Some Recent Research Aspects of Threshold Cryptography

Yvo Desmedt*

Department of Electrical Engineering and Computer Science, and
the Center of Cryptography, Computer and Network Security
University of Wisconsin–Milwaukee
PO Box 784, Milwaukee, WI 53201, U.S.A.
e-mail: desmedt@cs.uwm.edu

Department of Mathematics
Royal Holloway
University of London
U.K.

Abstract. In the traditional scenario in cryptography there is one sender, one receiver and an active or passive eavesdropper who is an opponent. Depending from the application the sender or the receiver (or both) need to use a secret key. Often we are not dealing with an individual sender/receiver, but the sender/receiver is an organization. The goal of threshold cryptography is to present practical schemes to solve such problems without the need to use the more general methods of mental games. In this paper we survey some recent research results on this topic. In particular on: DSS based threshold signatures, robust threshold cryptography, threshold cryptography without a trusted dealer, more optimal secret sharing schemes for threshold cryptography, proactive threshold cryptography and its generalizations.

1 Introduction

Public key [31] allows any sender to send private data to a known receiver (or to a receiver whose public key is authentic [58]). A public key system can also be used to digitally sign documents. Any receiver who knows the authentic public key of the sender can check whether data originated from the sender, or that it was created or modified by an outsider. Conventional cryptography achieves similar properties in a weaker sense [55, 64].

Traditionally, cryptography considers the case where there is one sender and one receiver. However, a lot of communication is between an individual and an organization, *e.g.*, a company, a governmental agency, a non-profit organization, etc. Examples are utility bills, tax forms, newsletters from professional organizations, etc. Organizations need also to communicate to each other. Moreover,

* A part of this work has been supported by NSF Grant NCR-9508528, the E.P.S.R.C. and by CNR AI n.94.00011.

many, and certainly security related, actions are taken by a group of people instead of by an individual. Indeed, the authority to sign a document is often *not* in the hands of a single person. In a bank wholesale transactions need to be signed by two co-signers. In the parliament (house) it is not the speaker who votes on a proposal for a new law, but a majority and this is also true in other democratic institutes (*e.g.*, a board of directors).

So there is a need for guaranteeing the authenticity of messages sent by a group of individuals to another group (or person) and this without enormous expansion of keys and/or messages. One often knows an organization (and its public key), but not necessarily who works in this organization or even less who has the power to sign in the name of the organization. So, to avoid a key management problem and to allow distribution of power an organization should have mainly one public key, instead of relying on the many public keys of the individuals inside that organization, who are unknown to the outside world. If the organization has one public key, the power to sign (or to decrypt, or to use a cryptosystem) should then be shared, to avoid abuse and to guarantee reliability. In our mechanical society such properties are often achieved. The goal of threshold cryptography is to make this possible in an electronic society. It combines threshold schemes [7, 62] (or secret sharing schemes [46]) with cryptography.

In this paper, we first discuss some of the first attempts to address the aforementioned issues, see Section 2. We then give a brief survey of some basic schemes that achieve this goal, see Section 3. Recent research results are briefly surveyed in Section 4. Finally in Section 5 we give some further details.

2 Early attempts

Shamir [62] was the first to discuss that a company's secret key, used to digitally sign documents, should not be given to a single entity. He noted that:

> Consider, for example, a company that digitally signs all checks (see RSA ...). If each executive is given a copy of the company's secret signature key, the system is convenient but easy to misuse. If the cooperation of all the company's executives is necessary in order to sign each check, the system is safe but inconvenient. The standard solution requires at least three signatures per check, and it is easy to implement with a (3,n) threshold scheme. Each executive is given a small magnetic card with one D_i piece, and the company's signature generating device accepts any three of them in order to generate (and later destroy) a temporary copy of the actual signature key D. The device does not contain any secret information and thus need not be protected against inspection. An unfaithful executive must have at least two accomplices in order to forge the company's signature in this scheme.

The solution suffers from many problems, the company's signature generating device can:

- leak the master key,
- modify the message being signed (indeed the co-signers have no control over the message that is allegedly being signed), and
- sign extra messages.

So full trust is necessary in the manufacturer of the device and the one who operates it. So this solution is not very secure.

The fact that sometimes a ciphertext needs to be decrypted jointly by a group of users, instead of by a single user, was addressed in [17] and later in [26]. The first solution presented is not very secure and the second one is far from practical since it relies on mental games (general secure distributed computation) mechanisms [43, 4, 13].

3 Basic schemes

In this section, for simplicity, we mainly follow the description used in [19], and discuss a generalization of it in Section 4.4.

A cryptosystem corresponds to evaluate a function with two inputs. One of them is the key, and the other one varies from application to application, examples being: the plaintext, the ciphertext, the text to sign, a seed, etc. Often this function is written as having one input with a key as parameter, *e.g.*, $f_{\text{key}}(\text{input})$. In our context it is useful to view that input as a parameter, so that we have

$$f_{\text{key}}(\text{input}) = g_{\text{input}}(\text{key}), \tag{1}$$

where f and g are functions.

Some important cryptographic schemes have the property that a function g, playing a major component, is homomorphic [47], *i.e.*,

$$g_b(k_1 + k_2) = g_b(k_1) * g_b(k_2), \tag{2}$$

where b is the aforementioned input, and k_1, k_2 belong to the key space. As a first example consider the ElGamal encryption scheme [32]. First the public key is (g, y, p) where g has a large enough order, p is a large enough prime, $y = g^a \bmod p$, and a is the secret key. The ciphertext is $(c_1, c_2) = (g^k \bmod p, M \cdot y^k \bmod p)$, where $M \in Z_p$ is the message. Now, a major component is the computation of $g_{c_1}(a) = c_1^a \bmod p$, since, given c_2, it allows the computation of the plaintext M [21]. As a second example we consider RSA signature generation and decryption. When signing or decrypting one computes $g_b(d) = b^d \bmod n$, where b is the (hashed and processed) text to sign or respectively the ciphertext, d is the secret key and n is the public modulus, *i.e.*, the product of two primes.

Now Shamir's secret sharing scheme [62] satisfies the property that:

$$\text{key} = \sum_{i \in B} (\text{constant}_{i,B}) \cdot (\text{share}_i)\,, \tag{3}$$

where B is a subset of the set of all participants, called A, and $|B| = t$, the threshold. So, if Shamir's secret sharing scheme is used and g is homomorphic we obtain, using (3) and (2) that

$$g_{\text{input}}(\text{key}) = g_{\text{input}}\left(\sum_{i \in B}(\text{constant}_{i,B}) \cdot (\text{share}_i)\right) \tag{4}$$

$$= \prod_{i \in B} g_{\text{input}}\left(\text{constant}_{i,B} \cdot \text{share}_i\right)$$

$$= \prod_{i \in B} \left(g_{\text{input}}(\text{share}_i)\right)^{\text{constant}_{i,B}}. \tag{5}$$

So, in (5) $g_{\text{input}}(\text{share}_i)$ can be evaluated by the shareholder and sent together with his identity (*i.e.*, i) to a reliable combiner. If enough shareholders responded, the combiner, knowing the identities, can compute a set B' which cardinality must be at least as large as the threshold t, *i.e.*, $|B'| \geq t$. The combiner chooses a $B \subseteq B'$ such that $|B| = t$, computes $\text{constant}_{i,B}$ for all $i \in B$ and then evaluates (5).

Originally $g_{\text{input}}(\text{share}_i)$ was called the partial result, but now one also refers to it as partial signature, partial decryption, etc., depending from the context.

Let us now discuss the security aspects. The combiner needs to be reliable, otherwise a fault tolerant implementation is required. If the final result of the computation will be public (*e.g.*, when signing), the combiner is allowed to reveal all the information he receives, otherwise (*e.g.*, when decrypting ciphertext) he cannot. The security goal is that the resulting scheme is as secure as the original one. It should be observed that a cryptanalyst may now receive as extra information:

- up to $t-1$ shares, given by up to $t-1$ corrupted shareholders,
- up to $l = |A|$ partial results, and this for each $g_{\text{input}_j}(\text{key})$ that was revealed to the cryptanalyst.

So the task of the cryptanalyst should be as hard regardless whether he received this extra information or not. This is usually achieved by requiring that any $t-1$ shares can be simulated (zero-knowledge [44]) and that the partial results can be simulated when $g_{\text{input}_j}(\text{key})$ is given (minimal-knowledge [39]).

It should be noted that Shamir's original secret sharing scheme only works over a finite field. However, the secret key in RSA belongs to $Z_{\phi(n)}(+)$, which is not a finite field. The additive property of (3) is desirable since it allows one to obtain threshold cryptography as we explained in Equations (4)–(5). An extension of Shamir's secret sharing, presented in [25] works over any Abelian group (by viewing the secret as belonging to an "extended" key space). The equality:

$$g_{\text{input}}\left(\text{constant}_{i,B} \cdot \text{share}_i\right) = \left(g_{\text{input}}(\text{share}_i)\right)^{\text{constant}_{i,B}}$$

should then be viewed as the multiplicative notation of a scalar operation in a module [47] and not as an exponentiation. Other secret sharing schemes have

been presented to satisfy (3), as will be surveyed in Section 4.3 and discussed in more details in Section 5.2. Finally, $\phi(n)$ must remain secret, which implies that the shareholders should not know $Z_{\phi(n)}$. The reader interested in detailed descriptions of how these technical issues in these basic schemes have been solved can consult [10, 9, 33, 29, 22, 34, 19].

4 Recent research: a brief survey

For several cryptoschemes and applications one has developed threshold crypto variants, such as threshold zero-knowledge proofs, threshold pseudorandom generators, etc. The concept of threshold cryptography has also been extended to general access structures [46], so that the subsets of A (the set of the participants) authorized to jointly use the cryptoscheme are not necessarily specified by a threshold. Note that the security of these schemes varies. One has unconditionally secure schemes, proven secure (under a computational complexity assumption) ones, some are (proven) as secure as the original cryptoscheme, and finally some threshold cryptoschemes have heuristic security. We refer the reader to [24] for a survey of research done by 1994 on these topics. While that survey was very general, we restrict ourselves to discuss only a few topics and discuss threshold cryptography in a more narrow context. So, for example, we will not discuss group signatures [14]. We refer the reader interested in that topic to [12].

Recent research has mainly focused on:

1. **reliability**. Threshold cryptoschemes that are reliable are called *robust* and we briefly discuss those in Sections 4.1 and 5.3.
2. **security enhancements**, such as:
   (a) **no trusted dealer** (see Sections 4.2 and 5.4).
   (b) **proactive security and its generalization** (see Sections 4.2 and Section 5.5).
   (c) **insiders' anonymity** (see Section 4.2).
3. **efficiency**, as discussed in Sections 4.3 and 5.2.
4. **generalizations**, such as
   (a) **threshold DSS** (see Sections 4.4).
   (b) **abstraction**, (see Section 4.4).

We now briefly survey these issues.

4.1 Reliability

To analyze the reliability aspect, let us focus on (5). It is clear that if one (or more) shareholder sends one wrong partial result, $g_{\text{input}}(\text{share}_i)$ the result will (likely) be wrong. If a public key system is used, one can using the public key verify that the result is wrong. When the numbers of wrong partial results is small, and a public key system is used, an exhaustive search will evidently find out who sent the wrong partial result [61]. One can then recompute the result

$g_{\text{input}}(\text{key})$ ignoring the wrong partial results, provided one has at least $t + e$ partial results, where e is the number of wrong ones.

Recent research has focused on the computation of $g_{\text{input}}(\text{key})$ without an exhaustive search [28, 42, 37, 41].

4.2 Security enhancements

Several security enhancements have been proposed, which we now briefly discuss.

No trusted dealer One can wonder who computes the share of a participant. In the first schemes a single trusted dealer was often used. It is clear that such an approach can best be avoided, however this is not always that easy. We refer the reader to [57, 42, 8, 16] and to Section 5.4 for some details.

Proactive security and its generalizations One can wonder what should happen when a share is stolen or lost. Worse, what happens when an outsider collects more shares than the threshold? As already observed in [26], it is a bad idea to change the public key of a group, in particular when this group is well known. Those who have not updated their public key database will use the old one. Also the new public key must be certified enough times independently before it can be trusted.

The solution that has been proposed to address this problem is to get new *guaranteed correct* shares without relying on a trusted dealer and to keep the old public key as long as is reasonable possible. The old shares should be destroyed and the update should be done frequently enough, taken the power of the enemy who may collect shares into account.

The following is a more general problem. How given shares for authorized subsets of the participants in A, specified by an access structure Γ_A, can one, without a dealer, distribute new shares for an access structure $\Gamma'_{A'}$, where A' is the new set of participants. If $\Gamma_A \not\subseteq \Gamma'_{A'}$ it is clear that some shareholders must destroy their shares.

The concept of proactive secret sharing is based on [56] and its combination with threshold cryptography has been studied in [45, 36, 59]. The generalization has been studied independently in [23] and [35]. Prior work on redistributing secret shares was done outside the scope of threshold cryptography, as can be found in [15, 2].

Insiders' anonymity An outsider, not receiving the help of insiders nor of the combiner, sees only $g_{\text{input}}(\text{key})$, and therefore is unable to find out who was active in the computation as observed in [19]. However, insiders and the combiner may (*e.g.*, in threshold signatures) see $g_{\text{input}}(\text{share}_i)$ and i. Therefore they may find out who the active insiders are. In voting, for example, this is not desired and one needs to guarantee the anonymity of the insiders. A first solution has been proposed in [40]. Note that the opposite problem, the one of tracing who

was involved, was already studied in [52] and that this problem can usually be solved in robust threshold cryptosystems.

4.3 Efficiency

For several unconditionally secure threshold authentication schemes, *e.g.*, [30] and for ElGamal based threshold decryption [21] each share is as long as the key. However, when one uses the extended Shamir's secret sharing scheme [25] for threshold RSA one has:

$$l * \text{length(key)} \leq \text{length(share}_i) < 2 * l * \text{length(key)},$$

where l is the number of shareholders. (Note that when $t = l$ one has that $\text{length(share}_i) = \text{length(key)}$ [10, 33]). Secret sharing schemes have been developed that allow more efficient threshold RSA schemes. Two approaches have been followed: the one is guaranteed to work, while the other one is likely to work. In the last approach it is possible that although a certain set of participants has a cardinality larger or equal to t, they will not be able to perform the threshold computation in (5). We refer the reader to [20, 1, 6, 49].

4.4 Generalizations

***g* is not homomorphic** What if the function g is not homomorphic? This problem in its generality corresponds with the mental games problem [43, 4, 13]. In its generality no practical solution has been proposed to address this problem. For some algorithms, such as DSS, a practical approach may be desirable. This was studied in [50, 42]. It should be noted that, even for the non-robust schemes, there is a significant difference between those solutions and the RSA solution. In threshold RSA, if one trusts t shareholders (or more), but not $t-1$ (or less), t (non-faulty) shareholders are sufficient to jointly compute the result. However, in the threshold DSS schemes more than t are required to co-sign. In fact, in the Gennaro-Jarecki-Krawczyk-Rabin scheme at least $2t-1$ participants are required, which implies that if l is even and t corresponds to majority, (*i.e.*, $\lfloor l/2 \rfloor + 1$) no practical threshold DSS signature scheme has been presented so far.

Abstraction One can wonder whether there is a need that g is a homomorphism. More general approaches have been discussed in [3]. We briefly focus on one of those (see also [11]).

Suppose that shareholders of a key want to compute $g_{\text{input}}(\text{key})$ in a practical distributed way. This is, for example, possible if there exist a recomputation function η' and functions g' such that

$$g_{\text{input}}(\text{key}) = \eta' \left(g'_{\text{input}}(\text{share}_{i_1}), \ldots, g'_{\text{input}}(\text{share}_{i_t}) \right) .$$

5 Some details

It is clear that seeing the large number of papers that have appeared on threshold cryptography, that a book is needed to profoundly cover the aforementioned subtopics. To avoid giving no details whatsoever, a few topics will be chosen and discussed in some depth. We do no longer order the subtopics as we did in Section 4.

We first remind the reader when a secret sharing scheme is called homomorphic [5].

5.1 Homomorphic secret sharing

Let $(s_1, s_2, \ldots, s_l)$ be a share assignment of the key k and similarly $(s'_1, s'_2, \ldots, s'_l)$ be the shares of the key k'. Assume operations, denoted using "+", are defined on the share spaces and the key space. A secret sharing scheme is called homomorphic [5] if $((s_1 + s'_1), (s_2 + s'_2), \ldots, (s_l + s'_l))$ is a possible share assignment of the key $k + k'$. Shamir secret sharing scheme is homomorphic. In fact any secret sharing scheme satisfying (3) in which the shares belong to a module [47] (an Abelian group with scalars belonging to a ring), the $\text{constant}_{i,B}$ are scalars, and the keys belong to a submodule, is homomorphic, as is easy to verify.

5.2 More efficient schemes

As we surveyed in Section 4.3, the problem of making more efficient secret sharing schemes useful for threshold cryptography is in particular important for threshold RSA.

We only discuss here a variant for $t = 2$ of a scheme given in [20] and then generalize it using Kurosawa-Stinson interpretation of the scheme given in [6]. We use the occasion to explain how to use these schemes for threshold RSA.

We first explain the case $t = 2$.

An example Let l be the number of participants. Let $K(+)$ be a group and $k \in K$ be the secret. We number the participants i from 0 till $l-1$ and represent i in binary representation, *i.e.*, i corresponds to the bits $(i_1, \ldots, i_{\lceil \log_2(l) \rceil})$.

When creating shares, the dealer will choose $\lceil \log_2(l) \rceil$ independent uniformly random elements $r_c \in K$ $(1 \leq c \leq \lceil \log_2(l) \rceil)$. Through a secure channel participant i receives as sub-share $s_{i,c} = r_c$ when $i_c = 0$ and otherwise $s_{i,c} = k - r_c$. So, a participant i receives $\lceil \log_2(l) \rceil$ sub-shares.

We now discuss how i and j can reconstruct the key. If $i \neq j$ then in their binary representation there will be at least one column c in which they differ, *i.e.*, $i_c \neq j_c$. Assume that $i_c = 0$, then $k = s_{j,c} + s_{i,c}$. If the group $K(+)$ is Abelian, then the scheme is homomorphic and the reconstruction works regardless whether $i_c = 0$ or not.

Now we explain how to use this scheme for threshold RSA [20]. Assume, as in [19], that n is the product of primes of equal length[1]. The distributor chooses the shares as we explained using $K = Z_{\phi(n)}(+)$ and $k = d$. When co-signing, assume that m is the hashed and processed message. Each participant i sends the number i and the sub-partial signatures $\sigma_{i,c} = m^{s_{i,c}} \bmod n$ for all c. Assume that this was done by participants, let say i and j. Giving correct shares, knowing i and j, one can find a column c where $i_c \neq j_c$ and compute the signature since $\sigma_{i,c} * \sigma_{j,c} = m^{s_{i,c}+s_{j,c}} = m^d$. If participants i and j know in advance that they will be co-signing, then they only need to send one $\sigma_{i,c}$ instead of $\lceil \log_2(l) \rceil$ many.

Note that the shares in [20] are as long as in the variant we discussed here, but that more randomness is required.

A generalization The 2-out-of-l previous scheme is based on $\lceil \log_2(l) \rceil$ independent 2-out-of-2 sharing schemes. The scheme in [6] satisfies a similar property. This scheme inspired Kurosawa and Stinson and they gave a generalization we now discuss [49].

Let A and A' be finite sets, B a subset of A, $l = |A|$ and $l' = |A'|$ and F be a set of functions from A to A'. We assume that $l' < l$. A function f from A to A' is a perfect hash for B if f restricted to B is one-to-one. F is a Perfect Hash Family (l, l', t) if for all subsets $B \subset A$ with cardinality t there is at least one function f in F such that f is a perfect hash for B. Note that the binary representation defines such a Perfect Hash Family from the set A to $A' = \{0, 1\}$.

A Perfect Hash Family can now be used to construct new secret sharing schemes from old ones. Suppose that one is given a t-out-of-l' sharing scheme and F, a Perfect Hash Family (l, l', t). One can then construct a t-out-of-l secret sharing scheme. Let us first discuss how to distribute shares. For each $f \in F$, the dealer uses the share generation algorithm of the t-out-of-l sharing scheme producing l shares $s_{a'}$ for all $a' \in A'$. If $f(i) = f(j) = a'$ participant i and j receive as subshare $s_{a'}$. The randomness used when distributing shares corresponding to f is independent of the randomness utilized for the f' iteration. The total number of subshares is $|F|$ and the length of the share of a participant is the sum of the length of his subshares.

When t shareholders, let us say in $B \subset A$, want to reconstruct the secret, they find which $f \in F$ is a perfect hash for B and they use subshares corresponding with that f. The reconstruction algorithm of the t-out-of-l' is then used.

If the original t-out-of-l' sharing scheme can be used for threshold RSA, then so can the t-out-of-l one.

5.3 Robustness

A simple example as an introduction We start by discussing an unconditionally secure threshold cryptosystem. In 1974 Gilbert, MacWilliams and Sloane proposed the following authentication scheme. The sender and receiver

[1] Otherwise, work modulo $\phi(n)\lfloor n^2/\phi(n) \rfloor$ instead of modulo $\phi(n)$, similar as in [34].

have a common $a, b \in_R GF(q)$, which describes a secret line in the vectorspace $GF(q) \times GF(q)$ of points with coordinates (x, y) satisfying the equation $y = a \cdot x + b$. (The original description of this scheme was over a projective space instead of over a vector space.) To authenticate a message $m \in GF(q)$ the sender sends the point (m, MAC) on the line, *i.e.* the Message Authentication Code is $\text{MAC} = a \cdot m + b$. The receiver accepts the received message m' as authentic if (m', MAC') is on the secret line. If the secret is used only once the probability of a successful impersonation or substitution attack if $1/q$.

If q is a prime, then any homomorphic secret sharing scheme might be used to transform this scheme into a threshold authentication one. Indeed, let s_i be a share of a and s'_i be a share of b, then $\text{MAC}_i = m \cdot s_i + s'_i$ is a share of the MAC, called a partial MAC [22, 30, 27]. If Shamir's secret sharing scheme is used, as in [30], this scheme is not robust.

Using the connection between threshold schemes and error-correcting codes [18, 54, 48] this scheme can easily be made robust. Indeed, let the shares of a, *i.e.*, $(s_1, s_2, \ldots, s_l)$, and the shares of b, *i.e.*, $(s'_1, s'_2, \ldots, s'_l)$, be chosen as codewords in an appropriate linear code over $GF(q)$. If this is a good error-correcting code, then one can correct errors provided sufficiently many participants compute their partial MAC, *i.e.*, MAC_i. For security purposes it is necessary that the scheme is perfect and composite [5]. Perfectness means that $t - 1$ shares do not reveal anything about the key and compositeness that the revelation of *all* the shares of $k_1 + k_2$, where $k_1 + k_2 \in GF(q)$, does not reveal anything more about (k_1, k_2) than what $k_1 + k_2$ does. Reed-Solomon error-correcting codes [60] codes satisfy these properties when used as in [54].

Making threshold RSA robust From [54] it is rather obvious that the generalization of Shamir secret sharing scheme implies a generalization of Reed-Solomon codes. It therefore seems that RSA could be made robust using this generalized Reed-Solomon code. However, when the number of errors is rather large, no algorithm is known to locate the errors in this generalized Reed-Solomon code. Note that it is easy to prove that if one could efficiently generalize the Berlekamp-Massey algorithm [53] to work for this generalized Reed-Solomon code, that the discrete log problem and factoring problem would both be easy [28]. Whether there exists a polynomial-time algorithm to detect the error locations is still an open problem. However, the problem need not to be addressed to obtain robust threshold RSA.

The common method used to achieve robust RSA is to send more data and to rely on the fact that RSA is a public key algorithm. If each of t participants proves that he has sent the correct partial result, then one can ignore all other partial results. This implies that one only needs $t + e$ partial results, where e is the number of incorrect partial results. Note that in the McEliece-Sarwate's use of Reed-Solomon codes one needs $t + 2e$ shares to recompute the key. Several methods have been developed (see Section 4.1) and we only discuss one of those.

Gennaro-Jarecki-Krawczyk-Rabin [41] developed two methods to achieve robust threshold RSA. One is interactive and the other one is non-interactive.

We discuss the non-interactive one which is based on the Gilbert-MacWilliams-Sloane authentication scheme.

The scheme assumes that $n = pq$ is the product of safe primes (a prime p is safe if $p = 2p' + 1$ where p' is a prime). To detect an error the combiner will receive extra information $a_{i,j}$ and $b_{i,j}$, where i indicates the participant and j the subshare (see Section 5.2). In fact, for each subshare $s_{i,j}$ the dealer chooses uniformly random an $a_{i,j}$, such that $1 \leq a_{i,j} \leq n^{\delta_1}$ and an $b_{i,j}$ such that $1 \leq b_{i,j} \leq n^{1+\delta_1+\delta_2}$, where δ_1 and δ_2 are appropriately chosen [41]. The dealer gives participant i privately, the subshare $s_{i,j}$ and the integer $y_{i,j}$, where $y_{i,j} = a_{i,j} \cdot s_{i,j} + b_{i,j}$, for all j. The dealer also gives the combiner privately the integers $a_{i,j}$ and $b_{i,j}$ for all i and j. We trust that the combiner will keep these values secret and that the combiner will perform the verifications we now describe. Otherwise more combiners might be used as described in [41].

When co-signing participant i computes the sub-partial signatures $\sigma_{i,j}$ of m and sub-mac $Y_{i,j} = m^{y_{i,j}} \bmod n$ for each j and send those to the combiner. We now explain how the combiner can locate the correct partial signatures. To verify whether it is correct the combiner verifies if

$$(\sigma_{i,j})^{a_{i,j}} \cdot m^{b_{i,j}} = Y_{i,j} \bmod n,$$

for all received $\sigma_{i,j}$. The combiner now uses these partial signatures for which the verification was successful. Note that it is still possible that the correct sub-partial signature is $-\sigma_{i,j}$. For all existing threshold RSA schemes this implies that the actual signature may have to be multiplied by -1. Knowing e the combiner can easily check whether this is necessary.

5.4 Avoiding a trusted dealer

If the secret key can be chosen randomly, as is usually the case in a discrete log setting, then the following use of homomorphic secret sharing can be utilized towards abolishing the need for a trusted dealer.

The first participant chooses a uniformly random key k_1 and plays distributor of this key generating shares $(s_{1,1}, s_{1,2}, \ldots, s_{1,l})$. The first participant sends, using a secure channel, the shares $s_{1,i}$ to participant i and $1 \leq i \leq l$. Now, t participants, let say those in $B \subset A$, will perform similar operations (choosing the randomness independently) and create shares $s_{j,i}$ instead of $s_{1,i}$ and send those privately to participant i. A participant i can then compute the share $s_i = \sum_{j \in B} s_{j,i}$. Since the sharing scheme is homomorphic, s_i is a share of the key $k = \sum_{j \in B} k_j$. If the keys k_j belong to an Abelian group (see also [38]) and the secret sharing scheme is perfect, then $t-1$ shareholders have no information about the secret key k.

The first use of this idea in the context of threshold cryptography was in [57]. Pedersen's scheme also guarantees that the distribution is verifiable, *i.e.*, that the shares the shareholders received will always recompute the same secret key. Pedersen's scheme also guarantees that this secret key corresponds to the public key that is made public.

Note that the problem of avoiding a trusted dealer is much more complex in the context of RSA [8].

5.5 Proactive

We briefly explain how the use of homomorphic secret sharing is useful towards achieving proactive threshold cryptography. We assume that the secret sharing scheme is homomorphic.

If $(s_1, s_2, \ldots, s_l)$ is a share assignment for the key k and $(s'_1, s'_2, \ldots, s'_l)$ is a uniformly random share assignment for the "key" 0, then $(s''_1, s''_2, \ldots, s''_l) = (s_1 + s'_1, s_2 + s'_2, \ldots, s_l + s'_l)$ is a new share assignment for the same key k. Assume that one trusts t shareholders. Then t participants, denoted by j, can each contribute their own random $(s'_{j,1}, s'_{j,2}, \ldots, s'_{j,l})$. This is done in a similar way as in Section 5.4, however the shares correspond with the "keys" 0. When working in an Abelian group (see also [38]) and when the secret sharing scheme is perfect, the resulting share $s''_i = s_i + \sum_{j \in B} s'_{j,i}$ will be guaranteed independent of the original share s_i, due to the properties of the one-time-pad [63]. Both s_i and s''_i are shares of the same key k.

One should note that the schemes are more complex since each contributing shareholder needs to prove that his contribution $(s'_1, s'_2, \ldots, s'_l)$ consists of shares of 0. Also, achieving proactive threshold RSA is more complex (see Section 4.2).

6 Final comments

As mentioned, we did not discuss all aspects of threshold cryptography in this survey. For example, some threshold cryptoschemes (not cited in this paper) have some security problems as was pointed out in [51].

Acknowledgment

The author thanks the many researchers with whom he had discussions on the topic.

References

1. N. Alon, Z. Galil, and M. Yung. Efficient dynamic-resharing "verifiable secret sharing" against mobile adversary. In P. G. Spirakis, editor, *Algorithms — ESA '95, Third Annual European Symposium, Proceedings (Lecture Notes in Computer Science 979)*, pp. 523–537. Springer-Verlag, 1995. Corfu, Greece, September 25–27.
2. F. Bao, R. Deng, Y. Han, and A. Jeng. Design and analysis of two basic protocols for use in ttp-based key escrow. In V. Varadharajan, J. Pieprzyk, and Y. Mu, editors, *Information Security and Privacy, Second Australian Conference, ACISP '97, (Lecture Notes in Computer Science 1270)*, pp. 261–270. Springer-Verlag, 1997. Sydney, NSW, Australia, July 7–9.
3. A. Beimel, M. Burmester, Y. Desmedt, and E. Kushilevitz. Computing functions of a shared secret. Manuscript, 1995.

4. M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the twentieth annual ACM Symp. Theory of Computing, STOC*, pp. 113–131, May 2–4, 1988.
5. J. C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In A. Odlyzko, editor, *Advances in Cryptology, Proc. of Crypto '86 (Lecture Notes in Computer Science 263)*, pp. 251–260. Springer-Verlag, 1987. Santa Barbara, California, U.S.A., August 11–15.
6. S. R. Blackburn, M. Burmester, Y. Desmedt, and P. R. Wild. Efficient multiplicative sharing schemes. In U. Maurer, editor, *Advances in Cryptology — Eurocrypt '96, Proceedings (Lecture Notes in Computer Science 1070)*, pp. 107–118. Springer-Verlag, 1996. Zaragoza, Spain, May 12–16.
7. G. R. Blakley. Safeguarding cryptographic keys. In *Proc. Nat. Computer Conf. AFIPS Conf. Proc.*, pp. 313–317, 1979. vol.48.
8. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In B. S. Kaliski, editor, *Advances in Cryptology — Crypto '97, Proceedings (Lecture Notes in Computer Science 1294)*, pp. 425–439. Springer-Verlag, 1997. Santa Barbara, California, U.S.A., August 17–21.
9. C. Boyd. Some applications of multiple key ciphers. In C. G. Günther, editor, *Advances in Cryptology, Proc. of Eurocrypt '88 (Lecture Notes in Computer Science 330)*, pp. 455–467. Springer-Verlag, May 1988. Davos, Switzerland.
10. C. Boyd. Digital multisignatures. In H. Beker and F. Piper, editors, *Cryptography and coding*, pp. 241–246. Clarendon Press, 1989. Royal Agricultural College, Cirencester, December 15–17, 1986.
11. M. Burmester. Homomorphisms of secret sharing schemes. In U. Maurer, editor, *Advances in Cryptology — Eurocrypt '96, Proceedings (Lecture Notes in Computer Science 1070)*, pp. 96–106. Springer-Verlag, 1996. Zaragoza, Spain, May 12–16.
12. J. Camenish and M. Stadler. Efficient group signature schemes for large groups. In B. S. Kaliski, editor, *Advances in Cryptology — Crypto '97, Proceedings (Lecture Notes in Computer Science 1294)*, pp. 410–424. Springer-Verlag, 1997. Santa Barbara, California, U.S.A., August 17–21.
13. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM Symp. Theory of Computing, STOC*, pp. 11–19, May 2–4, 1988.
14. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology, Proc. of Eurocrypt '91 (Lecture Notes in Computer Science 547)*, pp. 257–265. Springer-Verlag, April 1991. Brighton, U.K.
15. L. Chen, D. Gollmann, and C. Mitchell. Key escrow in mutually mistrusting domains. In M. Lomas, editor, *Security Protocols (Lecture Notes in Computer Science 1189)*, pp. 139–153. Springer-Verlag, 1997. Cambridge, United Kingdom April 10-12, 1996.
16. C. Cocks. Split knowledge generation of RSA paremeters. Presented at the 6th IMA Conference on Coding and Cryptography, Cirencester, England, to appear in the proceedings, December 17–19, 1997.
17. R. A. Croft and S. P. Harris. Public-key cryptography and re-usable shared secrets. In H. Beker and F. Piper, editors, *Cryptography and coding*, pp. 189–201. Clarendon Press, 1989. Royal Agricultural College, Cirencester, December 15–17, 1986.
18. G. I. Davida, R. DeMillo, and R. Lipton. Protecting shared cryptographic keys. In *Proceedings of the 1980 Symposium on Security and Privacy*, pp. 100–102. IEEE Computer Society, April 1980. IEEE Catalog No. 80 CH1522-2.

19. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proceedings of the twenty-sixth annual ACM Symp. Theory of Computing (STOC)*, pp. 522–533, May 23–25, 1994. Montréal, Québec, Canada.
20. Y. Desmedt, G. Di Crescenzo, and M. Burmester. Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology — Asiacrypt '94, Proceedings (Lecture Notes in Computer Science 917)*, pp. 21–32. Springer-Verlag, 1995. Wollongong, Australia, November/December, 1994.
21. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology — Crypto '89, Proceedings (Lecture Notes in Computer Science 435)*, pp. 307–315. Springer-Verlag, 1990. Santa Barbara, California, U.S.A., August 20–24.
22. Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto '91, Proceedings (Lecture Notes in Computer Science 576)*, pp. 457–469. Springer-Verlag, 1992. Santa Barbara, California, U.S.A., August 12–15.
23. Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Tech. Report ISSE-TR-97-01, George Mason University, July 1997. ftp://isse.gmu.edu/pub/techrep/97_01_jajodia.ps.gz.
24. Y. G. Desmedt. Threshold cryptography. *European Trans. on Telecommunications*, 5(4), pp. 449–457, July-August 1994. (Invited paper).
25. Y. G. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics*, 7(4), pp. 667–679, November 1994.
26. Y. Desmedt. Society and group oriented cryptography : a new concept. In C. Pomerance, editor, *Advances in Cryptology, Proc. of Crypto '87 (Lecture Notes in Computer Science 293)*, pp. 120–127. Springer-Verlag, 1988. Santa Barbara, California, U.S.A., August 16–20.
27. Y. Desmedt. Threshold cryptography. In W. Wolfowicz, editor, *Proceedings of the 3rd Symposium on: State and Progress of Research in Cryptography*, pp. 110–122, February 15–16, 1993. Rome, Italy, invited paper.
28. Y. Desmedt. Extending Reed-Solomon codes to modules. In *Proceedings 1995 IEEE International Symposium on Information Theory*, p. 498, Whistler, BC, Canada, September 17–22, 1995.
29. Y. Desmedt and Y. Frankel. Perfect zero-knowledge sharing schemes over any finite Abelian group. In R. Capocelli, A. De Santis, and U. Vaccaro, editors, *Sequences II (Methods in Communication, Security, and Computer Science)*, pp. 369–378. Springer-Verlag, 1993. Positano, Italy, June 17–21, 1991.
30. Y. Desmedt, Y. Frankel, and M. Yung. Multi-receiver / multi-sender network security: efficient authenticated multicast/ feedback. In *IEEE INFOCOM '92, Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 2045–2054, Florence, Italy, May 4–8, 1992. .
31. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT–22(6), pp. 644–654, November 1976.
32. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31, pp. 469–472, 1985.
33. Y. Frankel. A practical protocol for large group oriented networks. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology, Proc. of Eurocrypt '89 (Lecture Notes in Computer Science 434)*, pp. 56–61. Springer-Verlag, 1990. Houthalen, Belgium, April 10–13.

34. Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Tech. Report TR–92–04–02, Dept. of EE & CS, Univ. of Wisconsin–Milwaukee, April 1992. ftp://ftp.cs.uwm.edu/pub/tech_reports/desmedt-rsa-threshold_92.ps.
35. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal resilience proactive public key cryptosystems. In *38th Annual Symp. on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, October 20–22, 1997. Miami Beach, Florida, U.S.A.
36. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In B. S. Kaliski, editor, *Advances in Cryptology — Crypto '97, Proceedings (Lecture Notes in Computer Science 1294)*, pp. 440–454. Springer-Verlag, 1997. Santa Barbara, California, U.S.A., August 17–21.
37. Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking and robust function sharing. In *Proceedings of the Twenty-Eighth Annual ACM Symp. on Theory of Computing*, pp. 499–508, May, 22–24, 1996.
38. Y. Frankel, Y. Desmedt, and M. Burmester. Non-existence of homomorphic general sharing schemes for some key spaces. In E. F. Brickell, editor, *Advances in Cryptology — Crypto '92, Proceedings (Lecture Notes in Computer Science 740)*, pp. 549–557. Springer-Verlag, 1993. Santa Barbara, California, U.S.A., August 16–20.
39. Z. Galil, S. Haber, and M. Yung. Minimum-knowledge interactive proofs for decision problems. *SIAM J. Comput.*, 18(4), pp. 711–739, August 1989.
40. C. Gehrmann and Y. Desmedt. Truly anonymous secret sharing. Manuscript.
41. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In N. Koblitz, editor, *Advances in Cryptology — Crypto '96, Proceedings (Lecture Notes in Computer Science 1109)*, pp. 157–172. Springer-Verlag, 1996. Santa Barbara, California, U.S.A., August 18–22.
42. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. Maurer, editor, *Advances in Cryptology — Eurocrypt '96, Proceedings (Lecture Notes in Computer Science 1070)*, pp. 354–371. Springer-Verlag, 1996. Zaragoza, Spain, May 12–16.
43. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth annual ACM Symp. Theory of Computing, STOC*, pp. 218–229, May 25–27, 1987.
44. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1), pp. 186–208, February 1989.
45. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing. In D. Coppersmith, editor, *Advances in Cryptology — Crypto '95, Proceedings (Lecture Notes in Computer Science 963)*, pp. 339–352. Springer-Verlag, 1995. Santa Barbara, California, U.S.A., August 27–31.
46. M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. In *Proc. IEEE Global Telecommunications Conf., Globecom'87*, pp. 99–102. IEEE Communications Soc. Press, 1987.
47. N. Jacobson. *Basic Algebra I.* W. H. Freeman and Company, New York, 1985.
48. E. D. Karnin, J. W. Greene, and M. Hellman. On secret sharing systems. *IEEE Tr. Inform. Theory*, 29(1), pp. 35–41, January 1983.
49. K. Kurosawa and D. Stinson, June 1996. Personal communication.
50. S. K. Langford. Threshold DSS signatures without a trusted party. In D. Coppersmith, editor, *Advances in Cryptology — Crypto '95, Proceedings (Lecture Notes in Computer Science 963)*, pp. 397–409. Springer-Verlag, 1995. Santa Barbara, California, U.S.A., August 27–31.

51. S. K. Langford. Weaknesses in some threshold cryptosystems. In N. Koblitz, editor, *Advances in Cryptology — Crypto '96, Proceedings (Lecture Notes in Computer Science 1109)*, pp. 74–82. Springer-Verlag, 1996. Santa Barbara, California, U.S.A., August 18–22.
52. C. Li, T. Hwang, and N. Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In A. De Santis, editor, *Advances in Cryptology — Eurocrypt '94, Proceedings (Lecture Notes in Computer Science 950)*, pp. 194–204. Springer-Verlag, May 9–12, 1995. Perugia, Italy, May 9–12.
53. F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes.* North-Holland Publishing Company, 1978.
54. R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Comm. ACM*, 24(9), pp. 583–584, September 1981.
55. A. Menezes, P. van Oorschot, and S. Vanstone. *Applied Cryptography.* CRC, Boca Raton, 1996.
56. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of the 10-th Annual ACM Symp. on Principles of Distributed Computing*, pp. 51–60, August 19–21, 1991. Montreal, Quebec, Canada.
57. T. P. Pedersen. A threshold cryptosystem without a trusted party. In D. W. Davies, editor, *Advances in Cryptology, Proc. of Eurocrypt '91 (Lecture Notes in Computer Science 547)*, pp. 522–526. Springer-Verlag, April 1991. Brighton, U.K.
58. G. J. Popek and C. S. Kline. Encryption and secure computer networks. *ACM Computing Surveys*, 11(4), pp. 335–356, December 1979.
59. T. Rabin. A simplified approach to threshold and proactive RSA. Manuscript.
60. I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *SIAM Journal on Applied Mathematics*, 8, pp. 300–304, 1960.
61. M. K. Reiter and K. P. Birman. How to securely replicate services. *ACM Transactions on programming languages and systems*, 16(3), pp. 986–1009, 1994.
62. A. Shamir. How to share a secret. *Commun. ACM*, 22, pp. 612–613, November 1979.
63. C. E. Shannon. Communication theory of secrecy systems. *Bell System Techn. Jour.*, 28, pp. 656–715, October 1949.
64. D. R. Stinson. *Cryptography: Theory and Practice.* CRC, Boca Raton, 1995.