes. When a trusted dealer cannot be assumed (and key generation must therefore be done in a distributed fashion), our threshold schemes are more efficient than previous solutions not requiring a trusted dealer [15, 22]. The threshold schemes presented here may be easily executed in a modular manner following a "streamlined" version of the distributed key-generation protocols of [5, 27]: all information required by the present schemes is in place upon completion of these key-generation protocols, and we do not require that N be a product of safe primes. A "streamlined" version of these protocols may be used because *we do not require computation of an inverse over a shared (secret)*

[2] The recent work of [1] shows how N of this form can be generated efficiently in a distributed fashion; even so, it remains more efficient to generate N without this added requirement.

modulus (and therefore are done once N has been generated). We are therefore able to avoid *altogether* the step whose efficiency is improved by [10].

Finally, we believe the methods outlined in this paper are interesting in their own right; the sharing of the factors of N alone, without the need to additionally share a "decryption exponent", is a new paradigm for threshold cryptography over composite moduli and may prove useful in the design of future schemes. It is specifically useful whenever the function to be computed can be expressed as a combination of the factors and where the computation of its partial results is enabled by shares of the factors.

2 Model and Definitions

2.1 The Model

PARTICIPANTS. The participants are ℓ servers $\{P_1, \ldots, P_\ell\}$ and a trusted dealer D.[3] The dealer 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 executions of the protocol. Following [30], we assume the participants are connected by a complete network of private channels. In addition, all players have access to an authenticated broadcast channel so that the true sender of a message can always be correctly determined. These assumptions allow us to focus on high-level descriptions of the protocols; however, they may be instantiated using standard cryptographic techniques (in the proactive setting, care needs to be taken; see [41, 35]).

THE ADVERSARY. Our k-out-of-ℓ schemes assume a non-adaptive adversary who may corrupt up to $k-1$ participants in advance of protocol execution. 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. 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 consider both types of adversaries in what follows. In the case of threshold signature schemes, the adversary may submit signing requests to the system at any time; in the case of threshold decryption, we consider both chosen plaintext and chosen ciphertext attacks.

2.2 Security

Formal definitions of security for threshold cryptosystems have appeared elsewhere [31]. 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

[3] We stress that this trusted dealer is not essential to our schemes since a distributed algorithm (adapting [5, 27]) may be run when a dealer is not available.

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 [31], 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 A Threshold Homomorphic Encryption Scheme

We begin by describing how to achieve threshold decryption for the well-known homomorphic encryption scheme of Goldwasser and Micali [34] (henceforth, GM). The GM encryption scheme is as follows: the public key is a composite $N = pq$, where p and q are prime 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. To do this, first calculate the Jacobi symbol $J = (\frac{C}{N})$. If $J \neq 1$, 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 $b' = \pm 1$ and furthermore C is a quadratic residue iff $b' = 1$. The original plaintext can be recovered as $b = (1 - b')/2$. This scheme is semantically secure under the quadratic residuosity assumption [34].

3.1 An ℓ-out-of-ℓ Protocol

For simplicity and clarity of exposition, we describe in this section a protocol for "basic" threshold GM decryption (cf. Figure 1) which 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. 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 discuss how to remove the trusted dealer and perform the initial key generation 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 $d = (N - p - q + 1)/4$; note that d is always an integer. For all i, the dealer chooses integers $p_i, q_i \in_R (0, 2^{2n})$ 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). We note that it would suffice for the dealer to send

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 $q_0 = q - \sum_{i=1}^{\ell} q_i$
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 is done publicly)
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 $b = \left(1 - \Pi_{i=0}^{\ell} b_i \bmod N\right)/2$

Fig. 1. ℓ-out-of-ℓ decryption for the GM cryptosystem

$(p_i + q_i)/4$ to each party — and this is likely what would be done in practice — but we prefer the present description for pedagogical reasons.

Decryption. Decryption of a ciphertext C proceeds as follows: first, the Jacobi symbol $J = (\frac{C}{N})$ is computed; 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 exponent is an integer and hence b_i can be efficiently computed). Players publicly compute $b_0 = C^{(N - p_0 - q_0 + 1)/4} \bmod N$ (again, by design, the exponent is an integer). 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 $b = \frac{1 - b'}{2}$.

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-Williams signature scheme given below (cf. Theorem 4), and is therefore omitted.

3.2 Extensions

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 (but see [4] for another approach).

One approach is to adapt the suggestions of Rabin [45] to our setting. 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 the field $\mathbb{Z}_P$ such that $f_i(0) = p_i$. To player j, the dealer sends $f_i(j)$ for $1 \le i \le \ell$. 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. 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 [45].

A problem with this approach is that it may unfairly penalize servers which are temporarily off-line or otherwise unable to participate in an execution of the protocol. If this happens, this player's share is publicly reconstructed and hence available to an adversary eavesdropping on the protocol. Note that it may be much easier for an adversary to disconnect or prevent communication from a player than to corrupt a player (even passively). By "disconnecting" users one-by-one — possibly in parallel — an adversary may be able to obtain the secret key of the system.[4]

An alternative is to use ideas motivated by the protocols of Frankel, et al. [26]. Let $L = \ell!$. Instead of the ℓ-out-of-ℓ *additive* sharing illustrated in Figure 1, the dealer now performs k-out-of-ℓ *polynomial* sharing as follows: The dealer chooses $s^* \in_R (0, 2^{2n})$ subject to $s^* = 0 \bmod 4$, and additionally chooses a $(k-1)$-degree polynomial f over the integers — with coefficients chosen uniformly from $\{0, 4L, \ldots, L^3 2^{3n} k\}$ — such that $f(0) = L^2 s^*$. The dealer distributes $s_i \stackrel{\text{def}}{=} f(i)$ to player i. Finally, the dealer broadcasts the value $p + q - L^2 s^*$. To decrypt, the players first choose a random subset Λ consisting of k players. Each player in Λ computes the appropriate Lagrange interpolation coefficient $z_{i,\Lambda}$ and sets his (temporary) share to $\hat{s}_i = z_{i,\Lambda} \cdot s_i$. Note that, due to the careful choice of the polynomial f, the $\{\hat{s}_i\}$ may be computed over the integers and furthermore $\hat{s}_i = 0 \bmod 4$ for all i. The $\{\hat{s}_i\}_{i \in \Lambda}$ thus constitute a k-out-of-k *additive* sharing of $L^2 s^*$, and may be used to decrypt as in Figure 1. Techniques to achieve robustness for the above approach are given in [26].

Theorem 2. *The protocol of Figure 1 modified using 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.*

(Informal Idea of the) Proof The approach of Rabin [45] may be viewed as a "generic" approach which converts *any* ℓ-out-of-ℓ scheme to a k-out-of-ℓ scheme. The approach of Frankel, et al. [26] must be more carefully modified for

[4] This was pointed out to us by an anonymous referee.

the cryptosystem at hand; for the modification sketched above, however, a proof follows easily using their techniques. ■

Robustness. We may distinguish two methods for adding robustness to the above protocol: methods which work for arbitrary N, and methods which work only when N is a product of strong primes[5]. Methods specialized for the latter case can be more efficient; on the other hand, when distributed key generation is required, methods which work for arbitrary N may be preferred because distributed generation of N a product of safe primes [1] is less efficient.

Gennaro, et al. [31] give two methods for verifying correctness of the partial outputs b_i when N is a product of strong primes. One method, which is non-interactive, requires the dealer to distribute verification information to all players during the dealing phase; namely, $V_{i,j}$ is sent to player i to enable his verification of player j. When executing the protocol, player i outputs b_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 $O(\ell^2)$ memory for each player, and also increases the communication of the protocol (per player) to $O(\ell^2)$.

A second approach of [31] 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 $\log_g w_i = \log_C b_i$. Unfortunately, this approach seems to require interaction even in the random oracle model. More recently, Shoup [46] (based on earlier work of [11]) describes a non-interactive, zero-knowledge proof (in the random oracle model) for equality of discrete logarithms. Here, players work in the subgroup of quadratic residues $Q_N \subset \mathbb{Z}_N^*$: the dealer chooses $g \in Q_N$ and player i now proves that $\log_g w_i = \log_{C^2} b_i^2$ (squaring is necessary to ensure that values are in Q_N).

The above approaches suffice for N a product of strong primes. For general N, however, we must use other techniques to achieve robustness.[6] One possibility is to use the cryptographic program-checking method of [24], 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 [11] 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 [27], 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. Soundness can thus be guaranteed by 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. Fouque and Stern [22] suggest another method for achieving robustness; they require N of a special form but show how such N can be generated efficiently in a distributed manner.

---

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

[6] Although we still refer to a dealer, the techniques described here can be implemented easily following the (robust) distributed key-generation protocol of [27].

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, thus 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 ($k \leq \ell/2$) threshold (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 evident 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., [46]). (Even though a protocol has recently been given [1] for efficiently generating N of this form in a distributed fashion, this protocol remains less efficient than protocols for more general N [5, 27].) Furthermore, (2) an expensive step of the distributed key-generation protocol can be skipped entirely. Specifically, computation of an inverse[7] over $\varphi(N)$ (recall that $\varphi(N)$ must remain hidden from the players) is not required in our scheme.

The protocol of Figure 1 may be combined modularly with the distributed key-generation protocols of [5, 27]. Following execution of these key-generation protocols, all the players *already* have additive shares (p_i, q_i) of the factors of N. A small complication is that the protocol requires all players to have $p_i = q_i = 0 \bmod 4$. To deal with this, simply have player i choose $p_i = q_i = 0 \bmod 4$. Additionally, the "public remainder" may be set to $(p_0, q_0) = (3, 3)$. Decryption is then done as before. A similar approach was used in, e.g., [5] where they require $p = q = 3 \bmod 4$.

Proactive security. Proactive security may be added to our protocols using known techniques. For example, if the approach of Rabin [45] 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 Frankel, et al. [26] 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.

Chosen-ciphertext security. A generic method for making threshold cryptosystems secure against chosen-ciphertext attack was recently described [20], adapting the method of Naor and Yung [40] for the random oracle model. What is required are two schemes and an honest-verifier ZK proof of knowledge that two encryptions are of the same plaintext. Such a proof system for the GM cryptosystem is presented in Appendix A. Although the protocol given there is interactive, it can be made non-interactive (and reasonably efficient) in the random-oracle model.

[7] This is precisely the step whose efficiency is improved by [10]. Here, we avoid this step altogether!

4 A Threshold Signature Scheme Based on Factoring

Distributing the prime factors of the modulus among the participants offers a new paradigm for the construction of threshold systems over composite moduli. As a further example of the applicability of our technique, we describe a method for distributing the Rabin-Williams signature scheme [44], variants of which have been standardized as ISO 9796-2 and PKCS#1 v1.5. This scheme is particularly interesting since it offers the first threshold signature scheme whose security can be based on the hardness of factoring (in the random oracle model) [12].

4.1 The (Modified) Rabin Signature Scheme

The modified Rabin signature scheme [39, Section 11.3.4] is defined as follows: 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 $N = pq$ (N of this form are called *Williams integers*). The private key is $d = (N - p - q + 5)/8$.

Messages m to be signed are assumed to be appropriately encoded and the resulting underlying message space is $\mathcal{M} = \{m : m = 6 \bmod 16\}$ (see [12]). 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 $m \stackrel{?}{=} \tilde{m}$
- If $\tilde{m} = 3 \bmod 8$, verify whether $m \stackrel{?}{=} 2\tilde{m}$
- If $\tilde{m} = 7 \bmod 8$, verify whether $m \stackrel{?}{=} N - \tilde{m}$
- If $\tilde{m} = 2 \bmod 8$, verify whether $m \stackrel{?}{=} 2(N - \tilde{m})$

We refer the reader to [39, Section 11.3.4] for a proof of correctness and further discussion.

4.2 An ℓ-out-of-ℓ 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 $p'_i, q'_i \in_R (0, 2^{2n})$ such that $p_i = q_i = 0 \bmod 8$. 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., as discussed

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 is done publicly)
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 $s = \Pi_{i=0}^{\ell} s_i \bmod N$

Fig. 2. ℓ-out-of-ℓ signing for the Rabin signature scheme

above). 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 design, the exponent is an integer and hence s_i can be efficiently computed). Players publicly compute $s_0 = \tilde{m}^{(N-p_0-q_0+5)/8} \bmod N$ (again, by design, the exponent is an integer). Finally, the signature is computed as $s = \Pi_{i=0}^{\ell} s_i \bmod N$. Verification of the signature is exactly as described in Section 4.1.

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

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 $\{p_i, q_i\}_{1 \le i \le \ell-1}$ have the same distribution as in a real execution of the protocol.
– The distribution on (p_0, q_0), conditioned on the values of $\{p_i, q_i\}_{1 \le i \le \ell-1}$ 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.

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 $J = \left(\frac{m}{N}\right)$
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 $s_\ell = s / \left(\Pi_{i=0}^{\ell-1} s_i\right) \bmod N$

Fig. 3. Simulator for ℓ-out-of-ℓ threshold Rabin signature scheme

Simulatability of the signature generation phase (in particular, the value s_ℓ) follows easily from the simulatability of the dealing phase. ■

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 *encryption* scheme [44], whose semantic security may be based on the hardness of factoring.

References

1. J. Algesheimer, J. Camenisch, and V. Shoup. Efficient Computation Modulo a Shared Secret with Application to the Generation of Shared Safe-Prime Products. Crypto 2002.
2. D. Beaver and S. Haber. Cryptographic Protocols Provably Secure Against Dynamic Adversaries. Eurocrypt '92.
3. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures — How to Sign with RSA and Rabin. Eurocrypt '96.
4. S.R. Blackburn. Combinatorics and Threshold Cryptography. In *Combinatorial Designs and their Applications*, F.C. Holroyd, et al., eds., CRC Press, 1999.
5. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. Crypto '97.
6. C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, eds., *Cryptography and Coding*, Clarendon Press, 1989.
7. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively Secure Multi-Party Computation. STOC '96.
8. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive Security for Threshold Cryptosystems. Crypto '99.
9. R. Canetti and S. Goldwasser. An Efficient Threshold Public-Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. Eurocrypt '99.

10. D. Catalano, R. Gennaro, and S. Halevi. Computing Inverses over a Shared Secret Modulus. Eurocrypt 2000.
11. D. Chaum and T. Pedersen. Wallet Databases and Observers. Crypto '92.
12. J.S. Coron. Security Proof for Partial-Domain Hash Signature Schemes. Crypto 2002.
13. R. Cramer, I. Damgård, and J.B. Nielson. Multiparty Computation from Threshold Homomorphic Encryption. Eurocrypt 2001.
14. I. Damgård and M. Jurik. A Generalization, a Simplification, and Some Applications of Paillier's Probabilistic Public-Key System. PKC 2001.
15. I. Damgård and M. Koprowski. Practical Threshold RSA Signatures without a Trusted Dealer. Eurocrypt 2001.
16. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. STOC '94.
17. Y. Desmedt. Society and Group-Oriented Cryptography: A New Concept. Crypto '87.
18. Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto '89.
19. Y. Desmedt and Y. Frankel. Shared Generation of Authenticators and Signatures. Crypto '91.
20. P.-A. Fouque, and D. Pointcheval, Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. Asiacrypt 2001.
21. P.-A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. Financial Cryptography, 2000.
22. P.-A. Fouque and J. Stern. Fully Distributed Threshold RSA under Standard Assumptions. Asiacrypt 2001.
23. Y. Frankel. A Practical Protocol for Large Group-Oriented Networks. Eurocrypt '89.
24. Y. Frankel, P. Gemmell, and M. Yung. Witness-Based Cryptographic Program Checking and Robust Function Sharing. STOC '96.
25. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. Crypto '97.
26. Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptography. FOCS '97.
27. Y. Frankel, P. MacKenzie, and M. Yung. Robust Efficient Distributed RSA Key Generation. STOC '98.
28. Y. Frankel, P. MacKenzie, and M. Yung. Adaptively-Secure Distributed Public-Key Systems. European Symposium on Algorithms '99.
29. M. Franklin and S. Haber. Joint Encryption and Message-Efficient Secure Computation. J. Crypto 9(4): 217–232 (1996).
30. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. Eurocrypt '96.
31. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. J. Crypto 13(2): 273–300 (2000).
32. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log-Based Cryptosystems. Eurocrypt '99.
33. O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game. STOC '87.
34. S. Goldwasser and S. Micali. Probabilistic Encryption. JCSS 28(2): 270–299 (1984).
35. A. Herzberg, M. Jakobsson, S, Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Systems. CCCS '97.
36. S. Jarecki and A. Lysyanskaya, Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. Eurocrypt 2000.

37. J. Katz, S. Myers, and R. Ostrovsky. Cryptographic Counters and Applications to Electronic Voting. Eurocrypt 2001.
38. E. Kushilevitz and R. Ostrovsky. Replication is not Needed: Single Database Computationally-Private Information Retrieval. FOCS '97.
39. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1999.
40. M. Naor and M. Yung. Public-key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. STOC '90.
41. R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC '91.
42. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt '99.
43. T. P. Pedersen. A Threshold Cryptosystem Without a Trusted Party. Eurocrypt '91.
44. M. O. Rabin. Digital Signatures and Public Key Functions as Intractable as Factoring. Technical Memo TM-212, Lab. for Computer Science, MIT, 1979.
45. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. Crypto '98.
46. V. Shoup. Practical Threshold Signatures. Eurocrypt 2000.
47. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Against Chosen Ciphertext Attack. Eurocrypt '98.

A Proof of Equality for GM Ciphertexts

Input: Blum integers N_1, N_2 and X_1, X_2 where:
$\{X_1 = (-1)^b x_1^2 \bmod N_1, X_2 = (-1)^b x_2^2 \bmod N_2\}$ with $x_j \in Z^*_{N_j}$ and $b \in \{0, 1\}$.

Repeat k times:

1. The prover chooses a random bit c and "twin encrypts" it; i.e.,
   $\{V_1 = (-1)^c v_1^2 \bmod N_1, V_2 = (-1^{)} c v_2^2 \bmod N_2\}$ for random $v_j \in Z^*_{N_j}$.
   The prover sends V_1, V_2.
2. The verifier chooses a challenge bit d and sends it.
3. The prover responds by sending:
   $\{m_1 = v_1 x_1^d \bmod N_1, m_2 = v_2 x_2^d \bmod N_2\}$
4. The verifier checks that there exists a bit a such that both:
   $m_1^2 = (-1)^a \cdot V_1 \cdot X_1^d \bmod N_1$ and $m_2^2 = (-1)^a \cdot V_2 \cdot X_2^d \bmod N_2$

The verifier accepts only if the checks succeed in all iterations.

Fig. 4. Proof of knowledge of "twin" GM-encryption

The above proof system is complete and sound; furthermore, it is easy to show that it is an honest-verifier zero-knowledge proof of knowledge (in fact, it remains honest-verifier zero-knowledge when the k iterations are run in parallel). To turn this to a non-interactive proof of knowledge in the random oracle model, we can use the standard Fiat-Shamir technique.