Des. Codes Cryptogr. (2014) 73:841–864
DOI 10.1007/s10623-013-9831-6

New results and applications for multi-secret sharing schemes

Javier Herranz · Alexandre Ruiz · Germán Sáez

Received: 8 November 2012 / Revised: 8 May 2013 / Accepted: 8 May 2013 /
Published online: 23 May 2013
© Springer Science+Business Media New York 2013

Abstract In a multi-secret sharing scheme (MSSS), ℓ different secrets are distributed among the players in some set $\mathcal{P} = \{P_1, \ldots, P_n\}$, each one according to an access structure. The trivial solution to this problem is to run ℓ independent instances of a standard secret sharing scheme, one for each secret. In this solution, the length of the secret share to be stored by each player grows linearly with ℓ (when keeping all other parameters fixed). Multi-secret sharing schemes have been studied by the cryptographic community mostly from a theoretical perspective: different models and definitions have been proposed, for both unconditional (information-theoretic) and computational security. In the case of unconditional security, there are two different definitions. It has been proved that, for some particular cases of access structures that include the threshold case, a MSSS with the strongest level of unconditional security must have shares with length linear in ℓ. Therefore, the optimal solution in this case is equivalent to the trivial one. In this work we prove that, even for a more relaxed notion of unconditional security, and for some kinds of access structures (in particular, threshold ones), we have the same efficiency problem: the length of each secret share must grow linearly with ℓ. Since we want more efficient solutions, we move to the scenario of MSSSs with computational security. We propose a new MSSS, where each secret share has constant length (just one element), and we formally prove its computational security in the random oracle model. To the best of our knowledge, this is the first formal analysis on the computational security of a MSSS. We show the utility of the new MSSS by using it as a key ingredient in the design of two schemes for two new functionalities: multi-policy signatures and multi-policy

Communicated by C. Mitchell.

J. Herranz (✉) · A. Ruiz · G. Sáez
Department of Matemàtica Aplicada IV, Universitat Politècnica de Catalunya – BarcelonaTech,
C. Jordi Girona, 1-3, Mòdul C3, Barcelona, Spain
e-mail: jherranz@ma4.upc.edu

A. Ruiz
e-mail: aruiz@ma4.upc.edu

G. Sáez
e-mail: german@ma4.upc.edu

decryption. We prove the security of these two new multi-policy cryptosystems in a formal security model. The two new primitives provide similar functionalities as attribute-based cryptosystems, with some advantages and some drawbacks that we discuss at the end of this work.

Keywords Multi-secret sharing schemes · Multi-policy cryptosystems · Entropy · Provable security

Mathematics Subject Classification 94A62 · 94A60 · 94A17

1 Introduction

In public key cryptography, some operations (like encrypting a message or verifying a signature) can be done by any user in the system, with access to the public information of the other users. However, the associated *secret* operations (decrypting a ciphertext or signing a message) can be done only by the user who knows the corresponding secret information. Security of public key cryptosystems (against polynomial-time adversaries) is proved in a *computational* sense, by reduction to the hardness of some mathematical problem.

In some situations, such secret tasks are too important and sensitive to rely on a single user or machine; a good solution then is to use *distributed* (in particular, *threshold*) public key cryptography: the secret information is distributed among a set of users, and the cooperation of some authorized subset (in a fixed access structure) of them is required in order to correctly perform the corresponding secret task. Depending on the considered secret task, this approach leads to either distributed decryption schemes or distributed signature schemes.

In the design of such distributed schemes, a key ingredient are *secret sharing schemes*. These schemes have received a lot of attention since their introduction in 1979 [3,8,13,27,28]. Most of the secret sharing schemes proposed and analyzed so far enjoy *unconditional* (or *information-theoretic*) security, which means that the value of the shared secret is perfectly hidden to an (even computationally unbounded) adversary who controls any non-authorized subset of users. When secret sharing schemes are used in the design of distributed public key cryptosystems (that can enjoy computational security, at most), one could argue that requiring unconditional security for the underlying secret sharing schemes may be innecessarily restrictive. However, secret sharing schemes may have future applications in other scenarios with unconditional security, or in scenarios requiring security during concurrent executions of a protocol, and they are very interesting and mathematically rich by themselves. It is thus a good idea to obtain results about secret sharing schemes with unconditional security, for instance about lower and upper bounds on the ratio between the length of the secret and the length of the share to be stored by each user.

In this work we consider an extension of the standard scenario of distributed (public key) cryptography. In some cases, setting a single access structure of authorized subsets of users for all the executions of the secret task may be unrealistic. For instance, some messages encrypted for a receiver entity $\mathcal{P}$ may be more sensitive or confidential than others, and so require the cooperation of more or less members of $\mathcal{P}$ in order to be decrypted. With this motivation in mind, we will consider *multi-policy* distributed cryptosystems: in the setup of the system, a list of ℓ possible (and different) acces structures is chosen; later, for each execution of the cryptographic operation, a specific access structure in this list is chosen "on the fly", depending on the desired security level. Only those subsets of players authorized with respect to this specific access structure will be able to perform the secret task, by using

their secret shares of information. A trivial way of implementing multi-policy distributed cryptosystems is by running ℓ independent instances of a standard distributed cryptosystem, one for each of the access structures in the list. This solution has the drawback that the length of the secret information to be stored by each user is linear in ℓ, when keeping all other parameters fixed; we look for more efficient solutions.

As it happens with standard distributed cryptosystems, where standard secret sharing schemes are a key ingredient in their design, it is natural that *multi-secret sharing schemes* (MSSSs, for short) are a key tool when designing multi-policy distributed cryptosystems. In a MSSS, ℓ different secrets are distributed among the players in some set $\mathcal{P} = \{P_1, \ldots, P_n\}$, each one according to a (possibly different) access structure. Again, a trivial solution to design a MSSS is to run ℓ independent standard secret sharing schemes, one for each secret and access structure; the length of each secret share is linear in ℓ.

MSSSs have been studied, *per se*, in different works (see [4,23,25], for instance). As far as we know, no specific application of a MSSS into a more general scenario or cryptographic protocol has been explicitly proposed. Most of the works on MSSSs have focused on unconditionally secure MSSSs. Blundo et al. [4] introduced a strong definition for the unconditional security of a MSSS, and gave some lower bounds on the length of the secret shares to be stored in a MSSS enjoying that level of security. Masucci proposed in [25] a weaker (althoug still information-theoretic) notion of security for MSSSs, and also gave some lower bounds on the length of secret shares for schemes enjoying the two notions. For some particular cases, which include the threshold case where each access structure is defined by a threshold value, the results in [4,25] imply that the length of each secret share in a MSSS with the strong level of unconditional security must be, at least, linear in ℓ.

The first result in this paper, in Sect. 2, is a proof that this is also the case for MSSSs enjoying security in the weaker (but still information-theoretic) sense proposed by Masucci. That is, we show that for some lists of access structures (in particular, when all of them are threshold ones), the length of each secret share in a MSSS for these access structures will be linear in ℓ, even if the MSSS enjoys weaker unconditional security. Since our final goal is the design of multi-policy distributed schemes (in particular, for threshold policies) with shorter secret shares of information than those provided by the trivial solution (with length linear in ℓ), our first result is quite negative, and forces us to move to the weaker setting of MSSSs with computational security.

We stress here that computationally secure MSSSs will be enough for our purposes, because the security of multi-policy distributed cryptosystems can be at most computational, anyway. After describing formally the computational security of a MSSS, we present in Sect. 3 the second contribution of this paper: a new MSSS, inspired by that in [23], with a formal proof of computational security, in the random oracle model. Although we describe and analyze the scheme for the case of threshold access structures, it can be easily extended to more general access structures (see Appendix B).

Finally, we use this new MSSS as a key tool to design a new multi-policy distributed signature scheme (in Sect. 4) and a new multi-policy distributed decryption scheme (in Sect. 5). We prove the security of these two schemes, in the random oracle model, by taking into account formal security models that we introduce in the corresponding sections. Only the details for the signature scheme are included. Again, and for simplicity, we describe the schemes for the threshold case, but extensions to the case of more general policies are easy to do, as we discuss in Appendix B. The efficiency of the new multi-policy distributed cryptosystems is essentially the same as the efficiency of the standard distributed cryptosystems (Boldyreva [5] for signatures and Shoup-Gennaro [31] for decryption) by which they are inspired. Namely, the length of secret shares, ciphertexts and signatures, and the cost of encryption, decryption,

signature and verification are the same; the only change is in the size of the public parameters and public key of the set $\mathcal{P}$, which is increased by a factor of $n \cdot \ell$.

In Sect. 6 we discuss the relations between the new primitives of multi-policy distributed cryptosystems and attribute-based cryptosystems. Even if any attribute-based cryptosystem leads to a multi-policy distributed cryptosystem, the obtained scheme has some drawbacks that are not present in the solutions that we propose in Sects. 4 and 5. For instance, our new MSSS and our multi-policy distributed schemes can be modified (see Appendix A) so that no external and trusted entity is needed in the life of the system; this is not possible in attribute-based solutions, where a trusted master entity generates and distributes the secret information to the members of the set $\mathcal{P}$.

2 Multi-secret sharing schemes

In this section we recall the notions of information entropy, standard secret sharing schemes and MSSSs. Then we discuss the security properties required for multi-secret sharing schemes, in both an information-theoretic and a computational scenario. As our first (negative) result, we will prove that MSSSs enjoying information-theoretic security, for some families of access structures that include the threshold case, must have shares which are as long as the secret.

2.1 Entropy of random variables

Let X be a random variable that takes values in a finite set X. For any $x \in \mathsf{X}$, let $p(x) = \Pr[X = x]$ be the probability that X takes the value x. The *entropy* $H(X)$ of X is defined as

$$H(X) = -\sum_{x \in \mathsf{X}} p(x) \cdot \log(p(x)),$$

where $0 \cdot \log 0$ should be treated as being equal to zero. The entropy $H(X)$ measures the uncertainty on the value taken by the random variable X. It always satisfies $0 \leq H(X) \leq \log |\mathsf{X}|$. The minimum value $H(X) = 0$ is achieved if and only if there exists $x_0 \in \mathsf{X}$ such that $p(x_0) = 1$, and the maximum value $H(X) = \log |\mathsf{X}|$ is achieved if and only if the probability is distributed uniformly (that is, $p(x) = 1/|\mathsf{X}|$ for all $x \in \mathsf{X}$).

Given two random variables X, Y, their *joint entropy* is defined as

$$H(X, Y) = -\sum_{(x,y) \in \mathsf{X} \times \mathsf{Y}} p(x, y) \cdot \log(p(x, y)),$$

where $p(x, y) = \Pr[X = x \text{and} Y = y]$.

If we denote $p(x|y) = \Pr[X = x \mid Y = y]$, then the *conditional entropy* $H(X|Y)$ is defined as

$$H(X|Y) = -\sum_{x \in \mathsf{X}} \sum_{y \in \mathsf{Y}} p(y) p(x|y) \cdot \log(p(x|y)),$$

and it satisfies $H(X|Y) = H(X, Y) - H(Y)$.

Similarly, we can define $H(X|\, Y, Z)$ or $H(X, Y\, |Z)$, for random variables X, Y, Z. In the proof of our Theorem 1 in Sect. 2.3, we will use the following well-known results about the entropy of random variables. They are more or less easy to deduce from the previous definitions, so we only include one of the proofs, as an illustrative example. The interested reader can consult [11] for more results on the entropy of random variables.

Lemma 1 *For all random variables X, Y, it holds $H(X) + H(Y) \geq H(X, Y) \geq H(X)$.*

Lemma 2 *For all random variables X, Y, Z, if $H(X|Y) = 0$, then $H(Z|X) \geq H(Z|Y)$.*

Proof First of all, it is easy to see that $H(Z|X) \geq H(Z|\, X, Y)$ and $H(Z, X\,|Y) \geq H(Z|Y)$, for any random variables X, Y, Z. Now we have

$$H(Z|X) \geq H(Z|\, X, Y) = H(Z, X, Y) - H(X, Y) = H(Z, X, Y) - \big(H(Y) + H(X|Y)\big) \overset{(*)}{=}$$
$$= H(Z, X, Y) - H(Y) = H(Z, X\,|Y) \geq H(Z|Y),$$

where we have used in $(*)$ the fact that $H(X|Y) = 0$. □

Lemma 3 *For all random variables X, Y, if $H(X|Y) = H(X)$, then $H(X, Y) = H(X) + H(Y)$.*

Lemma 4 *For all random variables X, Y, if $H(X|Y) = 0$, then $H(X, Y) = H(Y)$.*

2.2 Standard secret sharing schemes

The idea of *secret sharing schemes* was independently introduced by Shamir [28] and Blakley [3]. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n players. In this set of players, a family of authorized or qualified subsets $\Gamma \subset 2^{\mathcal{P}}$ is defined. This family is called the *access structure* of the scheme, and it must be monotone increasing; that is, if $A_1 \in \Gamma$ and $A_1 \subset A_2 \subset \mathcal{P}$, then $A_2 \in \Gamma$.

Given a monotone increasing access structure Γ and a secret to be shared, the idea behind a secret sharing scheme is that each player $P_i \in \mathcal{P}$ receives during the distribution phase a share sh_i of a secret $s \in \mathcal{K}$, where $\mathcal{K}$ is the space of possible secrets. Later, during the reconstruction phase, the secret can be univocally recovered from the shares of any authorized subset, $A \in \Gamma$. On the other hand, from the shares of a non-authorized subset, out of Γ, no information about the secret should be obtained. These two requirements (correctness and unconditional security) can be formalized by using information-theoretic tools such as the entropy of a random variable. Namely, if we use notation S for the random variable associated to the secret, SH_i for the random variable associated to the share of player P_i, and more generally SH_A for the (vector) random variable associated to the shares of players in $A \subset \mathcal{P}$, the two required properties for a *perfect* secret sharing scheme become: (1) $H(S|\mathsf{SH}_A) = 0$ for any subset $A \in \Gamma$, and (2) $H(S|\mathsf{SH}_B) = H(S)$ for any subset $B \notin \Gamma$.

Shamir proposed in [28] a *threshold* scheme, where subsets that can recover the secret are those with at least t members (t is the threshold); in other words, the access structure is $\Gamma = \{A \subset \mathcal{P} : |A| \geq t\}$. The set $\mathcal{K}$ of possible secrets is a finite field. To share a secret $s \in \mathcal{K}$, a random polynomial $f(x) \in \mathcal{K}[x]$ with degree at most $t - 1$ is chosen, such that $f(0) = s$. The share received by every player $P_i \in \mathcal{P}$ is $\mathsf{sh}_i = f(\alpha_i)$, where α_i are non-zero, pairwise different and publicly known elements of $\mathcal{K}$. Any subset of t or more shares allow recovery of the secret by polynomial interpolation, whereas any set of less than t shares give no information at all about the secret s. We will denote as $T(t, n)$ such a threshold access structure.

2.3 Syntax and security of multi-secret sharing schemes

Blundo *et al.* introduced in [4] the notion of *MSSSs*: ℓ secrets $s_1, \ldots, s_\ell \in \mathcal{K}$ are distributed at the same time between a set $\mathcal{P}$ of n players, according to ℓ access structures $\Gamma_1, \ldots, \Gamma_\ell \subset 2^{\mathcal{P}}$. Again, sh_i denotes the share of secret information received by each player P_i in the

distribution phase. The reconstruction phase takes as input a subset of shares and an index $j \in \{1, 2, \ldots, \ell\}$, and the expected output is the secret s_j. In [4], two requirements are defined for MSSSs, one related to correctness and one related to information-theoretic privacy.

1. *Correctness* If the reconstruction phase takes a subset of shares $\{(i, \mathsf{sh}_i)\}_{P_i \in A}$ and an index j as inputs, and $A \in \Gamma_j$, then the recovered secret is actually s_j. In other words, $H(S_j|\mathsf{SH}_A) = 0$ for any subset $A \in \Gamma_j$.
2. *Strong information-theoretic security* From the knowledge of a non-authorized subset of shares $\{(i, \mathsf{sh}_i)\}_{P_i \in B}$, with $B \notin \Gamma_j$, and of some secrets, different from s_j, the information obtained on the secret s_j is the same as if the shares $\{(i, \mathsf{sh}_i)\}_{P_i \in B}$ were not known. In the entropy language: for any subset $B \notin \Gamma_j$ and any subset $T \subset \{S_1, \ldots, S_\ell\} \backslash \{S_j\}$, it holds $H(S_j|\mathsf{SH}_B, T) = H(S_j|T)$.

This strong security requirement has an impact on the efficiency of MSSSs. Blundo et al. give in [4] lower bounds for the size of the shares sh_i in such a strongly secure MSSS. In particular, for the case where all the access structures are threshold, that is $\Gamma_j = \{A \subset \mathcal{P} : |A| \geq t_j\}$ and $1 \leq t_1 < t_2 < \cdots < t_\ell \leq n$, Blundo et al. proved that the entropy $H(\mathsf{SH}_i)$ of each individual share sh_i must be greater than or equal to the entropy $H(S)$ of the global secret $S = (S_1, \ldots, S_\ell)$, in any MSSS satisfying this strong security condition. This means that running ℓ independent instances of Shamir's threshold secret sharing scheme gives an optimal MSSS for the threshold case, if strong information-theoretic security is required.

Other works [20,25] consider a weaker (but maybe more realistic in actual applications of secret sharing) security notion for MSSSs, which does not consider the possibility that the adversary obtains some other subset T of secrets.

– *Weak information-theoretic security* No information at all on the secret s_j can be obtained from a non-authorized subset of shares $\{(i, \mathsf{sh}_i)\}_{P_i \in B}$, with $B \notin \Gamma_j$. In the entropy language: for any subset $B \notin \Gamma_j$, it holds $H(S_j|\mathsf{SH}_B) = H(S_j)$.

Masucci gives in [25] lower bounds for the size of the shares sh_i in such weakly information-theoretically secure MSSSs. However, these bounds do not lead to any result for the case of multi-secret sharing with threshold access structures. Therefore, according to the results that we have up to now, it may still be possible to design a multi-secret sharing scheme for the threshold case which enjoys weak information-theoretic security, and where the share of some participant is shorter than the secret. However, we prove below that this cannot be the case.

Theorem 1 *Let $\Gamma_1, \ldots, \Gamma_\ell \subset 2^{\mathcal{P}}$ be ℓ access structures, and let $P_i \in \mathcal{P}$. Assume there exist subsets of players $B_1 \subset B_2 \subset \cdots \subset B_\ell \subset \mathcal{P} - \{P_i\}$ satisfying, for all $j = 1, \ldots, \ell$, the following three conditions:*

(i) $B_j \in \Gamma_{j-1}$, *whenever* $j > 1$.
(ii) $B_j \notin \Gamma_j$,
(iii) $B_j \cup \{P_i\} \in \Gamma_j$,

Then, for any MSSS for $\Gamma_1, \ldots, \Gamma_\ell$ with weak unconditional security, it holds $H(\mathsf{SH}_i) \geq \sum_{j=1}^{\ell} H(S_j)$.

Proof First of all, by combining conditions (ii) and (iii) in the Theorem, and Lemmas 3 and 4, we have that, for all $j = 1, \ldots, \ell$, it holds

$$H(\mathsf{SH}_i,\ \mathsf{SH}_{B_j}) - H(\mathsf{SH}_{B_j}) = H(\mathsf{SH}_i,\ \mathsf{SH}_{B_j},\ S_j) + H(S_j) - H(\mathsf{SH}_{B_j},\ S_j) \qquad (1)$$

Now, for all $j = 1, \ldots, \ell - 1$, we have

$$\begin{aligned} H(\mathsf{SH}_i,\, \mathsf{SH}_{B_j},\, S_j) &= H(\mathsf{SH}_{B_j},\, S_j) + H(\mathsf{SH}_i \mid \mathsf{SH}_{B_j},\, S_j) \\ &\geq H(\mathsf{SH}_{B_j},\, S_j) + H(\mathsf{SH}_i \mid \mathsf{SH}_{B_{j+1}},\, S_j) \\ &= H(\mathsf{SH}_{B_j},\, S_j) + H(\mathsf{SH}_i,\, \mathsf{SH}_{B_{j+1}},\, S_j) - H(\mathsf{SH}_{B_{j+1}},\, S_j) \\ &= H(\mathsf{SH}_{B_j},\, S_j) + H(\mathsf{SH}_i,\, \mathsf{SH}_{B_{j+1}},\, S_j) - H(\mathsf{SH}_{B_{j+1}}) \\ &\geq H(\mathsf{SH}_{B_j},\, S_j) + H(\mathsf{SH}_i,\, \mathsf{SH}_{B_{j+1}}) - H(\mathsf{SH}_{B_{j+1}}). \end{aligned}$$

The first inequality is deduced by applying Lemma 2 to SH_{B_j} and $\mathsf{SH}_{B_{j+1}}$, because $B_j \subset B_{j+1}$. The last inequality is deduced by applying Lemma 1. Furthermore, we have applied Lemma 4 to deduce that $H(\mathsf{SH}_{B_{j+1}}, S_j) = H(\mathsf{SH}_{B_{j+1}})$, because $B_{j+1} \in \Gamma_j$.

Plugging the inequality into Eq. 1, we obtain

$$H(\mathsf{SH}_i, \mathsf{SH}_{B_j}) - H(\mathsf{SH}_{B_j}) \geq H(S_j) + H(\mathsf{SH}_i, \mathsf{SH}_{B_{j+1}}) - H(\mathsf{SH}_{B_{j+1}}) \tag{2}$$

for all $j = 1, \ldots, \ell - 1$.

For $j = \ell$, we have $H(\mathsf{SH}_i, \mathsf{SH}_{B_\ell}) = H(\mathsf{SH}_i, \mathsf{SH}_{B_\ell}, S_\ell)$, because $B_\ell \cup \{P_i\} \in \Gamma_\ell$. This equality can be rewritten as $H(\mathsf{SH}_{B_\ell}) + H(\mathsf{SH}_i \mid \mathsf{SH}_{B_\ell}) = H(\mathsf{SH}_{B_\ell}) + H(\mathsf{SH}_i, S_\ell \mid \mathsf{SH}_{B_\ell})$. But now we can use that $H(\mathsf{SH}_i, S_\ell \mid \mathsf{SH}_{B_\ell}) \geq H(S_\ell \mid \mathsf{SH}_{B_\ell}) = H(S_\ell)$, because $B_\ell \notin \Gamma_\ell$. Putting all together, we conclude that $H(\mathsf{SH}_i, \mathsf{SH}_{B_\ell}) \geq H(\mathsf{SH}_{B_\ell}) + H(S_\ell)$, which we rewrite as

$$H(\mathsf{SH}_i, \mathsf{SH}_{B_\ell}) - H(\mathsf{SH}_{B_\ell}) \geq H(S_\ell). \tag{3}$$

Now we can use $H(\mathsf{SH}_i) + H(\mathsf{SH}_{B_1}) \geq H(\mathsf{SH}_i, \mathsf{SH}_{B_1})$, by Lemma 1, to start a chain of inequalities, where we apply inequality 2 for $j = 1, \ldots, \ell - 1$, and finally inequality 3, to obain the desired result:

$$\begin{aligned} H(\mathsf{SH}_i) &\geq H(\mathsf{SH}_i, \mathsf{SH}_{B_1}) - H(\mathsf{SH}_{B_1}) \overset{(2)}{\geq} H(S_1) + H(\mathsf{SH}_i, \mathsf{SH}_{B_2}) - H(\mathsf{SH}_{B_2}) \\ &\overset{(2)}{\geq} \cdots \overset{(2)}{\geq} \sum_{j=1}^{\ell-1} H(S_j) + H(\mathsf{SH}_i, \mathsf{SH}_{B_\ell}) - H(\mathsf{SH}_{B_\ell}) \overset{(3)}{\geq} \sum_{j=1}^{\ell} H(S_j). \end{aligned}$$

□

Corollary 1 *For any MSSS, for access structures defined by thresholds $1 \leq t_1 < t_2 < \cdots < t_\ell \leq n$, that enjoys weak information-theoretic security, it holds $H(\mathsf{SH}_i) \geq \sum_{j=1}^{\ell} H(S_j)$, for any player $P_i \in \mathcal{P}$.*

Proof Just apply the previous theorem to a sequence of subsets $B_1 \subset B_2 \subset \cdots B_\ell \subset \mathcal{P} - \{P_i\}$ such that $|B_j| = t_j - 1$, for all $j = 1, 2, \ldots, \ell$. □

This means that the optimal MSSS for the threshold case with weak unconditional security, in terms of the ratio between the length of shares and the length of the global secret, is equivalent to running ℓ independent instances of Shamir's secret sharing scheme.

We stress that the three conditions in the statement of Theorem 1 imply that all the access structures must be different. If there was some repeated access structure, but the non-repeated ones did still satisfy these three conditions, then some variations of the theorem could be easily proved. For instance, if $\Gamma_1 = \Gamma_2$ but the rest of access structures satisfy the conditions, then we can ensure that $H(\mathsf{SH}_i) \geq \sum_{j=2}^{\ell} H(S_j)$ for all player $P_i \in \mathcal{P}$.

As an explicit example where Theorem 1 cannot be applied, and where $H(\mathsf{SH}_i) < \sum_{j=1}^{\ell} H(S_j)$, let us consider the case of ℓ threshold access structures with the same threshold:

$\Gamma_j = T(t,n)$ for all $j \in \{1,\ldots,\ell\}$. We can share the global secret $\mathbf{s} = (s_1,\ldots,s_\ell) \in (\mathbb{Z}_p)^\ell$ by following the ideas proposed in [12], provided $\ell \leq t(n-t)$. For some big prime number p, there are ℓ values $x_{j,0} \in \mathbb{Z}_p$, for $j \in \{1,\ldots,\ell\}$, assigned to the secrets, and there are $n-t$ values $x_{i,k}$ assigned to player P_i, for $k \in \{1,\ldots,n-t\}$. All these values must be pairwise different and public. To distribute the secret $\mathbf{s}$, a random polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree at most $t(n-t)-1$ is chosen, such that $f(x_{j,0}) = s_j$ for all $j \in \{1,\ldots,\ell\}$. Player P_i receives the share $\mathsf{sh}_i = (f(x_{i,1}),\ldots,f(x_{i,n-t})) \in (\mathbb{Z}_p)^{n-t}$. If t players work together, they can interpolate the polynomial $f(x)$ at any point and recover any of the secrets, because they hold $t(n-t)$ evaluations of the polynomial, which has degree at most $t(n-t)-1$. If less than t players cooperate, they obtain no information on any secret s_j, for $j \in \{1,\ldots,\ell\}$, and so the scheme enjoys weak unconditional security. If $n-t < \ell$, the length of each sh_i is strictly smaller than the length of the global secret $\mathbf{s}$.

Anyway, for the applications of multi-secret sharing that we have in mind, we are looking for efficient ways of sharing ℓ secrets for ℓ different access structures, in particular for the threshold case. The result in Theorem 1, although very interesting from a theoretical point of view, is quite negative for our interests, and we thus move to the scenario of computationally secure multi-secret sharing. This is not a big problem, taking into account that our final goal is to use MSSSs as an ingredient to implement cryptographic primitives (multi-policy distributed decryption and signatures) whose security is going to be at most computational, in any case.

2.4 Computational security for multi-secret sharing schemes

In the setting of computational security, a multi-secret sharing scheme $\Omega = (\Omega\cdot\mathsf{Stp}, \Omega\cdot\mathsf{Dist}, \Omega\cdot\mathsf{Rec})$ consists of three protocols. The setup protocol takes as input a security parameter $\lambda \in \mathbb{N}$, the set of players $\mathcal{P}$ and the ℓ access structures $\Gamma_1,\ldots,\Gamma_\ell$, and outputs some public and common parameters pms for the scheme (such as the access structures and set of players, mathematical groups, hash functions, etc.). We implicitly assume that pms also contains the descriptions of $\mathcal{P}$ and the access structures. We denote an execution of this protocol as $\mathsf{pms} \leftarrow \Omega\cdot\mathsf{Stp}\left(1^\lambda, \mathcal{P}, \{(j,\Gamma_j)\}_{1\leq j\leq \ell}\right)$.

The distribution protocol takes as input pms and the global secret $\mathbf{s} = (s_1,\ldots,s_\ell)$ to be distributed, and produces the set of shares $\{(i,\mathsf{sh}_i)\}_{P_i\in\mathcal{P}}$ and possibly some public output $\mathsf{out}_{\mathsf{pub}}$. We write $\left(\mathsf{out}_{\mathsf{pub}}, \{(i,\mathsf{sh}_i)\}_{P_i\in\mathcal{P}}\right) \leftarrow \Omega\cdot\mathsf{Dist}\,(\mathsf{pms},\mathbf{s})$.

The reconstruction protocol takes as input pms, $\mathsf{out}_{\mathsf{pub}}$, an index $j \in \{1,\ldots,\ell\}$, and the shares $\{(i,\mathsf{sh}_i)\}_{P_i\in A}$ of the players in some subset $A \subset \mathcal{P}$, and outputs a possible value $\tilde{s}_j$ for the j-th secret. We write $\tilde{s}_j \leftarrow \Omega\cdot\mathsf{Rec}\left(\mathsf{pms}, \mathsf{out}_{\mathsf{pub}}, j, \{(i,\mathsf{sh}_i)\}_{P_i\in A}\right)$.

For correctness, we require that, for any index $j \in \{1,\ldots,\ell\}$ and any subset $A \in \Gamma_j$, it holds

$$\Omega\cdot\mathsf{Rec}\left(\mathsf{pms}, \mathsf{out}_{\mathsf{pub}}, j, \{(i,\mathsf{sh}_i)\}_{P_i\in A}\right) = s_j,$$

if $\{(i,\mathsf{sh}_i)\}_{P_i\in A} \subset \{(i,\mathsf{sh}_i)\}_{P_i\in P}$ and $\left(\mathsf{out}_{\mathsf{pub}}, \{(i,\mathsf{sh}_i)\}_{P_i\in\mathcal{P}}\right) \leftarrow \Omega\cdot\mathsf{Dist}\,(\mathsf{pms},\mathbf{s})$ is a distribution of the secret $\mathbf{s} = (s_1,\ldots,s_j,\ldots,s_\ell)$, and the setup protocol has produced $\mathsf{pms} \leftarrow \Omega\cdot\mathsf{Stp}\left(1^\lambda, \mathcal{P}, \{(j,\Gamma_j)\}_{1\leq j\leq\ell}\right)$.

The computational security of a MSSS is defined by the following game $\mathcal{G}$ between a challenger and an adversary $\mathcal{A}_{\mathsf{MSS}}$. Although other (stronger) security games and notions could be considered, we have chosen the following one, which is the direct analogue of the standard notion of semantic security for encryption schemes.

1. The adversary $\mathcal{A}_{\mathsf{MSS}}$ publishes the set of players $\mathcal{P}$ and the ℓ access structures $\Gamma_1, \ldots, \Gamma_\ell \subset 2^{\mathcal{P}}$.
2. The challenger runs $\mathsf{pms} \leftarrow \Omega \cdot \mathsf{Stp}\left(1^\lambda, \mathcal{P}, \{(j, \Gamma_j)\}_{1 \le j \le \ell}\right)$ and sends pms to $\mathcal{A}_{\mathsf{MSS}}$.
3. The adversary $\mathcal{A}_{\mathsf{MSS}}$ broadcasts a subset $\widetilde{B} \subset \mathcal{P}$ of corrupted players.
4. $\mathcal{A}_{\mathsf{MSS}}$ broadcasts two different global secrets $\mathbf{s}^{(0)} \neq \mathbf{s}^{(1)}$ with the restriction:

$$s_j^{(0)} = s_j^{(1)}, \quad \forall j \in \{1, \ldots, \ell\} \quad \text{s.t. } \widetilde{B} \in \Gamma_j.$$

5. **[Challenge]** The challenger chooses at random a bit $b \in \{0, 1\}$, runs $\left(\mathsf{out}_{\mathsf{pub}}, \{(i, \mathsf{sh}_i)\}_{P_i \in \mathcal{P}}\right) \leftarrow \Omega \cdot \mathsf{Dist}\left(\mathsf{pms}, \mathbf{s}^{(b)}\right)$ and sends $\left(\mathsf{out}_{\mathsf{pub}}, \{(i, \mathsf{sh}_i)\}_{P_i \in \widetilde{B}}\right)$ to $\mathcal{A}_{\mathsf{MSS}}$.
6. Finally, $\mathcal{A}_{\mathsf{MSS}}$ outputs a bit b'.

The advantage of $\mathcal{A}_{\mathsf{MSS}}$ in breaking the multi-secret sharing scheme Ω is defined as

$$\mathbf{Adv}_{\mathcal{A}_{\mathsf{MSS}}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

The scheme Ω is said to enjoy computational security if $\mathbf{Adv}_{\mathcal{A}_{\mathsf{MSS}}}(\lambda)$ is a negligible function in λ, for any polynomial-time adversary $\mathcal{A}_{\mathsf{MSS}}$. We recall that a function $f(k)$ is negligible in k if there exist a polynomial $p(\cdot)$ and a value $k_0 \in \mathbb{N}$ such that $f(k) \le 1/p(k)$ for any $k \ge k_0$).

3 A computationally secure multi-secret sharing scheme

We introduce in this section a computationally secure multi-secret sharing scheme with provable security in the random oracle model. The new scheme is very similar to some previous schemes [17,23], which however did not have a formal security proof. Although we describe and analyze the scheme in the setting of different threshold access structures, it can be easily extended to work with more general access structures (see Appendix B).

After the description and analysis of this scheme, we will use it as a building block in the design of new multi-policy signature and decryption schemes, in Sects. 4 and 5.

3.1 The new scheme

Setup: $\Omega \cdot \mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of n users and let $1 \le t_1 < t_2 < \cdots < t_\ell \le n$ be the ℓ different thresholds that define the access structures $\{\Gamma_j\}_{1 \le j \le \ell}$. A prime number $p > n$ is chosen, such that p is at least λ bits long. A hash function $H : \mathbb{N} \times \mathbb{Z}_p^\star \to \mathbb{Z}_p$ is also chosen. Each player P_i is assigned the value i. The public parameters are $\mathsf{pms} = (p, H, \mathcal{P}, t_1, \ldots, t_\ell)$.

Distribution of the shares: $\Omega \cdot \mathsf{Dist}(\mathsf{pms}, \mathbf{s})$.

The secret to be distributed is $\mathbf{s} = (s_1, \ldots, s_\ell) \in (\mathbb{Z}_p)^\ell$. For simplicity, we assume the distribution is done by an external dealer; see Appendix A for a discussion on how the own members of $\mathcal{P}$ could run this protocol.

1. Choose random values $\mathsf{sh}_i \in \mathbb{Z}_p^\star$, pairwise different for $i = 1, 2, \ldots, n$, as the secret shares.
2. Choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree at most $t_j - 1$, for $j = 1, \ldots, \ell$, such that $f_j(0) = s_j$.

3. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \ell$, compute the values $h_{ij} = H(j, sh_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$.
4. The secret share sh_i is sent to player P_i via a secure channel, whereas the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{(i, j, r_{ij})\}_{P_i \in \mathcal{P}, j \in \{1,\ldots,\ell\}}$.

Reconstruction of the secrets: $\Omega \cdot \mathsf{Rec}(\mathsf{pms}, \mathsf{out}_{\mathsf{pub}}, j, \{(i, \mathsf{sh}_i)\}_{P_i \in A})$.

When the players of an authorized subset $A \in \Gamma_j$ (i.e. $|A| \geq t_j$) want to recover the secret s_j, they must cooperate performing the following steps.

1. Each player $P_i \in A$ computes his pseudo secret share as $h_{ij} = H(j, \mathsf{sh}_i)$.
2. Take the values $\{(i, j, r_{ij})\}_{P_i \in A}$ from $\mathsf{out}_{\mathsf{pub}}$ and compute $f_j(i) = r_{ij} + h_{ij} \bmod p$, for every $P_i \in A$.
3. Use the values $\{(i, f_j(i))\}_{P_i \in A}$ to interpolate the polynomial $f_j(x)$ and recover the j-secret $s_j = f_j(0)$.

Note that the correctness of the proposed scheme holds directly via interpolation.

3.2 Security analysis

In this section we prove the computational security of the proposed scheme, assuming that the hash function H behaves as a random oracle [2].

Theorem 2 *For any adversary $\mathcal{A}_{MSS}$ against the described threshold MSS scheme that makes at most q_H queries to the random oracle for H, we have* $\mathbf{Adv}_{\mathcal{A}_{MSS}}(\lambda) \leq \frac{q_H(q_H+n)}{2^{\lambda+1}} + o\left(\left(\frac{q_H(q_H+n)}{2^{\lambda+1}}\right)^2\right)$.

Proof Let $\mathcal{A}_{\mathsf{MSS}}$ be an adversary against the computational security of the MSSS. We act as the challeger of the security game described in Sect. 2.4. $\mathcal{A}_{\mathsf{MSS}}$ starts the game by choosing the set of users $\mathcal{P} = \{P_i\}_{1 \leq i \leq n}$ and the threshold access structures $\{\Gamma_j\}_{1 \leq j \leq \ell}$ with $\Gamma_j = T(t_j, n)$. We then run $\mathsf{pms} \leftarrow \Omega \cdot \mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$ and send $\mathsf{pms} = (p, H, \mathcal{P}, t_1, \ldots, t_\ell)$ to $\mathcal{A}_{\mathsf{MSS}}$, who chooses a subset $\widetilde{B} \subset \mathcal{P}$ of corrupted players with $|\widetilde{B}| = t^\star$. Let $\mathbb{J}^\star = \{j \in \{1, \ldots, \ell\} \text{ s.t. } t_j \leq t^\star\}$.

We choose random pairwise different elements $\mathsf{sh}_i \in \mathbb{Z}_p$, for $P_i \in \mathcal{P}$. If $\mathcal{A}_{\mathsf{MSS}}$ makes a hash query (j, x) to the random oracle such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and $x \in \{\mathsf{sh}_i\}_{P_i \notin \widetilde{B}}$, then we abort the game. Otherwise, the query is answered by choosing a random element $h \in \mathbb{Z}_p$, storing the relation $H(j, x) = h$ in a hash table, and sending back the output h to $\mathcal{A}_{\mathsf{MSS}}$. If a hash query (j, x) by $\mathcal{A}_{\mathsf{MSS}}$ is already in the hash table, the stored value h is sent back to $\mathcal{A}_{\mathsf{MSS}}$.

Challenge At some point, $\mathcal{A}_{\mathsf{MSS}}$ outputs two different multi-secrets $\mathbf{s}^{(0)} \neq \mathbf{s}^{(1)}$, such that $s_j^{(0)} = s_j^{(1)}$ for all $j \in \mathbb{J}^\star$. We choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree at most $t_j - 1$ and such that $f_j(0) = s_j^{(0)}$, for all $j \in \mathbb{J}^\star$. For those values of $j \in \mathbb{J}^\star$, we compute (via the hash-table procedure) the values $h_{ij} = H(j, \mathsf{sh}_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$, for all $P_i \in \mathcal{P}$.

For the rest of values $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$, we choose random pairs of polynomials $f_j^{(0)}(x), f_j^{(1)}(x) \in \mathbb{Z}_p[x]$ of degree at most $t_j - 1$, such that $f_j^{(0)}(0) = s_j^{(0)}$, $f_j^{(1)}(0) = s_j^{(1)}$, and $f_j^{(0)}(i) = f_j^{(1)}(i)$ for all corrupted players $P_i \in \widetilde{B}$. For indices i such that $P_i \in \widetilde{B}$, we compute (via the hash-table procedure) the values $h_{ij} = H(j, \mathsf{sh}_i)$ and $r_{ij} = f_j^{(0)}(i) - h_{ij} \bmod p$. For indices i such that $P_i \notin \widetilde{B}$, we choose at random $r_{ij} \in \mathbb{Z}_p$.

We give to $\mathcal{A}_{\mathsf{MSS}}$ the shares $\{(i, \mathsf{sh}_i)\}_{P_i \in \widetilde{B}}$ of the corrupted players, as well as the public output of the protocol $\mathsf{out}_{\mathsf{pub}} = \{(i, j, r_{ij})\}_{P_i \in \mathcal{P}, j \in \{1,\ldots,\ell\}}$.

For indices $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and indices i such that $P_i \notin \widetilde{B}$, let us define $h_{ij}^{(0)} = r_{ij} - f_j^{(0)}(i) \mathrm{mod} p$ and $h_{ij}^{(1)} = r_{ij} - f_j^{(1)}(i) \mathrm{mod} p$. We can choose at random a bit $\beta \in \{0, 1\}$ and include in the hash-table the values $H(j, \mathsf{sh}_i) = h_{ij}^{(\beta)}$, for all i, j such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and $P_i \notin \widetilde{B}$. In this way, we are perfectly simulating an execution of the distribution of shares for the secret $\mathbf{s}^{(\beta)} = (s_1^{(\beta)}, \ldots, s_\ell^{(\beta)})$. The key point here is that, as long as $\mathcal{A}_{\mathsf{MSS}}$ does not make any hash query $H(j, \mathsf{sh}_i)$ for indices i, j such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and $P_i \notin \widetilde{B}$, the information that $\mathcal{A}_{\mathsf{MSS}}$ gets is the same as if the shared secret was $\mathbf{s}^{(1-\beta)}$.

Final analysis Therefore, to compute the probability that $\mathcal{A}_{\mathsf{MSS}}$ guesses the correct shared secret, we distinguish between two cases, depending on whether $\mathcal{A}_{\mathsf{MSS}}$ makes a hash query $H(j, \mathsf{sh}_i)$ for indices i, j such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and $P_i \notin \widetilde{B}$. If this is the case, which happens with probability δ, then we assume the best case for $\mathcal{A}_{\mathsf{MSS}}$: he always guesses the correct secret in that case. On the other hand, if $\mathcal{A}_{\mathsf{MSS}}$ does not make such a hash query, which happens with probability $1 - \delta$, then the probability that $\mathcal{A}_{\mathsf{MSS}}$ guesses correctly is exactly $1/2$. Summing up, the probability that $\mathcal{A}_{\mathsf{MSS}}$ guesses the correct secret is at most $\delta + 1/2(1 - \delta)$. Therefore, the advantage of $\mathcal{A}_{\mathsf{MSS}}$ is

$$\mathbf{Adv}_{\mathcal{A}_{\mathsf{MSS}}}(\lambda) = \left| \Pr[\mathcal{A}_{\mathsf{MSS}} \text{ guesses}] - \frac{1}{2} \right| \leq \frac{1}{2}\delta(\lambda).$$

The probability $\delta(\lambda)$ that some of q_H randomly chosen elements falls in a perfectly hidden subset $\{\mathsf{sh}_i\}_{P_i \notin \widetilde{B}}$ of $n - t^\star$ random elements of $\mathbb{Z}_p$ can be bounded as

$$\delta(\lambda) < 1 - \left(\frac{p - (n + q_H - t^\star - 1)}{p} \right)^{q_H} = \frac{q_H(q_H + n)}{p} + o\left(\left(\frac{q_H(q_H + n)}{p} \right)^2 \right).$$

Using that $p > 2^\lambda$, we obtain the desired result $\mathbf{Adv}_{\mathcal{A}_{\mathsf{MSS}}}(\lambda) \leq \frac{q_H(q_H+n)}{2^{\lambda+1}} + o\left(\left(\frac{q_H(q_H+n)}{2^{\lambda+1}} \right)^2 \right)$. □

The dominant term in the expression stated in the previous theorem is q_H^2, because the number n of players will usually be small compared to q_H. In cryptography, the number of hash queries is usually estimated as $q_H \leq 2^{60}$. Therefore, the multi-secret sharing scheme described in this section satisfies a 80-bit security level (that is, $\mathbf{Adv}_{\mathcal{A}_{\mathsf{MSS}}}(\lambda) < 2^{-80}$) when $\lambda \geq 200$ or, equivalently, when $p \geq 2^{200}$.

3.3 Efficiency and comparison with other MSSS

In this section we analyze the efficiency of our new multi-secret sharing scheme, and we compare it with other schemes (with computational security, as well) in the literature that lead to secret shares with constant length. We stress that our scheme is the only one with a formal and complete security analysis. For simplicity, we focus on the case of threshold access structures for this efficiency analysis.

In our scheme, the cost of sharing a global secret in $(\mathbb{Z}_p)^\ell$ among n players, according to threshold access structures with thresholds $t_1 \leq t_2 \leq \ldots \leq t_\ell$, is (roughly) equivalent to $\sum_{j=1}^{\ell} t_j n$ operations in $\mathbb{Z}_p$. The public output contains $\ell \cdot n$ elements in $\mathbb{Z}_p$, and the secret share sh_i of each player contains a single element in $\mathbb{Z}_p$. The cost of recovering a particular secret s_j depends linearly on the needed threshold t_j.

Table 1 Comparison between some threshold MSSS with computational security.

	Cost of $\Omega \cdot \mathsf{Dist}$	Cost of $\Omega \cdot \mathsf{Rec}(s_j)$	Length of $\mathsf{out}_{\mathsf{pub}}$	Length of sh_i
Cachin's MSSS [9]	$\sum_{j=1}^{\ell} \binom{n}{t_j}$	$\mathcal{O}(t_j)$	$\left(\sum_{j=1}^{\ell} \binom{n}{t_j}\right) \lvert p \rvert$	$\lvert p \rvert$
Schemes in [17,23]	$\sum_{j=1}^{\ell} t_j n$	$\mathcal{O}(t_j)$	$n \cdot \ell \cdot \lvert p \rvert$	$\lvert p \rvert$
Our new scheme	$\sum_{j=1}^{\ell} t_j n$	$\mathcal{O}(t_j)$	$n \cdot \ell \cdot \lvert p \rvert$	$\lvert p \rvert$

The efficiency properties of the two schemes [17,23] are essentially the same. Indeed, these two schemes are very similar to our new scheme; roughly speaking, the scheme in [17] uses two-variable one-way functions and the scheme in [23] uses iterated one-way hash functions, instead of our hash function H. Again, we insist that the security analysis of the schemes in [17,23] is not complete or formal.

We include another MSSS in the comparison that we summarize in Table 1, the one proposed by Cachin in [9], which produces constant-length secret shares but, again, lacks a formal security analysis. The length of $\mathsf{out}_{\mathsf{pub}}$ in Cachin's scheme, as well as the cost of the distribution phase, depend linearly on the number of minimal authorized subsets[1] in each access structure; when the access structure is a threshold one, with threshold t, this number is $\binom{n}{t}$, very big. Cachin's MSS could be a good alternative for situations where all the access structures have few minimal authorized subsets.

Table 1 below summarizes the efficiency aspects of these MSSS, when applied to share ℓ secrets $s_1, \ldots, s_\ell$, each one in $\mathbb{Z}_p$, among n players, according to ℓ threshold access structures, for thresholds $t_1 \leq \cdots \leq t_\ell$.

4 Application to multi-policy distributed signatures

In this section, we deal with distributed signatures. Threshold (or distributed) signatures have received also a lot of attention from the cryptographic community [5,14,30]; they have applications in scenerios where the cooperation of more than one single entity is necessary to authenticate a message. In our multi-policy setting, we will have a set of users $\mathcal{P} = \{P_1, \ldots, P_n\}$ as the possible signers of messages. Depending on the content and the importancy of the message, more or less members of $\mathcal{P}$ can be required to participate in the signing process. In other words, the subset of real signers $A \subset \mathcal{P}$ will choose ad-hoc a signing policy Γ_j among a set of pre-defined different signing policies $\Gamma_1, \ldots, \Gamma_\ell$, such that $A \in \Gamma_j$, and will cooperate to sign the message on behalf of that policy Γ_j. Each user $P_i \in \mathcal{P}$ will have a share sh_i of secret information, and will use it to perform his part of the signing process. The final verification step will take as inputs the index j, the message, the signature, and the global public key of the set $\mathcal{P}$, in order to check the validity of the signature. Note that the knowledge of identities of the real signers (in subset A) is not necessary to verify a signature, which provides some kind of anonymity (if desired) to the process.

[1] Given an access structure $\Gamma \subset 2^{\mathcal{P}}$, a subset $A \in \Gamma$ is minimal authorized if $A - \{P_i\} \notin \Gamma$, for all $P_i \in A$.

After defining the syntactic definition and the security model for this primitive of multi-policy distributed signatures, we present a scheme for the case of threshold signing policies (that uses as a building block the MSSS in Sect. 3) and we prove its security. The scheme can be extended to work with more general access structures, as we discuss in Appendix B.

4.1 Syntactic definition

A multi-policy distributed signature scheme $\Theta = (\Theta \cdot \mathsf{St}, \Theta \cdot \mathsf{KG}, \Theta \cdot \mathsf{Sign}, \Theta \cdot \mathsf{Ver})$ consists of four probabilistic polynomial-time algorithms:

- The randomized *setup* algorithm $\Theta \cdot \mathsf{St}$ takes a security parameter $\lambda \in \mathbb{N}$ and outputs some public parameters params that will be common to all the users in the system: the mathematical groups, generators, hash functions, etc. We write $\mathsf{params} \leftarrow \Theta \cdot \mathsf{St}(1^\lambda)$ to denote an execution of this algorithm.
- The *key generation* algorithm $\Theta \cdot \mathsf{KG}$ for a collective $\mathcal{P} = \{P_1, \ldots, P_n\}$ of n users and ℓ different signature policies $\Gamma_j \subset 2^B$ for $j = 1, \ldots, \ell$ has as public output a public key PK that will be used in both the signing and verification steps. We implicitly assume that PK contains the description of $\mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell$. Each user $P_i \in \mathcal{P}$ receives a secret share sh_i. This key generation process for the collective $\mathcal{P}$ can be run either by a trusted third party, or by the users in $\mathcal{P}$ themselves. We will write $(\{(i, \mathsf{sh}_i)\}_{1 \le i \le n}, PK) \leftarrow \Theta \cdot \mathsf{KG}(\mathsf{params}, \mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell)$ to refer to this key generation protocol.
- The *joint signature* algorithm $\Theta \cdot \mathsf{Sign}$ is a distributed protocol run by some subset of users $A \subset \mathcal{P}$. The common inputs are params, PK, a message m, the secret shares sh_i of the users $P_i \in A$, and the index j of the desired signature policy Γ_j, where $j \in \{1, \ldots, \ell\}$. The outputs are a signature σ and the index j of the chosen signature policy. We write $(\sigma, j) \leftarrow \Theta \cdot \mathsf{Sign}(\mathsf{params}, PK, m, A, \{(i, \mathsf{sh}_i)\}_{P_i \in A}, j)$ to refer to an execution of this protocol.
- The *verification* algorithm $\Theta \cdot \mathsf{Ver}$ takes as input params, a message m, a signature (σ, j), and the public key PK of the intended receiver group $\mathcal{P}$. The output will be one if (σ, j) is a valid signature of m and zero otherwise. We denote an execution of this algorithm as $\{1, 0\} \leftarrow \Theta \cdot \mathsf{Ver}(\mathsf{params}, m, \sigma, j, PK)$.

For correctness, $\Theta \cdot \mathsf{Ver}(\mathsf{params}, m, \Theta \cdot \mathsf{Sign}(\mathsf{params}, PK, m, A, \{(i, \mathsf{sh}_i)\}_{P_i \in A}, j), PK) = 1$ is required, whenever $A \in \Gamma_j$ and the values $\mathsf{params}, \{(i, \mathsf{sh}_i)\}_{1 \le i \le n}, PK$ have been obtained by properly executing the protocols $\Theta \cdot \mathsf{St}$ and $\Theta \cdot \mathsf{KG}$.

4.2 Security model

A multi-policy distributed signature scheme must be *robust*. The robustness property holds when the protocols $\Theta \cdot \mathsf{KG}$ and $\Theta \cdot \mathsf{Sign}$ always complete successfully, even under the action of a polynomial-time adversary that is allowed to corrupt an unauthorized set of users.

As any other primitive related to signatures, a multi-policy distributed signature scheme Θ must also be *unforgeable*. That is, any polynomial-time adversary that is allowed to corrupt a subset of users $\widetilde{B} \subset \mathcal{P}$ such that $\widetilde{B} \notin \Gamma_j$ must have negligible probability of success in producing a new valid signature for some message with respect to signing policy Γ_j, even if this adversary has access to a signing oracle for messages and signing policies of his choice. This property is known as *unforgeability against chosen message attacks* (UNF security, for short) and is defined, for a security parameter $\lambda \in \mathbb{N}$, by considering the following game that an attacker $\mathcal{A}_{\mathsf{UNF}}$ plays against a challenger:

1. The challenger runs $\mathsf{params} \leftarrow \Theta \cdot \mathsf{St}(1^\lambda)$ and gives params to $\mathcal{A}_{\mathsf{UNF}}$.

2. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of users, ℓ different signature policies $\Gamma_j \subset 2^{\mathcal{P}}$, for $j = 1, \ldots, \ell$, and a subset $\widetilde{B} \subset \mathcal{P}$ of users, to be corrupted. The challenger runs $(\{(i, \mathsf{sh}_i)\}_{1 \le i \le n}, PK) \leftarrow \Theta \cdot \mathsf{KG}(\mathsf{params}, \mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell)$ and gives to $\mathcal{A}_{\mathsf{UNF}}$ the values PK and $\{(i, \mathsf{sh}_i)\}_{P_i \in \widetilde{B}}$. We consider only *static* adversaries who choose the subset $\widetilde{B}$ of corrupted users at the beginning of the attack.
3. [**Queries**] $\mathcal{A}_{\mathsf{UNF}}$ can make adaptive queries to a distributed signing oracle for the target set $\mathcal{P}$: $\mathcal{A}_{\mathsf{UNF}}$ sends a tuple (m, j) for the signature policy Γ_j. The challenger runs the distributed signature algorithm $(\sigma, j) \leftarrow \Theta \cdot \mathsf{Sign}(\mathsf{params}, m, A, \{(i, \mathsf{sh}_i)\}_{P_i \in A}, j)$ for an authorized subset $A \in \Gamma_j$. The attacker $\mathcal{A}_{\mathsf{UNF}}$ must be given all the information that corrupted players (in $\widetilde{B}$) would obtain during the execution of this protocol $\Theta \cdot \mathsf{Sign}$, including the final signature and all the broadcast information.
4. At some point, $\mathcal{A}_{\mathsf{UNF}}$ outputs a forgery $(j^\star, m^\star, \sigma^\star)$. We say that $\mathcal{A}_{\mathsf{UNF}}$ is successful if: (1) $\widetilde{B} \notin \Gamma_j$, (2) $\Theta \cdot \mathsf{Ver}(\mathsf{params}, m^\star, \sigma^\star, j^\star, PK) = 1$, and (3) $(j^\star, m^\star, \sigma^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ in a signature query (step 3).

The advantage of such a (static) adversary $\mathcal{A}_{\mathsf{UNF}}$ in breaking the UNF security of the multi-policy distributed signature scheme is defined as the probability that $\mathcal{A}_{\mathsf{UNF}}$ is successful in the game above.

A multi-policy distributed signature scheme Θ is UNF secure if the advantage of any such polynomial-time (static) adversary $\mathcal{A}_{\mathsf{UNF}}$ is a negligible function of the security parameter λ.

4.3 A new multi-policy signature scheme

We propose here a new multi-policy distributed signature scheme, that we describe (for simplicity) for the case where all the signing policies are threshold ones: $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$, where $1 \le t_1 < t_2 < \cdots < t_\ell \le n$. See Appendix B for an extension of this scheme to the case of more general access structures. The scheme is inspired by the (single) threshold signature scheme proposed by Boldyreva in [5] (which is itself inspired by the individual signature scheme of Boneh–Lynn–Shacham [7]). A key ingredient in the design of the new scheme will be the MSSS proposed and analyzed in Sect. 3. We also need to introduce the notion of Gap Diffie–Hellman groups, which is related to the Diffie–Hellman problems.

Given a security parameter $\lambda \in \mathbb{N}$, let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p, such that p is λ bits long. The *Computational Diffie–Hellman (CDH, for short) problem* can be defined as the problem of computing the value g^{ab} on input the values (g, g^a, g^b), for random elements $a, b \in \mathbb{Z}_p^*$. The *Computational Diffie–Hellman Assumption* states that the CDH problem is hard to solve. A bit more formally, for any polynomial-time algorithm $\mathcal{A}_{\mathsf{CDH}}$ that receives as input $\mathbb{G}, g, g^a, g^b$, for random elements $a, b \in \mathbb{Z}_p^*$, we can define as $\mathsf{Adv}_{\mathcal{A}_{\mathsf{CDH}}}(\lambda)$ the probability that $\mathcal{A}_{\mathsf{CDH}}$ outputs the value g^{ab}. The *Computational Diffie–Hellman Assumption* states that $\mathtt{Adv}_{\mathcal{A}_{\mathsf{CDH}}}(\lambda)$ is negligible in λ.

The (easier) *Decisional Diffie–Hellman (DDH) problem* tries to decide whether the four group elements (g, g^a, g^b, h) are all random or they are a valid Diffie–Hellman tuple, that is $h = g^{ab}$. Groups where the CDH problem is hard to solve but the DDH problem is easy are called *Gap Diffie–Hellman (GDH) groups*. See [6,21,26] for more details on GDH groups; up to now, the only known GDH groups are related to bilinear pairings on elliptic curves.

The protocols of the new multi-policy signature scheme Θ work as follows.

Setup $\Theta \cdot \mathsf{St}(1^\lambda)$.

Given a security parameter $\lambda \in \mathbb{N}$, a GDH group $\mathbb{G} = \langle g \rangle$ of prime order p, such that p is λ bits long, is chosen. Two hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_1 : \{0,1\}^* \to \mathbb{G}$ are chosen. The output of this protocol is $\mathsf{params} = (p, \mathbb{G}, g, H_0, H_1)$.

Key generation: $\Theta \cdot \mathsf{KG}(\mathsf{params}, \mathcal{P}, t_1, \ldots, t_\ell, n)$.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n users and let $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$ be the threshold signing policies defined on $\mathcal{P}$, where $1 \leq t_1 < t_2 < \cdots < t_\ell \leq n$.

For $j = 1, \ldots, \ell$, the value $PK_j = g^{s_j}$ is computed, for a random value $s_j \in \mathbb{Z}_p^*$ that will remain unknown to the members of $\mathcal{P}$. These ℓ secret values will correspond to a secret vector $\mathbf{s} = (s_1, \ldots, s_\ell)$ of the multi-secret sharing scheme described in Sect. 3, that will be shared by running the distribution protocol $\Omega \cdot \mathsf{Dist}(\mathcal{P}, t_1, \ldots, t_\ell, \mathbf{s})$, with hash function H_0:

1. Choose random values $\mathsf{sh}_i \in \mathbb{Z}_p^\star$, pairwise different for $i = 1, 2, \ldots, n$, as the secret shares.
2. Choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree at most $t_j - 1$, for $j = 1, \ldots, \ell$, such that $f_j(0) = s_j$.
3. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \ell$, compute the values $h_{ij} = H_0(j, \mathsf{sh}_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$. Compute the public verification values $D_{ij} = g^{h_{ij} + r_{ij}}$.
4. The secret share sh_i is sent to player P_i via a secure channel, whereas the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{(i, j, r_{ij})\}_{P_i \in \mathcal{P}, j \in \{1, \ldots, \ell\}}$.

The global public key is $PK = (PK_1, \ldots, PK_\ell, \mathsf{out}_{\mathsf{pub}}, \{(i, j, D_{ij})\}_{P_i \in \mathcal{P}, 1 \leq j \leq \ell})$, whereas the secret share for each player P_i is sh_i.

Joint signature: $\Theta \cdot \mathsf{Sign}(\mathsf{params}, PK, m, A, \{(i, \mathsf{sh}_i)\}_{P_i \in A}, j)$.

Let $A \subset \mathcal{P}$ be a subset of users in $\mathcal{P}$ that want to cooperate to sign a message m with respect to a signing policy $\Gamma_j = T(t_j, n)$ for which they form an authorized subset, $A \in \Gamma_j$. Members of A proceed as follows:

1. Each $P_i \in A$ computes $h_{ij} = H_0(j, \mathsf{sh}_i)$, recovers r_{ij} from $\mathsf{out}_{\mathsf{pub}}$ and broadcasts his signature share $\sigma_{ij} = H_1(m, j)^{h_{ij} + r_{ij}} \in \mathbb{G}$.
2. The rest of members of A verify if $(g, D_{ij}, H_1(m, j), \sigma_{ij})$ is a valid Diffie–Hellman tuple.
3. If there are not t_j valid signature shares (i, σ_{ij}), then stop and output $\perp$. Otherwise, from t_j valid signature shares $\{(i, \sigma_{ij})\}_{P_i \in A}$, one can consider the Lagrange interpolation coefficients $\lambda_{ij}^A \in \mathbb{Z}_p$ such that $s_j = f_j(0) = \sum_{P_i \in A} \lambda_{ij}^A \cdot f_j(i)$.
4. Return the resulting signature and index (σ, j), where $\sigma = \prod_{P_i \in A} \sigma_{ij}^{\lambda_{ij}^A}$.

Verification: $\Theta \cdot \mathsf{Ver}(\mathsf{params}, m, \sigma, j, PK)$

In the verification step it is enough to check if $(g, PK_j, H_1(m, j), \sigma)$ is a valid Diffie–Hellman tuple. Return 1 if this is the case, or 0 otherwise.

4.4 Security analysis

The multi-policy threshold signature scheme described in the previous section, with a trusted dealer in charge of the key generation phase, is trivially robust: during the joint signature generation phase, cheating players are detected in step 2 and rejected from the protocol. Assuming the remaining players are enough (i.e. they are at least t_j), the signing protocol finishes correctly. One way to ensure this is by requiring that an adversary can corrupt at most $n - t_\ell$ players.

Regarding unforgeability, we now prove that the proposed scheme is UNF secure provided the Computational Diffie–Hellman (CDH) problem is hard in the GDH group $\mathbb{G}$. The proof is in the random oracle model for hash functions $H_0,\ H_1$.

Theorem 3 *In the random oracle model, the scheme Θ is UNF secure, assuming the Computational Diffie–Hellman problem is hard to solve in the GDH group $\mathbb{G}$.*

Proof The proof is by reduction, assuming that hash functions $H_0,\ H_1$ are modeled as random oracles. An adversary $\mathcal{A}_{\mathsf{UNF}}$ that has non-negligible success in forging a new valid signature is used to construct an algorithm $\mathcal{A}_{\mathsf{CDH}}$ that solves the CDH problem in $\mathbb{G}$.

$\mathcal{A}_{\mathsf{CDH}}$ receives as input $(g,\ g^a,\ g^b)$, where $\mathbb{G} = \langle g \rangle$ is a GDH group of prime order p. The goal of $\mathcal{A}_{\mathsf{CDH}}$ is to compute g^{ab}. The algorithm $\mathcal{A}_{\mathsf{CDH}}$ initializes the attacker $\mathcal{A}_{\mathsf{UNF}}$ by giving $\mathsf{params} = (p, \mathbb{G}, g, H_0, H_1)$ to him. Since the hash functions $H_0,\ H_1$ are supposed to behave as random oracles, $\mathcal{A}_{\mathsf{CDH}}$ will create and maintain tables TAB_0 and TAB_1 to answer the hash queries from $\mathcal{A}_{\mathsf{UNF}}$. These answers are produced by $\mathcal{A}_{\mathsf{CDH}}$ by first checking if there already exists an entry in the corresponding table for the input of the hash query; if so, $\mathcal{A}_{\mathsf{CDH}}$ responds with the existing output; otherwise, $\mathcal{A}_{\mathsf{CDH}}$ chooses a new random output, adds the new relation input–output to the corresponding table, and responds to $\mathcal{A}_{\mathsf{UNF}}$ with this output value. Hash queries $(m,\ j)$ to H_1 are answered in the following way. Let q_1 be the maximum number of such H_1 queries. $\mathcal{A}_{\mathsf{CDH}}$ chooses at random an index $k^\star \in \{1, \ldots, q_1\}$ for a special query $(\tilde{m},\ \tilde{j})$. For this special query, $\mathcal{A}_{\mathsf{CDH}}$ chooses a random value $\tilde{\beta} \in \mathbb{Z}_p$ and defines the relation $H_1(\tilde{m},\ \tilde{j}) = (g^a)^{\tilde{\beta}}$. For the rest of H_1 queries $(m,\ j)$, $\mathcal{A}_{\mathsf{CDH}}$ chooses a random value $\beta \in \mathbb{Z}_p$ and defines the relation $H_1(m,\ j) = g^{\beta}$. These relations are stored in TAB_1.

Key distribution $\mathcal{A}_{\mathsf{UNF}}$ chooses the target collective $\mathcal{P}^\star = \{P_1, \ldots, P_n\}$, the decryption policies $\Gamma_j = T(t_j, n) \subset 2^B$ for $j = 1, \ldots, \ell$ where $t_1 < t_2 < \cdots < t_\ell$, and also the subset of corrupted members $\widetilde{B} \subset \mathcal{P}^\star$. For simplicity, we assume $\widetilde{B} = \{P_1, \ldots, P_{t^\star}\}$, where $1 \leq t^\star \leq n$. Let us define the set of indices $\mathbb{J}^\star = \{j \in \{1, \ldots, \ell\} \text{ s.t. } t_j \leq t^\star\}$, so that the corrupted players can trivially sign messages for signing policies Γ_j, if $j \in \mathbb{J}^\star$.

For the corrupted members of $\mathcal{P}^\star$, the algorithm $\mathcal{A}_{\mathsf{CDH}}$ chooses randomly and independently the shares $\mathsf{sh}_i \in \mathbb{Z}_p$ producing the set $\{(i, \mathsf{sh}_i)\}_{P_i \in \widetilde{B}}$.

For every index $j \in \mathbb{J}^\star$, the algorithm $\mathcal{A}_{\mathsf{CDH}}$ chooses at random a secret $s_j \in \mathbb{Z}_p^*$ and a polynomial $f_j(x) \in \mathbb{Z}_p[x]$ of degree at most $t_j - 1$ such that $f_j(0) = s_j$. It computes (via the hash-table procedure) the values $h_{ij} = H_0(j, \mathsf{sh}_i)$, $r_{ij} = f_j(i) - h_{ij} \bmod p$, for all $P_i \in \widetilde{B}$. For the non-corrupted players, $P_i \notin \widetilde{B}$, the algorithm $\mathcal{A}_{\mathsf{CDH}}$ chooses random and independent values $r_{ij} \in \mathbb{Z}_p$, then computes the values $f_j(i)$ by using the chosen polynomial. Finally, $\mathcal{A}_{\mathsf{CDH}}$ computes the values $PK_j = g^{s_j}$ and $D_{ij} = g^{f_j(i)}$, for all $P_i \in \mathcal{P}^\star$.

For the rest of indices $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$, the algorithm $\mathcal{A}_{\mathsf{CDH}}$ chooses at random $\alpha_j \in \mathbb{Z}_p$ and defines $PK_j = (g^b)^{\alpha_j}$ (which implicitly defines $s_j = b \cdot \alpha_j$). For each $j \notin \mathbb{J}^\star$, $\mathcal{A}_{\mathsf{CDH}}$ chooses at random the values r_{ij}, for all $P_i \in \mathcal{P}$. In particular, this means that, for the corrupted members $P_i \in \widetilde{B}$, we have that the values $r_{ij} + H_0(j, \mathsf{sh}_i) \bmod p$ are already determined. Let $f_j(x) \in \mathbb{Z}_p[x]$ be an implicit polynomial of degree at most $t_j - 1$ such that $f_j(0) = b \cdot \alpha_j \bmod p$ and $f_j(i) = r_{ij} + H_0(j, \mathsf{sh}_i) \bmod p$, for every corrupted player $P_i \in \widetilde{B}$. Since $|\widetilde{B}| = t^\star < t_j$, the algorithm $\mathcal{A}_{\mathsf{CDH}}$ can compute the values $D_{ij} = g^{r_{ij} + H_0(j, \mathsf{sh}_i)}$, for $P_i \in \widetilde{B}$, and then combine these values with $PK_j = (g^b)^{\alpha_j}$ in order to obtain, by interpolation in the exponent, the rest of values $D_{ij} = g^{f_j(i)}$, for non-corrupted players $P_i \notin \widetilde{B}$.

Finally $\mathcal{A}_{\mathsf{CDH}}$ sends to the adversary $\mathcal{A}_{\mathsf{UNF}}$ the secret keys $\{(i, \mathsf{sh}_i)\}_{P_i \in \widetilde{B}}$ of the corrupted players, along with the public information $PK = (PK_1, \ldots, PK_\ell, \mathsf{out}_{\mathsf{pub}}, \{D_{ij}\}_{P_i \in \mathcal{P}^\star, 1 \leq j \leq \ell})$, where $\mathsf{out}_{\mathsf{pub}} = \{(i, j, r_{ij})\}_{P_i \in \mathcal{P}^\star, j \in \{1, \ldots, \ell\}}$.

Simulating H_0. As it happened in the proof of Theorem 2, the simulation of the hash function H_0 is consistent as long as the H_0 hash queries from $\mathcal{A}_{\mathsf{UNF}}$ do not cause a collision with the implicitly determined values $\{\mathsf{sh}_i\}_{P_i \notin \tilde{B}}$. If the number of hash queries for H_0 is q_0, such a collision happens with probability at most $\frac{q_0^2}{2p} + o\left(\left(\frac{q_0^2}{2p}\right)^2\right)$, which is a negligible function of the security parameter $\lambda \leq \log p$.

Signing queries Let (m, j) be a signing query asked by $\mathcal{A}_{\mathsf{UNF}}$. If $(m, j) = (\tilde{m}, \tilde{j})$, then $\mathcal{A}_{\mathsf{CDH}}$ aborts and outputs $\perp$. Otherwise, $\mathcal{A}_{\mathsf{CDH}}$ knows a value β such that $H_1(m, j) = g^\beta$. Then, $\mathcal{A}_{\mathsf{CDH}}$ can easily compute correct signature shares $\sigma_{ij} = H_1(m, j)^{h_{ij}+r_{ij}} = \left(g^{h_{ij}+r_{ij}}\right)^\beta = D_{ij}^\beta$, for every player $P_i \in \mathcal{P}^\star$.

Forgery At some point, and with non-negligible probability, $\mathcal{A}_{\mathsf{UNF}}$ outputs a valid signature $(\sigma^\star, j^\star)$ for some index $j \notin \mathbb{J}^\star$ and some message $m^\star$, different from the valid signatures obtained through signing queries. Since the signature is valid and H_1 behaves as a random function, A_{UNF} must have queried the pair $(m^\star, j^\star)$ to the hash oracle for H_1. With probability at least $1/q_1$, we have $(m^\star, j^\star) = (\tilde{m}, \tilde{j})$, and in this case we have $H_1(m^\star, j^\star) = (g^a)^{\tilde{\beta}}$, for some value β known by $\mathcal{A}_{\mathsf{CDH}}$, whereas the associated part of the public key is $PK_j = (g^b)^{\alpha_j}$, for some value α_j also known by $\mathcal{A}_{\mathsf{CDH}}$. Since $\sigma^\star$ is a valid signature, we have that $\left(g, (g^b)^{\alpha_j}, (g^a)^{\tilde{\beta}}, \sigma^\star\right)$ is a valid Diffie–Hellman tuple, which means that $\sigma^\star = g^{ab\alpha_j\tilde{\beta}}$. In this case, $\mathcal{A}_{\mathsf{CDH}}$ can easily obtain the desired solution g^{ab} of the given instance of the CDH problem. □

For simplicity, we have described a security reduction with a loss factor of q_1. It is possible to improve this reduction by using the techniques of Coron [10], and then the loss factor becomes linear in q_S, the number of signing queries, which is usually considered to be smaller than the number q_1 of hash queries.

5 Application to multi-policy distributed decryption

In this section, we consider the secret task of decryption, instead of signature. We will have a set of users $\mathcal{P} = \{P_1, \ldots, P_n\}$ as the receivers of confidential messages. There will be ℓ different access structures $\Gamma_j \subset 2^{\mathcal{P}}$ defined on $\mathcal{P}$, for $j = 1, \ldots, \ell$. When encrypting, the sender will choose the desired decryption access structure Γ_j. A ciphertext encrypted for the access structure (or decryption policy) Γ_j will be correctly decrypted only if the users in some subset $A \in \Gamma_j$ cooperate to run the protocol $\mathsf{Decrypt}$. Each user $P_i \in \mathcal{P}$ will have a share sh_i of secret information, and will use it to perform his part of the decryption process.

A multi-policy distributed decryption scheme $\Sigma = (\Sigma \cdot \mathsf{St}, \Sigma \cdot \mathsf{KG}, \Sigma \cdot \mathsf{Enc}, \Sigma \cdot \mathsf{Decrypt})$ consists of four probabilistic polynomial-time algorithms: setup, key generation, encryption (public), and decryption (secret and distributed). The encryption algorithm takes as input the index j for the intended decryption policy, and this index j must be included in the output ciphertext. The joint decryption protocol takes as input a ciphertext and secret shares of an authorized subset (according to Γ_j); the result is a plaintext or a special reject symbol $\perp$.

A correct encryption scheme must satisfy the proper confidentiality property. In the distributed setting that we are considering, confidentiality must hold even if an attacker corrupts many members of the collective of receivers, provided the corrupted members are not authorized to decrypt the challenge ciphertext. That is, a ciphertext on the message m addressed to $\mathcal{P}$ for access structure Γ_j leaks no information on m to an attacker who has corrupted a

subset of users $\widetilde{B} \subset \mathcal{P}$ such that $\widetilde{B} \notin \Gamma_j$. This attacker can also make decryption queries for pairs ciphertexts of his choice. If Σ is secure in front of this kind of (polynomial time) adversaries, then we say it enjoys the property of *indistinguishability under chosen ciphertext attacks* (IND-CCA security, for short). This property can be formalized with a security game, but we omit here the details, because it is basically a combination of the security game for the security of (single policy) distributed decryption (see for instance [31]) and the game for multi-policy signatures that we have described in Sect. 4.2.

5.1 A new multi-policy decryption scheme

We propose here a new multi-policy distributed decryption scheme. For simplicity we describe the scheme when all the decryption policies are threshold ones; that is, $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$, where $1 \leq t_1 < t_2 < \cdots < t_\ell \leq n$; the scheme can be extended to work with more general access structures (see Appendix B). The scheme is inspired by the (single) threshold decryption scheme proposed by Shoup and Gennaro in [31]. A key ingredient in the design of the new scheme is, again, the MSSS proposed and analyzed in Sect. 3. The protocols of the multi-policy decryption scheme Σ work as follows.

Setup: $\Sigma \cdot \mathsf{St}(1^\lambda)$.

Given a security parameter $\lambda \in \mathbb{N}$, a group $\mathbb{G} = \langle g \rangle$ of prime order p, such that p is λ bits long, is chosen. A positive integer $l \in \mathbb{N}$, which must be polynomial in λ, is chosen for the maximum number of bits of the messages to be encrypted. Five hash functions are chosen: $H_0 : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \to \{0,1\}^l$, $H_2 : \{0,1\}^* \to \mathbb{G}$, $H_3 : \{0,1\}^* \to \mathbb{Z}_p$, $H_4 : \{0,1\}^* \to \mathbb{Z}_p$. The output of this protocol is $\mathsf{params} = (p, \mathbb{G}, g, l, H_0, H_1, H_2, H_3, H_4)$.

Key generation: $\Sigma \cdot \mathsf{KG}(\mathsf{params}, \mathcal{P}, t_1, \ldots, t_\ell, n)$.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n users and $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$ the threshold decryption policies defined on $\mathcal{P}$, where $1 \leq t_1 < t_2 < \cdots < t_\ell \leq n$. For $j = 1, \ldots, \ell$, the value $PK_j = g^{s_j}$ is computed, for a random value $s_j \in \mathbb{Z}_p^*$ that will remain unknown to the members of $\mathcal{P}$. These ℓ secret values will correspond to a secret vector $\mathbf{s} = (s_1, \ldots, s_\ell)$ of the multi-secret sharing scheme described in Sect. 3, that will be shared by running the distribution protocol $\Omega \cdot \mathsf{Dist}(\mathcal{P}, t_1, \ldots, t_\ell, \mathbf{s})$, with hash function H_0:

1. Choose random values $\mathsf{sh}_i \in \mathbb{Z}_p^\star$, pairwise different for $i = 1, 2, \ldots, n$, as the secret shares.
2. Choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree at most $t_j - 1$, for $j = 1, \ldots, \ell$, such that $f_j(0) = s_j$.
3. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \ell$, compute the values $h_{ij} = H_0(j, \mathsf{sh}_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$.
4. The secret share sh_i is sent to player P_i via a secure channel, whereas the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{(i, j, r_{ij})\}_{P_i \in \mathcal{P}, j \in \{1,\ldots,\ell\}}$.

The secret share for each player P_i is sh_i, whereas the global public key is $PK = (PK_1, \ldots, PK_\ell, \mathsf{out}_{\mathsf{pub}})$. In case one wants to provide robustness to the threshold decryption process, the values $D_{ij} = g^{h_{ij} + r_{ij}}$ must be included in PK, for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell$.

Encryption: $\Sigma \cdot \mathsf{Enc}(\mathsf{params}, m, PK, j)$.

1. Choose at random the values $r_u, r_w \in \mathbb{Z}_p^*$.
2. Compute $c = H_1(PK_j^{r_u}, j) \oplus m$.
3. Use the element g to compute $u = g^{r_u}$ and $w = g^{r_w}$.
4. Use the value $\bar{g} = H_2(c, u, w, j)$ to compute $\bar{u} = \bar{g}^{r_u}$ and $\bar{w} = \bar{g}^{r_w}$.

5. Compute $e = H_3(\bar{g}, \bar{u}, \bar{w}, j)$ and $\sigma = r_w + r_u \cdot e \bmod p$.
6. Return (C, j) with the ciphertext $C = (c, u, \bar{u}, e, \sigma)$.

Joint decryption: $\Sigma \cdot \mathsf{Decrypt}(\mathsf{params}, C, j, PK, A, \{(i, \mathsf{sh}_i)\}_{P_i \in A})$

Let $A \subset \mathcal{P}$ be a subset of users in $\mathcal{P}$ that want to cooperate to decrypt a ciphertext $C = (c, u, \bar{u}, e, \sigma)$ according to the threshold decryption policy $T(t_j, n)$. We assume, thus, $|A| \geq t_j$. Players in A proceed as follows.

1. Each $P_i \in A$ checks if $e = H_3(\bar{g}, \bar{u}, \bar{w}, j)$, where $w = g^{\sigma}/u^e$, $\bar{g} = H_2(c, u, w, j)$, $\bar{w} = \bar{g}^{\sigma}/\bar{u}^e$.
   If this equality does not hold, P_i broadcasts $(i, \perp)$.
2. Otherwise, $P_i \in A$ chooses $v_{ij} \in \mathbb{Z}_p$ at random, recovers r_{ij} from $\mathsf{out}_{\mathsf{pub}}$, computes $h_{ij} = H_0(j, \mathsf{sh}_i)$ and broadcasts the tuple $(i, u_{ij}, e_{ij}, \sigma_{ij})$, where
$$u_{ij} = u^{h_{ij}+r_{ij}},\ \widehat{u}_{ij} = u^{v_{ij}},\ \widehat{h}_{ij} = g^{v_{ij}},\ e_{ij} = H_4(u_{ij},\ \widehat{u}_{ij}, \widehat{h}_{ij})$$
$$\text{and}\quad \sigma_{ij} = v_{ij} + (h_{ij} + r_{ij}) \cdot e_{ij} \bmod p$$
   [If robustness is required, the correctness of this tuple can be publicly verified by checking if $e_{ij} = H_4(u_{ij},\ \widehat{u}_{ij},\ \widehat{h}_{ij})$, where $\widehat{u}_{ij} = u^{\sigma_{ij}}/u_{ij}^{e_{ij}}$, $\widehat{h}_{ij} = g^{\sigma_{ij}}/D_{ij}^{e_{ij}}$. Note that this check ensures that (u, D_{ij}, u_{ij}) is a valid Diffie–Hellman triple.]
3. If there are no t_j valid shares, stop and output $\perp$. Otherwise, from t_j valid tuples $\{(i, u_{ij}, e_{ij}, \sigma_{ij})\}_{P_i \in A}$, different from $(i, \perp)$, one can consider the Lagrange interpolation coefficients $\lambda_{ij}^A \in \mathbb{Z}_p$ such that $s_j = f_j(0) = \sum_{P_i \in A} \lambda_{ij}^A \cdot f_j(i)$.
4. Return the message $m = c \oplus H_1(\prod_{P_i \in A} u_{ij}^{\lambda_{ij}^A}, j)$.

5.2 Security analysis

A first attempt to prove the security of the new multi-policy decryption scheme would be to reduce its security to the security of the inherent MSSS, which is proved in Sect. 3.2. However, such a reduction does not work, because in the new decryption scheme, the values $PK_j = g^{s_j}$ are public, for $j = 1, \ldots, \ell$. In this scenario it is trivial to distinguish between two potentially shared secrets $s_j^{(0)} \neq s_j^{(1)}$, chosen by the adversary. Therefore, in order to prove the security of the multi-policy decryption scheme, we have to construct a whole proof, simulating all the values that an adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ would see in the execution of the different protocols of Σ. We would use the hypothetical existence of such a successful adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ to solve a computationally hard problem, the CDH problem.

This proof is basically a combination of the techniques in the security proof of the threshold decryption scheme in [31] and the techniques that we have already used in the proofs of Theorems 2 and 3, and so it is omitted. The proof is in the random oracle model for the five hash functions H_0, H_1, H_2, H_3, H_4. The result that ensures the security of the new multi-policy distributed decryption scheme is the following theorem.

Theorem 4 *In the random oracle model, the scheme Σ is IND-CCA secure, assuming the Computational Diffie–Hellman problem is hard to solve in $\mathbb{G}$.*

6 Relations with attribute-based cryptography

In an attribute-based cryptosystem, each user has a subset of attributes $A \subset \mathcal{P}$ from a universe $\mathcal{P} = \{\mathsf{at}_1, \ldots, \mathsf{at}_n\}$ of attributes, and receives from a trusted master entity a secret key

according to those attributes. Later, the specific access policy $\Gamma \subset 2^{\mathcal{P}}$ defining the subsets of attributes that must be held by someone so that he is able to perform the secret task (either decrypting or signing) is chosen "on the fly", among all the possible (monotone increasing) access policies in $\mathcal{P}$. Attribute-based cryptosystems have received a lot of attention from the cryptographic community in the last years, and different schemes have been proposed both for encryption (see for instance [16,22]) and for signatures (for instance, in [19,24]).

It is easy to see that attribute-based cryptosystems are (in some way) a more general primitive than the primitives of multi-policy distributed cryptosystems that we have introduced in this work. Let us consider, for instance, the case of encryption/decryption (the case of signatures work in an analogous way). Let us take an attribute-based encryption scheme, and let us associate each attribute $\mathsf{at}_i \in \mathcal{P}$ with one player $P_i \in \mathcal{P}$. Each player receives from the master entity the secret key (or secret share) sh_i corresponding to the fact of holding only attribute at_i (all these secret keys $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ are computed by the master entity in a single execution of the key generation protocol, with a common randomness). Later, the sender of a message m addressed to $\mathcal{P}$ chooses the desired decryption policy $\Gamma \subset 2^{\mathcal{P}}$ and encrypts m by using the attribute-based encryption protocol. Only if an authorized subset of players $A \subset \mathcal{P}$, $A \in \Gamma$ put together their secret shares, they will be able to run the attribute-based decryption protocol and recover the message m.

Therefore, it seems that the primitives of multi-policy distributed decryption and signature can be already implemented by using existing attribute-based cryptosystems, and actually the resulting schemes are more general, because the allowed decryption / signing policies must not be, in principle, inside a pre-defined list $\{\Gamma_1, \ldots, \Gamma_\ell\}$, as it happens in our multi-policy distributed cryptosystems; they can be whatever access policy defined on the set $\mathcal{P}$. However, we will explain below a list of drawbacks suffered by this attribute-based approach, as opposed to the direct approach that we have followed in this work to design multi-policy distributed cryptosystems.

Restricted access policies Even if, in theory, an attribute-based cryptosystem could allow encryptions or signatures for any access policy $\Gamma \subset 2^{\mathcal{P}}$, in specific proposals this is not always the case. For instance, the most efficient attribute-based cryptosystems proposed up to date (in terms of the length of ciphertexts or signatures, the computational cost of the protocols, etc.) admit only threshold policies [1,19]. The resulting functionality can be achieved by our multi-threshold decryption and signature schemes, by taking $\ell = n$ and $t_j = j$, for all $j = 1, \ldots, n$.

Necessity of a trusted master entity In any attribute-based cryptosystem, a master entity must generate and distribute the secret keys between users. This means that this entity has to be trusted, because otherwise it could impersonate any user in the system. Although we have described our multi-threshold schemes with a trusted dealer who generates and distributes the secret shares, this is not an intrinsic property of this kind of schemes, and actually we show in Appendix A how the own players in $\mathcal{P}$ can generate the public parameters and the secret shares by themselves, without the participation of any external (and trusted) dealer.

Length of ciphertexts, public parameters, signatures and secret shares In most of the attribute-based cryptosystems proposed so far, the length of the ciphertexts or signatures is at least linear in the number of attributes involved in the access policy (which, for simplicity, we assume to be n). In our multi-threhold cryptosystems, the length of ciphertexts and signatures is constant. The only attribute-based cryptosystems with constant-length ciphertexts or signatures [1,19], on the other hand, have secret keys whose length is at least linear in n. In our multi-threshold schemes, each secret key (or share) contains a single element in $\mathbb{Z}_p$. The length of the public

parameters in our schemes, which is linear in $n \cdot \ell$, is comparable to the length of the public parameters in all existing attribute-based cryptosystems, which is at least linear in n, and sometimes linear in n^2.

Computational assumptions Up to now, all the existing (and moderately efficient) attribute-based cryptosystems with a formal proof of security make use of bilinear pairings, and base their security on (sometimes, quite artificial) computational assumptions related to bilinear groups. For multi-policy distributed cryptosystems, we have seen that they can be constructed by combining, essentially, a MSSS with a standard distributed cryptosystem. The particular instantiation of a multi-policy distributed decryption scheme that we have described and analyzed in Sect. 5, for instance, is provably secure under the well-established assumption that the Computational Diffie–Hellman problem is hard.

Appendix A: Joint generation of the secret shares

For simplicity, in both our MSSS and our multi-policy distributed decryption/signature schemes, we have assumed the existence of a trusted entity, a dealer, who generates some secret keys, distributes the shares between all the participants using a secure channel and outputs the public information.

Let us show how the own members of $\mathcal{P} = \{P_1, \ldots, P_n\}$ could do this task by themselves, for the proposed MSSS, without any interaction with a trusted entity. The setup protocol $\Omega \cdot \mathsf{Stp}(1^\lambda, \mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell)$ works exactly in the same way: we will have $1 \leq t_1 < t_2 < \cdots < t_\ell \leq n$ for the ℓ different thresholds that define the access structures $\{\Gamma_j\}_{1 \leq j \leq \ell}$, and will choose a large prime $p > n$ and a hash function $H : \mathbb{N} \times \mathbb{Z}_p^\star \to \mathbb{Z}_p$. For simplicity, we assume that the adversary is honest-but-curious, and so data is not corrupted.

Generation of the secret and distribution of the shares: $\Omega \cdot \mathsf{Dist}(\mathsf{pms})$.

Now the global secret $\mathbf{s} \in (\mathbb{Z}_p)^\ell$ that will be distributed is not an input of the protocol, because it will be generated by players in $\mathcal{P}$ "on the fly", according to the following steps.

1. Every $P_i \in \mathcal{P}$ chooses a random value $\mathsf{sh}_i \in \mathbb{Z}_p^\star$ as his secret share.
2. Every $P_i \in \mathcal{P}$ chooses random values $s_{ij}, a_{ij}^{(1)}, a_{ij}^{(2)}, \ldots, a_{ij}^{(t_j-1)}$ in $\mathbb{Z}_p$, for $j = 1, \ldots, \ell$, to use in the polynomial $f_{ij}(x) = s_{ij} + a_{ij}^{(1)} x + a_{ij}^{(2)} x^2 + \cdots + a_{ij}^{(t_j-1)} x^{t_j-1} \in \mathbb{Z}_p[x]$ of degree at most $t_j - 1$. The secret s_j is implicitly defined as $s_j = \sum_{P_i \in \mathcal{P}} s_{ij} = \sum_{P_i \in \mathcal{P}} f_{ij}(0)$, and the global secret is $\mathbf{s} = (s_1, \ldots, s_\ell) \in (\mathbb{Z}_p)^\ell$.
3. Every $P_i \in \mathcal{P}$ sends the ℓ values $\{f_{ij}(k)\}_{1 \leq j \leq \ell}$ to the other participants $P_k \in \mathcal{P}$. At this point, the participant $P_k \in \mathcal{P}$ is able to compute his own secret value $b_{kj} = \sum_{P_i \in \mathcal{P}} f_{ij}(k)$, which is a polynomial (Shamir) share of the secret s_j.
4. Every $P_i \in \mathcal{P}$ computes $h_{ij} = H(j, \mathsf{sh}_i)$ and broadcasts the values $r_{ij} = b_{ij} - h_{ij} \bmod p$, for $j = 1, 2, \ldots, \ell$.
5. Finally, the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{(i, j, r_{ij})\}_{P_i \in \mathcal{P}, j \in \{1, \ldots, \ell\}}$.

Reconstruction of the secrets: $\Omega \cdot \mathsf{Rec}(\mathsf{pms}, \mathsf{out}_{\mathsf{pub}}, j, \{(i, \mathsf{sh}_i)\}_{P_i \in A})$.

When the players of an authorized subset $A \in \Gamma_j$ (i.e. $|A| \geq t_j$) want to recover the secret s_j, they must cooperate performing the following steps.

1. Each player $P_i \in A$ computes $h_{ij} = H(j, \mathsf{sh}_i)$.
2. Take the values $\{(i, j, r_{ij})\}_{P_i \in A}$ from $\mathsf{out}_{\mathsf{pub}}$ and compute $b_{ij} = r_{ij} + h_{ij} \bmod p$, for every $P_i \in A$.

3. Use the values $\{b_{ij}\}_{P_i \in A}$ to interpolate the polynomial $F_j(x) = \sum_{P_i \in \mathcal{P}} f_{ij}(x)$ in $x = 0$, recovering in this way the j-th secret $s_j = F_j(0)$.

The idea to make the scheme secure against active adversaries (who can send incorrect values during the protocol) is to consider verifiable secret sharing techniques ([13,27]) as described for instance in [15]. Slight modifications of the scheme above are necessary to achieve this goal. For instance, every participant $P_i \in \mathcal{P}$ must publish, in Step 2 of the protocol $\Omega \cdot \mathsf{Dist}$, the commitments $A_{ij}^{(u)} = g^{a_{ij}^{(u)}}$ to the coefficients of his polynomials, with $j \in \{1, \ldots, \ell\}$ and $u \in \{1, \ldots, t_j - 1\}$. Later, these commitments are used to detect incorrect values that are sent in Step 3 of the protocol $\Omega \cdot \mathsf{Dist}$ or broadcast in Step 2 of the protocol $\Omega \cdot \mathsf{Rec}$.

Appendix B: More general access structures

We have described our new MSSS (Sect. 3) and our new multi-policy distributed decryption and signature schemes (Sects. 5.1 and 4.3) for the particular case where the access structures $\Gamma_1, \ldots, \Gamma_\ell$ are all threshold ones. However, all these protocols can be easily extended to the case of more general access structures, as long as they admit a linear and ideal secret sharing scheme (also known as *vector space secret sharing scheme* [8]).

An access structure Γ is realizable by vector space secret sharing scheme, over a finite field $\mathcal{K}$, if there exist a positive integer h and a map $\psi : \mathcal{P} \cup \{D\} \longrightarrow (\mathcal{K})^h$, such that $A \in \Gamma$ if and only if $\psi(D) \in \langle \psi(P_i) \rangle_{P_i \in A}$. If a dealer wants to distribute a secret value $s \in \mathcal{K}$ according to such an access structure, he takes a random vector $\omega \in (\mathcal{K})^h$, such that $\omega \cdot \psi(D) = s$. The share of a player $P_i \in \mathcal{P}$ is $\mathsf{sh}_i = \omega \cdot \psi(R_i) \in \mathcal{K}$. Let A be an authorized subset, $A \in \Gamma$; then, by definition, $\psi(D) = \sum_{P_i \in A} \lambda_i^A \psi(P_i)$, for some values $\lambda_i^A \in \mathcal{K}$. In order to recover the secret from their shares, players in A compute

$$\sum_{P_i \in A} \lambda_i^A \mathsf{sh}_i = \sum_{P_i \in A} \lambda_i^A \, (\omega \cdot \psi(P_i)) = \omega \cdot \sum_{P_i \in A} \lambda_i^A \psi(P_i) = \omega \cdot \psi(D) = s.$$

Shamir's threshold secret sharing scheme, with threshold t, can be seen as a particular case of vector space secret sharing, by defining $\psi(D) = (1, 0, \ldots, 0) \in (\mathbb{Z}_q)^t$ and $\psi(P_i) = (1, i, i^2, \ldots, i^{t-1}) \in (\mathcal{K})^t$ for every player $P_i \in \mathcal{P}$.

If we come back to our new MSSS, in Sect. 3, it can be easily modified to work in situations where each access structure Γ_j admits a vector space secret sharing scheme. We just have to replace polynomials with vectors, and polynomial evaluations with scalar products.

The same happens with our multi-policy distributed cryptosystems. If the access structures admit a vector space secret sharing scheme, then the Key Generation process can be modified as explained above; later, the encryption/decryption or signature/verification operations can be (slightly) modified to work with these more general access structures and linear secret sharing schemes, because they involve only linear operations (additions and multiplications with a constant value). Even the version of our protocols that works without any trusted dealer (see Appendix A) can be extended using the ideas and techniques from [18].

Acknowledgments Javier Herranz enjoys a *Ramón y Cajal* Grant, partially funded by the European Social Fund (ESF), from Spanish MICINN Ministry. The research of Javier Herranz and Germán Sáez is also supported by Projects MTM2009-07694 and ARES—CONSOLIDER INGENIO 2010 CSD2007-00004, of the same MICINN Ministry.

References

1. Attrapadung N., Herranz J., Laguillaumie F., Libert B., De Panafieu E., Ràfols C.: Attribute-based encryption schemes with constant-size ciphertexts. Theor. Comput. Sci. **422**, 15–38 (2012).
2. Bellare M., Rogaway P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Proceedings of CCS'93, pp. 62–73. ACM Press, New York (1993).
3. Blakley G.R.: Safeguarding cryptographic keys. Proc Natl Comput Conf Am Fed Inf Process Soc Proc **48**, 313–317 (1979).
4. Blundo C., De Santis A., Di Crescenzo G., Gaggia A.G., Vaccaro U.: Multi-secret Sharing Schemes. In: CRYPTO '94, LNCS 839, pp. 150–163. Springer, Heidelberg (1994).
5. Boldyreva A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie–Hellman-Group Signature Scheme. In: Proceedings of PKC'03, LNCS 2567, pp. 31–46. Springer, Berlin (2003).
6. Boneh D., Franklin M.: Identity-based encryption from the Weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003).
7. BBoneh D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004).
8. Brickell E.F.: Some ideal secret sharing schemes. J. Comb. Math. Comb. Comput. **9**, 105–113 (1989).
9. Cachin C.: On-line Secret Sharing. In: Proceedings of IMA conference'95, LNCS 1025, pp. 190–198. Springer, New York (1995).
10. Coron J.S.: On the Exact Security of Full Domain Hash. In: Proceedings of crypto'00, LNCS 1880, pp. 229–235. Springer, Berlin (2000).
11. Cover T.M., Thomas J.A.: Elements of Information Theory. Wiley, New York (1991).
12. Csirmaz L.: How to share secrets simultaneously. Cryptol. ePrint Arch. http://eprint.iacr.org/2011/386 (2011).
13. Feldman P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: Proceedings of FOCS'87, vol. 28, pp. 427–437 (1987).
14. Gennaro R., Jarecki S., Krawczyk H., Rabin T.: Robust Threshold DSS Signatures. In: Proceedings of Eurocrypt'96, LNCS 1070, pp. 354–371. Springer, Berlin (1996).
15. Gennaro R., Jarecki S., Krawczyk H., Rabin T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007).
16. Goyal V., Pandey O., Sahai A., Waters B.: Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In: Proceedings of ACM CCS'06, pp. 89–98. ACM Press, New York (2006).
17. He J., Dawson E.: Multisecret-sharing scheme based on one-way function. Electron. Lett. **31**(2), 93–95 (1995).
18. Herranz J., Sáez G.: Verifiable Secret Sharing for General Access Structures, with Application to Fully Distributed Proxy Signatures. In: Proceedings of financial cryptography'03, LNCS 2742, pp. 286–302. Springer, Berlin (2003).
19. Herranz J., Laguillaumie F., Libert B., Ràfols C.: Short Attribute-Based Signatures for Threshold Predicates. In: Proceedings of CT-RSA'12, LNCS 7178, pp. 51–67. Springer, Berlin (2012).
20. Jackson W.A., Martin K.M., OKeefe C.M.: A construction for multisecret threshold schemes. Des. Codes Cryptogr. **9**(3), 287–303 (1996).
21. Joux A., Nguyen K.: Separating decision Diffie–Hellman from computational Diffie–Hellman in cryptographic groups. J. Cryptol. **16**(4), 239–247 (2003).
22. Lewko A., Okamoto T., Sahai A., Takashima K., Waters B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Proceedings of Eurocrypt'10, pp. 62–91 (2010).
23. Lin H.Y., Yeh Y.S.: Dynamic multi-secret sharing scheme. Int. J. Contemp. Math. Sci. **3**(1), 37–42 (2008).
24. Maji H.K., Prabhakaran M., RosulekM.: Attribute-Based Signatures. In: Proceedings of CT-RSA'11, LNCS 6558, pp. 376–392. Springer, Berlin (2011).
25. Masucci B.: Sharing multiple secrets: models, schemes and analysis. Des. Codes Cryptogr. **39**, 89–111 (2006).
26. Okamoto T., Pointcheval D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Proceedings of PKC'01, LNCS 1992, pp. 104–118. Springer, Berlin (2001).
27. Pedersen T.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Proceedings of crypto'91, LNCS 576, pp. 129–140. Springer, Berlin (1991).
28. Shamir A.: How to share a secret. Commun. ACM **22**, 612–613 (1979).
29. Shoup V.: Lower Bounds for Discrete Alogarithms and Related Problems. In: Proceedings of eurocrypt'97, LNCS 1233, pp. 256–266. Springer, Berlin (1997).

30. Shoup V.: Practical Threshold Signatures. In: Proceedings of Eurocrypt'00, LNCS 1807, pp. 207–220. Springer, Berlin (2000).
31. Shoup V., Gennaro R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptol. **15**(2), 75–96 (2002).