UNIVERSITAT POLITÈCNICA DE CATALUNYA
Departament de Matemàtica Aplicada IV

Some Digital Signature Schemes with Collective Signers

PhD. Thesis, written by
Javier Herranz Sotoca
Advisor of the thesis:
Dr. Germán Sáez i Moreno

Some Digital Signature Schemes with Collective Signers

Javier Herranz Sotoca

Abstract

Digital signatures are one of the most important consequences of the appearance of public key cryptography, in 1976. These schemes provide authentication, integrity and non-repudiation to digital communications. Some extensions or variations of the concept of digital signature have been introduced, and many specific realizations of these new types of signature schemes have been proposed.

In this thesis, we deal with the basic definitions and required security properties of traditional signature schemes and two of its extensions: distributed signature schemes and ring signature schemes. We review the state of the art in these two topics; then we propose and analyze new specific schemes for different scenarios.

Namely, we first study distributed signature schemes for general access structures, based on RSA; then we show that such schemes can be used to construct other cryptographic protocols: distributed key distribution schemes and metering schemes. With respect to ring signatures, we propose schemes for both a scenario where the keys are of the Discrete Logarithm type and a scenario where the public keys of users are inferred from their personal identities. Finally, we also propose some distributed ring signature schemes, a kind of schemes which combine the concepts of distributed signatures and ring signatures.

We formally prove the security of all these proposals, assuming that some mathematical problems are hard to solve. Specifically, we base the security of our schemes in the hardness of either the RSA problem, or the Discrete Logarithm problem, or the Computational Diffie-Hellman problem.

Acknowledgements

I wish to thank different people for the support and the good moments they have given to me during the realization of this thesis.

Encara que sigui el més lògic i habitual començar agraint el director de la tesi, no seré jo qui deixi de fer-ho. Entre d'altres coses perquè el meu, el Germán Sáez, es mereix sobradament que els primers agraïments i els més sentits siguin per a ell, per haver acceptat dirigir-me la tesi i per haver-ho fet tan bé, per hores i hores de feina compartides, pels bons moments i els no tan bons, per estar sempre a punt per a ajudar-me, pels consells, les crítiques i per deixar-se aconsellar i criticar, també. I a part de la vessant més científica, doncs gràcies per ser un gran amic. No sé on ens portaran les nostres vides, si podrem continuar treballant junts o no, espero que sí, però l'amistat i l'afecte segur que no es perdran. Gràcies, i no canviïs mai!

Seguint per l'àmbit més professional, però també amb moltes connotacions afectives, doncs agrair el suport de tot el grup de criptografia del Departament de Matemàtica Aplicada IV, de la Universitat Politècnica de Catalunya. Seria injust no destacar la Vanesa Daza, amb qui he compartit moltes hores de feina i de converses, de propostes, de bestieses, de secrets. També el David Galindo, font inesgotable de comentaris i consells molt valuosos en el pla professional, i de moments bojos i divertits en altres plans. I també el Carles Padró, qui em va oferir la possibilitat d'introduïr-me en el món de la investigació criptogràfica mentre estudiava a la facultat. En general gràcies a tots els membres del grup, representats pels caps visibles, la Paz Morillo i el Jorge Villar. Espero que tot vagi tan bé en el futur com en aquests quatre anys que he compartit amb vosaltres.

Los agradecimientos institucionales son para el Ministerio de Ciencia y Tecnología (actualmente Ministerio de Educación y Ciencia) por concederme una beca FPI para los cuatro años de realización de esta tesis, y para el Departament de Matemàtica Aplicada IV de la Universitat Politècnica de Catalunya por acogerme durante este tiempo como un miembro más.

Je voudrais remercier chaleureusement le professeur Jacques Stern de m'accueillir pendant six mois dans son groupe de recherche au Département

d'Informatique de l'École Normale Supérieure, à Paris. J'ai passé de très bons moments là-bas, grâce à l'amabilité et la sympathie des membres du groupe: Dario Catalano, David Pointcheval, Phong Nguyen, Michel Abdalla, Duong Hieu Phan, etc.

Por último, ya fuera del terreno científico, quisiera en primer lugar dar las gracias a mis padres y mi hermana por el buen ambiente y la estabilidad que dan a nuestro hogar, por el amor y el afecto que no siempre se dice, pero siempre se siente. También quiero destacar al resto de amiguetes con los que he pasado y espero pasar tantos buenos ratos, tantas risas, tantas experiencias, tantos viajes, tantos chascarrillos... y aquí están incluidos los amigos de Esplegares, de la FME, los amigos del Sagrat Cor y allegados, y otras personas inclasificables que en algún u otro momento me han hecho feliz. Gracias, y espero haberos correspondido como os merecéis, al menos lo he intentado!

Contents

Introduction

Until 1976, cryptography was restricted to symmetric key encryption schemes: two users had a common secret key, that they used in order to encrypt and decrypt messages. In this way, confidentiality and authentication of the communication were ensured, but not other properties such as non-repudiation. Furthermore, to securely agree in a common secret key is not an easy task in practical situations. Last but not least, a user must agree in a secret key with any other user with whom he wants to communicate; this implies that any user must store a large number of secret keys.

To solve these drawbacks inherent to symmetric cryptography, Diffie and Hellman [40] introduced in 1976 the revolutionary concept of *public key cryptography*. This concept can be applied to design public key encryption schemes, which provide secure communication among users without having to establish a common secret key: each user A publishes his public key, which matches with a secret key that A only knows. To send a message to A, a different user B encrypts it by using the public key of A. Decrypting this ciphertext is only possible with the knowledge of the matching secret key, so only the user A can obtain the original message by using his secret key.

Apart from this, the introduction of the paradigm of public key cryptography involved a new method to provide authentication and non-repudiation to digital communications: *digital signature schemes*. In some sense, the idea is to have an equivalent of handwritten signatures for digital communication: the author of a message adds another piece of information, the signature, to prove that he, and only he, is the responsible of the contents of the message.

Obviously, the properties of authentication and non-repudiation are guaranteed as long as the signature scheme satisfies some security requirements. Roughly speaking, an attacker who does not know the secret key of some user, must not be able to compute a new valid signature for some message, even if he has obtained from the legitimate user valid signatures for messages that the attacker has chosen.

In Chapter 1 of this thesis, we explain in more detail the concept of digital signature scheme and the required security properties that such a

scheme must satisfy. We exemplify these explanations with the RSA signature scheme and the Schnorr's signature scheme.

During the last years, the concept of signature scheme has been extended to different scenarios. The security model of traditional signature schemes must also be properly extended to fit in with these new scenarios. In this work we deal basically with two of these types of signature schemes: distributed signature schemes and ring signature schemes.

Distributed Signature Schemes

In a distributed signature scheme, a set of players share the power to sign a message. There is a family of authorized subsets, and a family of corruptible subsets. If all players of some authorized subset agree, they can compute a valid signature; on the other hand, even if an attacker knows all the information of the players of some corruptible subset, it must not be able to obtain valid signatures.

The fact that the secret information is distributed among a set of players, and not centralized in a single user, as it happens in traditional signature schemes, makes distributed systems more secure and reliable. In effect, an attacker should corrupt many participants (or machines) in order to forge signatures. On the other hand, even if some machines are out of service, valid signatures can be computed if enough machines remain active.

Usually, these distributed schemes are designed from a standard (individual) scheme. The secret key of an individual user in the standard scheme is distributed in shares by means of a secret sharing scheme. Each participant of the set receives a share of the secret key. Later, each one uses his partial secret information to perform his part of the task (in our case, signing a message).

Distributed cryptography is also known as *threshold* cryptography, because most of the works in this area consider only threshold families of authorized and corruptible subsets, which contain all the subsets with a minimum (or maximum, for the family of corruptible subsets) number of participants.

First works on distributed cryptography appeared in the late eighties [15, 39]. In the area of distributed signature schemes, different proposals have appeared throughout the last fifteen years. With respect to schemes whose security is based on the difficulty of solving the Discrete Logarithm problem, we can cite the proposals in [53, 93]. With respect to schemes based on the RSA primitive (related to the difficulty of factoring large integers), the most significative proposals can be found in [38, 52, 91, 31, 47].

One of our first goals was to extend some proposals of threshold signature schemes to a framework with general structures for both the authorized

and the corruptible subsets.

In some cases, the generalization can be done without too many difficulties, by using general linear secret sharing schemes instead of threshold secret sharing schemes. This happens, for example, in the case of distributed Schnorr's signatures, where the delicate point is to extend verifiable secret sharing schemes, already proposed for the threshold case, to a general framework. We have done this and the proposal was published in [59]. In that work, we also proposed a fully distributed proxy signature scheme, where a distributed entity delegates its signing capabilities into a distributed proxy entity; the proxy entity can sign messages on behalf of the original entity, and the recipient verifies at the same time the delegation of the original entity and the signature of the proxy entity. The initial proposal in [59] was revised and extended with a formal security analysis [63].

In other cases, for example for RSA distributed signature schemes, the generalization is more complicated. We have studied the RSA threshold scheme proposed by Shoup in [91], because it is the most efficient and conceptually simple one. In distributed RSA schemes, traditional secret sharing schemes cannot be used, because the secret to be shared belongs to a set which is unknown to the participants. We have found the algebraic and combinatorial conditions that must be satisfied by the general families of authorized and corruptible subsets and by the particular secret sharing scheme in order to obtain a secure RSA distributed signature scheme. This work has been published in [58] and is explained in detail in Chapter 2 of this thesis.

There are many other practical situations where distributed cryptography can be useful. For example, metering schemes are designed to count the number of interactions between a set of clients and a set of servers; to do it, the usual strategy consists in sharing some secret information among these sets.

Another typical problem involves a set of users who want to obtain a secret common key, in order to securely communicate among them, or as an access key to restricted resources, for example. There are different solutions to this problem, depending on the situation, and we have proposed specific schemes for some of them: key distribution for dynamic groups [35], group key exchange [65], and distributed key distribution schemes [34]. In these last schemes, a set of servers provide the users with the necessary information to obtain the common secret key in a secure way.

In Chapter 3 of this thesis, we show that distributed signature schemes can be used as a primitive to design, on the one hand, metering schemes, and on the other hand, distributed key distribution schemes. If the distributed signature scheme is non-interactive and secure, then we get a secure metering scheme. If the distributed signature scheme is deterministic and secure, then

we also get a secure distributed key distribution scheme. A first version of this work, considering threshold access structures, was published in [36]; later, a version for general access structures that includes a more formal security analysis has been published in [37].

Ring Signature Schemes

In a ring signature scheme, a user anonymously computes a signature on behalf of a set of users that he chooses and which includes himself. The recipient of the signature is convinced that some member of the set has signed the message, but he has no information about who is the actual signer.

This fact is formally described with the concepts of anonymity and unforgeability of ring signature schemes. Anonymity means that nobody, even having unlimited computational and time resources, can distinguish who has really computed a ring signature. Unforgeability means that an attacker cannot compute a new valid ring signature for some message and some set of users, if he does not know any of the secret keys of the members of this set.

Ring signatures can be applied in situations where anonymity is required, but also as a primitive to construct other cryptographic schemes, like signature schemes with designated verifier or concurrent signature schemes, useful for situations where two parties must sign a contract in a (more or less) simultaneous way.

It is also possible to combine the concepts of distributed signatures and ring signatures. The resulting schemes, that we call distributed ring signature schemes, allow a group of users to jointly sign a message on behalf of some family of subsets that contains themselves. The recipient of the signature is convinced that all members of some of the subsets of the family have cooperated in the computation of the signature, but he cannot distinguish which subset is the actual signer. The difference with respect to standard distributed signature schemes is that the family of authorized subsets is chosen *ad-hoc* by the signing users; furthermore, each user has its own secret and public keys, generated individually, and not in a distributed process, as it happens in distributed signature schemes, where the whole set of users has a common public key and the secret key is shared among them.

The concept of ring signature schemes was formally introduced in [86]. Afterwards, some other proposals have been done in [16, 1, 41]. All these proposals run in a standard scenario where each user privately creates its secret and public keys, and therefore digital certificates must be provided by some trusted authority to link identities with public keys. A different scenario is that of identity-based cryptography, where public keys of the

users can be directly derived from their identities (for example, from their e-mail addresses). The only ring signature scheme for identity-based scenarios that had been published before our proposals is the one in [95].

With respect to distributed ring signature schemes, all previous proposals [16, 94, 2, 25] are restricted to the case where the ad-hoc family of possible signing subsets is necessarily threshold.

In Chapter 4 of this thesis, we introduce a generic family of ring signature schemes, and we state and prove a result concerning the security of this family of schemes. Then, we design a particular scheme which falls within this family. The scheme follows the ideas behind Schnorr's signature scheme. We prove the unconditional anonymity of the scheme and its unforgeability, by using the generic security result, assuming the hardness of the Discrete Logarithm problem. These results have been published in [60]. We extend this particular scheme to a distributed ring signature scheme which works with any ad-hoc family of signing subsets. The work where we propose this distributed scheme, including the proof that it is secure if the Discrete Logarithm problem is hard to solve, has been published in [61].

Finally, Chapter 5 is devoted to ring signature schemes in identity-based scenarios. We propose a new scheme, with better efficiency properties than the only previously known ring signature scheme based on identities [95]. Our scheme also falls within the family of generic ring signature schemes that we introduce in Chapter 4. Therefore, we can use the generic security result to prove that the scheme is secure, assuming in this case that the Computational Diffie-Hellman problem is hard to solve. We extend the scheme, as we do in Chapter 4, in order to obtain a distributed ring signature scheme for general families of signing subsets. These two schemes have been published in [62]. Finally, we propose an alternative construction of distributed ring signature schemes which use the concept of dual access structures to work also with general families of possible signing subsets; this last scheme is analyzed for the case of identity-based scenarios, but it can be extended to more general cases where users have different types of keys. The results concerning this last proposal can be found in [64]. The security of the two proposals of identity-based distributed ring signature schemes is also based on the difficulty of solving the Computational Diffie-Hellman problem.

Chapter 1

Preliminaries

In this chapter, we first explain the basic concepts related to public key cryptography. Then we concentrate on digital signatures. We give the intuition behind this concept, and later we provide the necessary formal definitions about the protocols that take part in a digital signature scheme and, perhaps more importantly, about the security requirements that such schemes must satisfy.

We illustrate these concepts with two examples of digital signature schemes: RSA and Schnorr's. We have chosen these schemes because they are maybe the most studied ones, and specially because they will be the basis of some of the constructions that we design in the rest of chapters of this thesis.

We also provide a simple example to explain what a proof by reduction means, in the context of cryptographic protocols, and why it is important to study the exact security of digital signature schemes.

1.1 Public Key Cryptography

In the past, cryptographic schemes were restricted to *symmetric* or *private key* cryptosystems: the two parties who wanted to securely communicate each other had a common secret key, which was used in order to encrypt and decrypt messages. Usually, cryptosystems consisted on transpositions and substitutions applied to the letters, or bits, of the message. For example, the Caesar cipher with secret key $k = 3$ consisted on applying a shift of 3 places to each letter, forward to encrypt, and backward to decrypt. In this way, the encryption of the word 'peace' would be 'shdfh', and so on.

Security for symmetric encryption schemes was defined in an *information-theoretic* model. That is, the goal is to prevent an attacker from obtaining

any information about a message, when it is given the ciphertext, even in the case that it has unlimited computational resources. Shannon proved [90] that this perfect level of security can be achieved if and only if the length of the secret key is equal or bigger than the length of the encrypted message. In this case, a secure symmetric encryption scheme can be constructed as follows: if the secret key is k and the message to encrypt is m (both expressed in bits), then the ciphertext is $c = m \oplus k$, which gives no information about the message m, provided the attacker does not know any bit of the secret key. Furthermore, if the receiver of the ciphertext knows the full secret key, he can easily decrypt the ciphertext and obtain the message $m = c \oplus k$.

Similar ideas were used to provide authentication to communications, with the concept of *message authentication codes*, where the sender and the receiver must also share the same secret key.

Of course, this requirement is quite hard, because it is not always easy to find a secure way to agree in a common secret key. *Public key cryptography* provided a solution to this problem.

Diffie and Hellman introduced in 1976 this revolutionary concept, in their paper [40]. The idea of a public key encryption scheme is the following: a user has a secret key sk that he only knows; there exists a matching public key pk which is known by everybody. If someone wants to send a private information to this user, he can apply to the message a function that depends on the public key pk. This function must be easy to compute, but computationally infeasible to invert, when knowing only the description of the function. However, with the knowledge of the secret key sk (a trapdoor), the function can be easily inverted and the original message can be recovered. For these reasons, such functions are known as *trapdoor one-way functions*.

In practice, public key encryption schemes are less efficient than private key ones. For this reason, the usual process is the following: a public key encryption scheme is used between two participants in order to agree on a common secret key k. Then, a more efficient symmetric encryption scheme is employed for future communication, by using the established common key k.

This simple and elegant idea completely modified the field of cryptography; mathematics started to play a more important role on it (whereas physics or computer science had been the main basis of secret key cryptography), because it was quite obvious that mathematics would be necessary to find specific trapdoor one-way functions. The first one was the RSA function, proposed by Rivest, Shamir and Adleman in [85].

Security of public key encryption schemes is not defined in an information-theoretic model, but in a *computational* one. Roughly speaking, the goal is that an attacker with limited computational resources cannot obtain any in-

formation about a plaintext when it is given the ciphertext. This is usually proved by reducing this problem to some well-known and difficult problem. For example, a public key encryption scheme is considered secure if one can prove a statement such as: "if an attacker can break the encryption scheme, then it could factor an integer which is the product of two big prime numbers".

The appearance of public key cryptography involved a new mechanism to provide not only authentication, but also non-repudiation, to communications: *digital signature schemes*. The process in such a scheme is kind of the opposite than in a public key encryption one: the user who wants to sign a message applies a function on it that depends on the secret key that only he knows. Later, everybody can use the public key of the user to verify that the signature is valid.

If the signature scheme is secure, which informally means that valid signatures cannot be forged without the knowledge of the secret key, this method provides, in effect, authentication and non-repudiation: the signer cannot deny that he is the author of the signature, because nobody else could have computed it.

1.2 Digital Signature Schemes

As we have just said, the idea behind a digital signature scheme is quite simple. A user has a secret key that he only knows, and a matching public key, known by everybody. In order to sign a message, this user applies to the message some function which depends on the secret key. The result of this function is the signature of the message. Later, anybody can use the public key to apply a verification algorithm to the signature and the message, checking in this way the correctness of the signature.

Usually, the message must be hashed into a value belonging to the domain of the signing function, before the computation of the signature. A *hash function* $H : \{0,1\}^* \to E$ takes as input a message $m \in \{0,1\}^*$, which is an arbitrarily long string of bits, and outputs an element in the appropriate domain E. To maintain the security of the signature scheme, the hash function must usually satisfy at least these two properties:

(i) given an output $b \in E$ it is computationally infeasible to find a message $m \in \{0,1\}^*$ such that $H(m) = b$ (*one-wayness*);

(ii) it is computationally infeasible to find two different messages $m, m' \in \{0,1\}^*$, $m \neq m'$, such that $H(m) = H(m')$ (*collision resistance*).

Now we explain a bit more formally what algorithms take part in a signature scheme.

Definition 1.1. *A signature scheme consists of three probabilistic polynomial time algorithms:*

- *Key-Gen: it takes as input a security parameter k, and outputs a pair (sk, pk), where sk is the secret key of the user, and pk is the matching public key.*
- *Sig: this algorithm takes as input a message m and the secret key sk, and produces a signature θ.*
- *Ver: finally, the verification algorithm takes as input a message m, a signature θ and the public key pk, and returns "true" if θ is a valid signature of m, and "false" otherwise.*

A signature scheme enjoys the *correctness* property if it satisfies the following condition: if $Key\text{-}Gen(k) = (sk, pk)$ and $Sig(m, sk) = \theta$, then $Ver(m, \theta, pk) = true$. In this case, we say that (m, θ) is a valid message-signature pair.

Now we explain two of the most popular signature schemes in the literature.

1.2.1 RSA Signature Scheme

Rivest, Shamir and Adleman proposed the first digital signature scheme in 1978 [85]. Some years later, Bellare and Rogaway [6] modified the original scheme with the inclusion of a hash function. The modified scheme, which was called FDH-RSA (Full Domain Hash), consists of the following protocols:

- *Key-Gen*: given a security parameter k, one must compute two prime k-bit numbers p and q. Let $n = pq$, and note that then we have $\phi(n) = (p-1)(q-1)$, where ϕ is the Euler's function. A random element e such that $\gcd(e, \phi(n)) = 1$ is chosen, and the value $d = e^{-1} \bmod \phi(n)$ is computed. A secure hash function $H : \{0,1\}^* \to \mathbb{Z}_n^*$ is chosen and made public.

  The secret key is $sk = (n, e, p, q, d)$, and the matching public key is $pk = (n, e)$.

- *Sig*: the signature of a message $m \in \{0,1\}^*$ is the value

$$\theta = H(m)^d \bmod n \ .$$

- *Ver*: to verify the correctness of a signature θ, one checks if $\theta^e = H(m) \bmod n$.

1.2.2 Schnorr's Signature Scheme

Schnorr proposed later a different method to obtain digital signatures [87]. The scheme goes as follows:

- *Key-Gen*: given a security parameter k, two prime numbers p and q are chosen, such that $q|p-1$ and such that q has k bits. An element g with order q in $\mathbb{Z}_p^*$ must also be chosen. A random value $x \in \mathbb{Z}_q^*$ is chosen, and the matching value $y = g^x \bmod p$ is computed. A secure hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is chosen and made public.

  The secret key of the user is $sk = (p, q, g, y, x)$, and his public key is $pk = (p, q, g, y)$.

- *Sig*: to sign a message $m \in \{0,1\}^*$, the user chooses a random $a \in \mathbb{Z}_q^*$ and computes $R = g^a \bmod p$. The signature is the pair $\theta = (R, \sigma)$, where
$$\sigma = a + x\ H(m, R) \ \bmod q\ .$$

- *Ver*: to verify the correctness of a signature $\theta = (R, \sigma)$, one checks if
$$g^\sigma = R\ y^{H(m,R)} \ \bmod p\ .$$

1.3 Security of Digital Signature Schemes

The concepts of negligible and polynomial functions appear when the security of cryptographic schemes is studied.

Definition 1.2. (Negligible function). *A function $f : \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible in k if, for every $c > 0$ there exists $k_0 \in \mathbb{N}$ such that $f(k) < \frac{1}{k^c}$, for all positive integer $k \geq k_0$. Otherwise, the function f is non-negligible in k.*

Definition 1.3. (Polynomial function). *A function $g : \mathbb{N} \rightarrow \mathbb{R}^+$ is polynomial in k if, for every $k_0 \in \mathbb{N}$ there exists a value $c > 0$ such that $f(k) < k^c$, for all positive integer $k \geq k_0$.*

Roughly speaking, the cryptographic schemes will be defined according to a security parameter k. We will consider such schemes secure if any adversary trying to attack them in polynomial time (in k) has a success probability which is a negligible function of k.

On the other hand, we say that an event has *overwhelming* probability with respect to k if the probability of its complementary is negligible in k.

To guarantee that a signature scheme provides authenticity and non-repudiation to digital communications, one must prove in some way that the scheme is secure. That is, only the owner of a secret key should be able to compute valid signatures with respect to the matching public key. This intuitive idea was formalized in [56], by considering an adversary which tries to break a signature scheme. The security of the scheme is defined according to the capabilities of this adversary, and its final goal.

With respect to its final goal, one can consider different levels of success for an adversary: to compute the secret key of a user; to find an efficient algorithm which emulates the signing algorithm of a user; to find a valid signature for a fixed message; to find a valid signature for some message.

With respect to the capabilities of the adversary, we list here some different situations: the adversary knows only the public key of the user; the adversary has access to valid signatures of a list of messages that it has not chosen; the adversary has access to valid signatures for messages that it can adaptively choose.

Signature schemes can achieve different levels of security. For example, a signature scheme can be proved to resist attacks whose goal is to compute the secret key, but on the other hand there can exist an attack against this scheme which finds a valid signature for some message.

Obviously, the maximum level of security for such a scheme consists of resisting attacks from an adversary with the most powerful capabilities (*adaptive chosen message attack*) but with the less ambitious goal (*existential forgery*, for some message). Nowadays, a signature scheme is considered secure (or unforgeable) only if it achieves this level of security.

Definition 1.4. (Unforgeability). *A signature scheme, with security parameter k, is unforgeable if no adversary which is given the public key and the signatures $\theta_1, \ldots, \theta_s$ of s messages $m_1, \ldots, m_s$ adaptively chosen by itself, can produce in polynomial time (in k) and with non-negligible probability (in k) a valid signature θ of some message m, such that $(m, \theta) \neq (m_i, \theta_i)$, for all $i = 1, \ldots, s$.*

Figure 1.1 gives an idea of what a successful forgery against a signature scheme is.

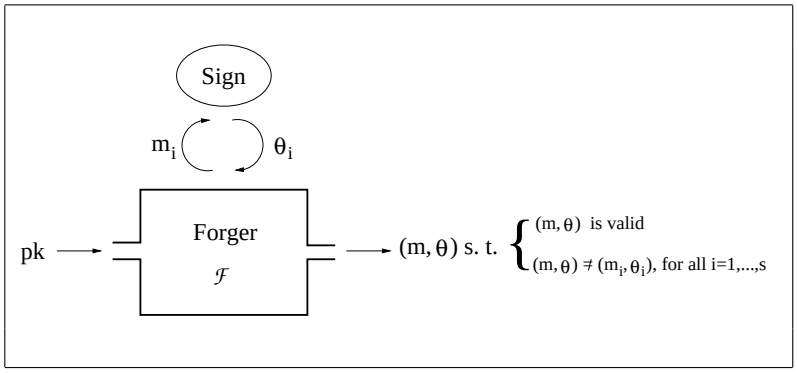

Figure 1.1: A successful forger against a signature scheme

The usual argument is to reduce the problem of forging a signature to a related computational problem. In other words, assuming the existence of a successful attack against the unforgeability of a scheme, one could solve the related problem. If this problem is assumed to be hard to solve, the reduction implies a contradiction, and one can conclude that the scheme is therefore unforgeable. Proving the unforgeability of a signature scheme in an absolute way, without such a reduction, seems to be a really hard problem.

However, constructing such a proof by reduction is not easy at all. The idea is to use the hypothetical existence of a successful adversary to solve an instance of the related computational problem. Roughly speaking, we receive an instance of the problem, and we try to set up the public parameters of the signature scheme in an ingenious way that allows:

1. to provide the adversary with valid signatures for messages that it adaptively chooses, when we execute (without knowing the secret key!) the hypothetical successful attack against the signature scheme; and then

2. to extract the solution of the problem from the signature forged by the adversary.

There exist very few proposals of signature schemes which can be proved secure in this formal (but restrictive) way. However, either the resulting schemes are not very efficient [56, 42, 27], or the security is based on stronger assumptions, like the Strong RSA Assumption, as it happens in the schemes proposed in [30, 51], or the q-Strong Diffie-Hellman Assumption, in the scheme of [11].

1.3.1 The Random Oracle Model

Bellare and Rogaway introduced in [5] a new paradigm that makes easier the issue of reducing the unforgeability of some signature schemes to the hardness of well-known computational problems. This paradigm is the *random oracle model*. In this model, hash functions are seen as oracles that produce a truly random value for each new input. Obviously, if the same input is asked twice, then the outputs must be identical.

The random oracle model is unreal, because any instantiation of a hash function is in fact a deterministic function: once the instantiation is made public, everybody can know which will be the output corresponding to any input. However, it is widely believed that proofs in this model guarantee the security of the overall signature scheme, provided the hash function has no weakness.

Despite there are some theoretical works which criticize the paradigm of the random oracle model [20, 78, 4], it has been adopted by the majority of the cryptographic community, because it allows to prove the security of many efficient schemes, like RSA or Schnorr's signature schemes. Furthermore, there are not known attacks against any practical or fully realistic scheme which is proved secure in the random oracle model.

1.3.2 Exact Security of Signature Schemes

When we study the exact security of a signature scheme in the random oracle model, we must consider the two following parameters:

- Q is the number of queries that the adversary can make to the random oracle which models the behaviour of the hash function H. If the hash function is denoted H_1, then we will use the notation Q_1 for the number of queries to the random oracle H_1, and so on.
- Q_s is the number of queries that the adversary can make to the signing oracle: the adversary chooses a message in an adaptive way (possibly depending on its previous choices and the received answers) and sends it to the oracle; as an answer, the adversary receives a valid signature for this message.

Definition 1.5. *We say that an adversary is a (T, ε, Q, Q_s)-forger against a signature scheme if:*

1. *it makes at most Q queries to the random oracle;*
2. *it makes at most Q_s queries to the signing oracle;*

3. *it runs in time at most T;*
4. *with probability at least ε, it obtains a new valid pair (message,signature), different from the pairs obtained in the queries to the signing oracle.*

Without loss of generality, we can assume that a successful (T, ε, Q, Q_s)-forger against a signature scheme always satisfies the condition $Q \geq 1$. Otherwise, since we assume that the hash function behaves as a random function, the forger should have guessed the value of the hash function corresponding to the forged signature, which is very unlikely.

Definition 1.6. *(Exact unforgeability of a signature scheme). We say that a signature scheme is (T, ε, Q, Q_s)-unforgeable, in the random oracle model, if there does not exist any (T, ε, Q, Q_s)-forger against it.*

The goal when one designs a signature scheme is to prove that it is (T, ε, Q, Q_s)-unforgeable, for any polynomial value of T and any non-negligible value of ε (with respect to the security parameter of the scheme).

As we have said before, it seems almost impossible to prove an absolute result. Therefore, the usual strategy is to reduce the problem of forging a signature to a difficult computational problem. For example, to prove that if there exists a (T, ε, Q, Q_s)-forger against our signature scheme, then the factorization of a large integer can be found in time at most T' and with probability at least ε', where T' and ε' are functions related to the parameters T, ε, Q, Q_s. A constant that will appear repeatedly in these relations is T_{exp}, a bound on the time needed to compute a modular exponentiation.

According to the relation between T', ε' and T, ε, Q, Q_s, the quality of the reduction will be better or worse.

Let us see a simple example of a bad reduction.

A Simple Example

Suppose that we design a signature scheme with security parameter k and we prove the following "theorem": if there exists a (T, ε, Q, Q_s)-forger against this scheme, then we construct an algorithm that factorizes a k-bit integer n, product of two primes, within time $T' \leq Q_s T$ and with probability $\varepsilon' \geq \varepsilon/Q$. The resulting expected time T'_{exp} to factorize with this method would be obtained by repeating $O(1/\varepsilon')$ times the forger. In this case, the expected time would be $T'_{exp} \leq \frac{QQ_sT}{\varepsilon}$.

Currently, the most efficient algorithm to factor such an integer n is the Number Field Sieve (NFS) method [71], which has a super-polynomial, but sub-exponential, expected time complexity

$$O\left(\exp((1.923 + o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3})\right).$$

For example, this means that with this method we can factor an integer of $2^{10} = 1024$ bits within expected time less than 2^{80}.

Table 1.1 compares the resulting times of the NFS method with the factorization method that we would obtain by reduction of the forger against the signature scheme, if we consider that the forger can make $Q = 2^{55}$ queries to the random oracle, and $Q_s = 2^{30}$ queries to the signing oracle. We consider two situations: in the first one the forger runs in polynomial time $T_1 = k^3$, whereas in the second one, its running time is $T_2 = k^6$. In both situations, we suppose that the success probability of the forger is $\varepsilon \geq 2^{-10}$.

Security parameter = Number of bits of n	**NFS**	**Our method**	
		$T_1 = k^3$	$T_2 = k^6$
$k = 2^{10} = 1024$	2^{80}	2^{125}	2^{155}
$k = 2^{11} = 2048$	2^{111}	2^{128}	2^{161}
$k = 2^{12} = 4096$	2^{149}	2^{131}	2^{167}
$k = 2^{13} = 8192$	2^{201}	2^{134}	2^{173}

Table 1.1: Comparing the times to factor n

The table shows that our security "theorem" for the signature scheme would have no practical effects for security parameters $k = 2^{10}$ and 2^{11} (i.e. working with numbers of 1024 or 2048 bits). In effect, if we assume that the signature scheme can be forged in time $T_1 = k^3$ or $T_2 = k^6$, then we would obtain a method for factoring a k-bit integer; but this method is slower than some known methods, so this fact does not give us a contradiction.

If we want to protect the signature scheme against forgers running in time less than $T_1 = k^3$, then the security parameter must be at least $k = 2^{12}$ (working with integers of 4096 bits). For this case, an hypothetical attack against the scheme would derive in a method for factoring, which is faster (running time 2^{131}) than the best known algorithm (running time 2^{149}). Therefore, we obtain a contradiction and can conclude that the signature scheme can not be attacked by adversaries running in time less than $T_1 = k^3$, and with success probability greater than 2^{-10}.

Analogously, if we want to protect the scheme against adversaries running in time less than $T_2 = k^6$, we must consider at least $k = 2^{13}$ (which means working with 8192-bit integers). And so on...

Summing up, the tighter the relation between T', ε' and T, ε, Q, Q_s is, the more practical the studied signature scheme will be, in the sense that it will achieve a real level of security with smaller values of the security parameter (and so, with smaller lengths for the integers that are used in the

scheme). A reduction is said to be *tight* when $T'/\varepsilon' \approx T/\varepsilon$; in this case, it is said to be *very tight* if it also satisfies $\varepsilon' \approx \varepsilon$ and $T' = T + \lambda(Q, Q_s)$, where λ is linear in the variables Q and Q_s. An interesting work about signature schemes with tight reductions is [70].

1.3.3 Computational Assumptions

We define here three computational assumptions related to the hardness of some mathematical problems. The security of some of the schemes proposed in this thesis is based on these assumptions.

Definition 1.7. (RSA Assumption). *Let $n = pq$, where p and q are two k-bit prime numbers. Let e be a random element such that $\gcd(e, \phi(n)) = 1$. Let y be a random element in $\mathbb{Z}_n^*$.*

Then, for any probabilistic polynomial time algorithm $\mathcal{F}$, the probability that $\mathcal{F}(n, e, y) = x$ such that $x^e = y \bmod n$ (that is, x is a solution of the RSA problem) is a negligible function in k. (The probability is taken over the randomness in the choice of p, q, e and y, and the random choices that $\mathcal{F}$ makes.)

Definition 1.8. (Discrete Logarithm Assumption). *Let p and q be two prime numbers such that q has k bits and such that $q|p-1$. Let g be a random element with order q in $\mathbb{Z}_p^*$. Let y be a random element in $\langle g \rangle$.*

Then, for any probabilistic polynomial time algorithm $\mathcal{F}$, the probability that $\mathcal{F}(p, q, g, y) = x$ such that $g^x = y \bmod p$ (that is, x is a solution of the Discrete Logarithm problem) is a negligible function in k. (The probability is taken over the randomness in the choice of p, q, g and y, and the random choices that $\mathcal{F}$ makes.)

Definition 1.9. (Computational Diffie-Hellman Assumption in an additive group). *Let q be a prime number such that q has k bits. Let $\mathbb{G}$ be an additive group of order q, generated by some element P. Let a, b be random elements in $\mathbb{Z}_q^*$.*

Then, for any probabilistic polynomial time algorithm $\mathcal{F}$, the probability that $\mathcal{F}(q, \mathbb{G}, P, aP, bP) = abP$ (that is, abP is a solution of the Computational Diffie-Hellman problem) is a negligible function in k. (The probability is taken over the randomness in the choice of q, $\mathbb{G}$, P, a and b, and the random choices that $\mathcal{F}$ makes.)

Note that the Computational Diffie-Hellman problem can be defined, as well, in multiplicative groups such as $\langle g \rangle \subset \mathbb{Z}_p^*$, where g is an element with

order q in $\mathbb{Z}_p^*$. However, in this thesis we use the additive notation when we design schemes whose security is based on the Computational Diffie-Hellman Assumption.

1.3.4 Security of RSA Signature Scheme

First of all, we explain why the original RSA signature scheme proposed in [85], without the use of a hash function, does not achieve the highest level of security. Given a secret key (p, q, d) matching with a public key (n, e), the signature of a message $m \in \mathbb{Z}_n^*$ is $\theta = m^d \bmod n$.

We explain how it is possible to forge a signature for m with a chosen message attack. Such an attacker can choose $\tilde{m} \in \mathbb{Z}_n^*$ and ask for a valid RSA signature of the message $M = m\tilde{m}^e \bmod n$. It obtains the value $M^d = m^d\tilde{m} \bmod n$, and it can obtain a valid signature $m^d = M^d\tilde{m}^{-1} \bmod n$ for message m.

As an example of the proofs by reduction in the random oracle model, we review the security proof of the FDH-RSA signature scheme explained in Section 1.2.1, with the inclusion of a hash function. The proof uses some techniques introduced by Coron in [26], which improve the original security result provided by Bellare and Rogaway in [6]. The result is the following:

Theorem 1.1. *If there exists a (T, ε, Q, Q_s)-forger against FDH-RSA signature scheme, then the RSA problem can be solved in time $T' \leq T + T_{exp}Q + Q_s$ and with probability $\varepsilon' \geq \frac{\varepsilon}{\mathbf{e}\cdot(Q_s+1)}$ (in the random oracle model, where $\mathbf{e}$ is the Euler's value $\mathbf{e} = 2,718281828459...$).*

Proof. We denote by $\mathcal{A}$ the (T, ε, Q, Q_s)-forger against FDH-RSA signature scheme. Let (n, e, y) be an input of the RSA problem (see Definition 1.7); the solution of the problem is the only element $x \in \mathbb{Z}_n^*$ such that $x^e = y \bmod n$.

We use $\mathcal{A}$ to construct a machine $\mathcal{F}$ that solves this instance of the RSA problem. This machine maintains a table TAB that is constructed during the interaction between $\mathcal{F}$ and $\mathcal{A}$.

The machine $\mathcal{F}$ executes the forger $\mathcal{A}$, providing him with the RSA public key (n, e). We can suppose, without loss of generality, that $\mathcal{A}$ asks for the hash value $H(m)$ before querying a valid signature for m.

Every time $\mathcal{A}$ asks for a hash value $H(m_i)$, the machine $\mathcal{F}$ looks for m_i in the table TAB. If it is already there, $\mathcal{F}$ returns the existing relation. Otherwise, $\mathcal{F}$ chooses a random value $z_i \in \mathbb{Z}_n^*$ (different from the previously chosen z_j's). Let μ be a value such that $0 < \mu < 1$. Then, with probability μ, the machine $\mathcal{F}$ chooses $c_i = 0$, whereas with probability $1 - \mu$, it chooses $c_i = 1$. It returns to $\mathcal{A}$ the relation $H(m_i) = y^{c_i} z_i^e \bmod n$, and stores the

values $(m_i, c_i, z_i, H(m_i))$ in a new entry of the table TAB. Since z_i is random and we are assuming that H behaves as a random function, this relation is consistent.

When $\mathcal{A}$ asks for a valid signature of the message m_i, the machine $\mathcal{F}$ looks at the value c_i. If $c_i = 1$, the machine $\mathcal{F}$ halts; if $c_i = 0$, it outputs z_i as the valid signature of m_i. Note that the probability that $\mathcal{F}$ does not halt in one of these steps is at least μ^{Q_s}, because $\mathcal{A}$ is assumed to ask for at most Q_s valid signatures.

By hypothesis, $\mathcal{A}$ obtains in time T and with non-negligible probability ε a valid RSA signature θ of a message m different from the messages queried to the signing oracle. Since we are in the random oracle model, $\mathcal{A}$ must have queried the message m to the random oracle with overwhelming probability. Therefore, we have that $m = m_j$ for some entry $(m_j, c_j, z_j, H(m_j))$ of the table TAB.

With probability $1 - \mu$, we have that $c_j = 1$. This means that $H(m) = H(m_j) = yz_j^e \bmod n$. But, since θ is a valid signature for m, we have also that $\theta^e = H(m) \bmod n$. Putting the two equalities together, we obtain:

$$y = \left(\theta \cdot z_j^{-1}\right)^e \bmod n.$$

Therefore, the machine $\mathcal{F}$ solves the given instance of the RSA problem, in time $T' \leq T + Q + Q_s$. The success probability of the machine $\mathcal{F}$ is $\varepsilon' \geq \varepsilon \cdot \mu^{Q_s} \cdot (1 - \mu)$. Now we can choose the value of μ that maximizes ε', which is $\mu = \frac{Q_s}{Q_s+1}$. This gives us a success probability

$$\varepsilon' \geq \varepsilon \cdot \left(\frac{Q_s}{Q_s + 1}\right)^{Q_s} \cdot \frac{1}{Q_s + 1} \geq \frac{\varepsilon}{\mathbf{e} \cdot (Q_s + 1)} \ ,$$

where $\mathbf{e} = 2,718281828459...$

□

Although the relation $T' \leq T + Q + Q_s$ can seem very tight, we must note the following: when considering methods to solve a problem, such as factorization, the RSA problem, etc., one studies the expected time in solving the problem. Our reduction method gives us an execution time T' and a success probability ε'. To compute the expected time needed to solve the RSA problem, in this case, we should repeat the experiment $O(1/\varepsilon')$ times. The expected time T_{exp} will be, then:

$$T_{exp} \leq T'/\varepsilon' \leq \frac{\mathbf{e} \cdot (Q_s + 1) \cdot (T + Q + Q_s)}{\varepsilon}.$$

Therefore, the reduction is not so tight as it seemed, but it is quite good.

1.3.5 Security of Schnorr's Signature Scheme

The unforgeability of Schnorr's signature scheme was proven, in the random oracle model, by Pointcheval and Stern in [83], by reduction to the Discrete Logarithm problem. Recall that the security parameter k is such that the prime q has k bits. They prove the following result:

Theorem 1.2. *If there exists a (T, ε, Q, Q_s)-forger against Schnorr's signature scheme, with $\varepsilon \geq \frac{10(Q_s+1)(Q_s+Q)}{q}$ (otherwise, the success probability ε would be negligible in k), then the Discrete Logarithm problem can be solved in expected time $T' \leq \frac{120686QT}{\varepsilon}$ (in the random oracle model).*

We do not give the proof in detail here, only some ideas behind it. The proof uses the technique of the *oracle replay attack*. The formalizations of this technique are known as *forking lemmas*, and they can be applied to a family of signature schemes that the authors call *generic*, and which includes Schnorr's scheme, and a modification of ElGamal's signature scheme [43]. The idea is to prove that if an adversary can forge a signature for such a scheme, then one can obtain, by replaying this attack, another forged signature of the same message and with the same randomness, but under a different instantiation of the random oracle model.

Applying this result to Schnorr's scheme, we obtain the following: let us denote a valid Schnorr signature of a message m as (R, h, σ), where $h = H(m, R)$ and $g^\sigma = Ry^h \bmod p$. The forking lemma asserts that, if an adversary can obtain, after an adaptive chosen message attack, a valid forgery (R, h, σ) for some message m, then we can replay this attack with the same random tape but a different instantiation of the random oracle, and obtain with non-negligible probability a new valid signature (R', h', σ') of the same message m, with the same randomness $R' = R$, but different random oracles (or hash functions) H and H' such that $h = H(m, R) \neq H'(m, R) = h'$. The most intricate point is to properly compute bounds on the necessary running time and the success probability for obtaining such two forged signatures.

Once this result is proven, it is not difficult to reduce the unforgeability of Schnorr's scheme to the Discrete Logarithm problem, taking into account as well that valid Schnorr's signatures can be perfectly simulated in the random oracle model, without knowing the secret key.

If (p, q, g, y) is an input of the Discrete Logarithm problem (recall Definition 1.8), then we execute the hypothetical attack against Schnorr's signature scheme with public key (p, q, g, y). We simulate consistent answers to the queries of the adversary, asking for valid signatures and outputs of the random oracle. By hypothesis, the adversary obtains a valid signature

(R, h, σ) on a message m. Applying the forking lemma, we will obtain a different valid signature (R, h', σ') on the same message. These two signatures satisfy $g^{\sigma} = Ry^h \bmod p$ and $g^{\sigma'} = Ry^{h'} \bmod p$, respectively. Dividing these two equations, we obtain $g^{\sigma-\sigma'} = y^{h-h'} \bmod p$, and therefore the solution to the Discrete Logarithm problem is $\frac{\sigma-\sigma'}{h-h'} \bmod q$.

1.4 Secret Sharing Schemes

Secret sharing schemes are an essential tool to work with distributed cryptographic systems, where the power to perform a secret task is distributed among different players. These schemes were introduced independently by Shamir [88] and Blakley [7] in 1979.

Let $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ be a set of ℓ players. In this set of players, a family of authorized or qualified subsets $\Gamma \subset 2^{\mathcal{P}}$ must be 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 $A_2 \in \Gamma$. Because of this property, an access structure is fully determined by its basis $\Gamma_0 = \{A \in \Gamma \mid A - \{P_i\} \notin \Gamma, \text{ for all } P_i \in A\}$.

For an arbitrary family of subsets $\mathcal{U} \subset 2^{\mathcal{P}}$ the *closure* of $\mathcal{U}$ is the minimum monotone access structure that contains $\mathcal{U}$, that is $cl(\mathcal{U}) = \{A \subset \mathcal{P} : \text{ there exists } B \in \mathcal{U} \text{ such that } B \subset A\}$. Of course for a monotone access structure Γ we have $\Gamma = cl(\Gamma_0)$.

Given a monotone increasing access structure Γ and a secret to be shared, the idea behind a secret sharing scheme is that each player of the set $\mathcal{P}$ receives from a trusted authority (the *dealer*, usually denoted by D) a share of the secret. From the shares of any authorized subset, in Γ, it is possible to recover the secret. However, a non-authorized subset, not in Γ, cannot obtain any information about the secret from their shares.

Shamir proposed in [88] a *threshold* scheme, where subsets that can recover the secret are those with at least t members (t is the threshold), or in other words, the access structure is $\Gamma = \{A \subset \mathcal{P} : |A| \geq t\}$ (such structures are called (t, ℓ)-threshold access structures).

In order to share a secret s in a finite field K with $|K| > \ell$, the dealer chooses a random polynomial $f(z) = s + a_1 z + \cdots + a_{t-1} z^{t-1} \in K[z]$ of degree $t-1$ and sends to the participant P_i his secret share $s_i = f(i)$, for $i = 1, \ldots, \ell$.

Let $A = \{P_{i_1}, \ldots, P_{i_t}\}$ be a qualified subset of t participants who want to recover the secret s. They have t different values of the polynomial $f(z)$, of degree $t-1$, so they can obtain the value $s = f(0) = \sum_{P_i \in A} \lambda_{0,i}^A f(i)$, where $\lambda_{0,i}^A$ are the Lagrange interpolation coefficients. It can also be proved that

any subset of less than t participants cannot obtain any information about the secret from the shares they hold.

Other works have proposed schemes realizing more general access structures, such as *vector space secret sharing schemes* [17]. An access structure Γ is realizable by such a scheme, defined in a finite field K, if there exist a positive integer r and a map $\psi : \mathcal{P} \cup \{D\} \longrightarrow K^r$ such that $A \in \Gamma$ if and only if $\psi(D) \in \langle \psi(P_i) \rangle_{P_i \in A}$. In this case, we say that Γ is a *vector space access structure*.

If a dealer wants to distribute a secret value $s \in K$, he takes a random vector $\mathbf{v} \in K^r$, such that $\mathbf{v} \cdot \psi(D) = s$. The share of a participant $P_i \in \mathcal{P}$ is $s_i = \mathbf{v} \cdot \psi(P_i) \in K$. 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 $\lambda_i^A \in K$. In order to recover the secret, the players of A compute

$$\sum_{P_i \in A} \lambda_i^A s_i \;=\; \sum_{P_i \in A} \lambda_i^A \mathbf{v} \cdot \psi(P_i) \;=\; \mathbf{v} \cdot \sum_{P_i \in A} \lambda_i^A \psi(P_i) \;=\; \mathbf{v} \cdot \psi(D) \;=\; s \;.$$

Note that (t, ℓ)-threshold access structures are in particular vector space ones, and Shamir's threshold secret sharing scheme can be seen as a vector space one, by taking $r = t$, $\psi(D) = (1, 0, \ldots, 0)$ and $\psi(P_i) = (1, i, i^2, \ldots, i^{t-1})$.

On the other hand, vector space secret sharing schemes can also be generalized. Simmons, Jackson and Martin [92] introduced *linear secret sharing schemes*, that can be seen as vector space secret sharing schemes in which each player is (possibly) associated with more than one vector. They proved that any access structure can be realized by a linear secret sharing scheme. In general, the construction that they proposed results in an inefficient secret sharing scheme, in the sense that its *information rate* is very small.

Informally, the information rate of a secret sharing scheme is the quotient between the length of the secret (in bits) and the maximum length of the distributed shares. This quantity is at most 1, which is the *ideal* case. This happens for example in Shamir's secret sharing schemes or in vector space ones, because all the shares and the secret belong to the same finite field K, so they all have the same length.

1.4.1 Dual Access Structures

For an access structure $\Gamma \subset 2^{\mathcal{P}}$, the *dual* of Γ is defined as $\Gamma^* = \{\mathcal{P} - A : A \notin \Gamma\}$ and it is also a monotone access structure (see [69] for more details on dual access structures). A basic property of the dual is that $(\Gamma^*)^* = \Gamma$; it is also easy to see, by the definition of Γ^*, that $A \in \Gamma_0$ if and only if $\mathcal{P} - A$ is a maximal subset verifying $\mathcal{P} - A \notin \Gamma^*$.

It is not difficult to prove that, if Γ is a vector space access structure, then Γ^* is also a vector space access structure (see [69], for example).

In fact, the construction that Simmons, Jackson and Martin presented in [92] to realize any possible access structure Γ is based on the use of the dual access structure and it is as follows. Let us suppose that the structure Γ is such that $(\Gamma^*)_0 = \{A_1, \ldots, A_d\}$, then ψ assigns vectors in $GF(q)^d$ in the following way: $\psi(D) = (1, 0, \ldots, 0)$ and $\psi(U) = \{(1, i, i^2, \ldots, i^{d-1}) : U \in A_i\}$ for any user $U \in \mathcal{P}$. This assignment ψ realizes the access structure Γ.

Chapter 2

RSA Distributed Signature Schemes

Distributed public key cryptography deals with scenarios where a cryptographic secret task is performed by a collective of users instead of an individual user. In this way, the systems win in security and reliability.

An important point in these schemes is to fix which subsets of users are authorized to perform the secret task, and which subsets of users can be corrupted by an attacker without compromising the security of the scheme. These families are respectively known as the access and the adversary structures. Most of the proposed distributed schemes consider threshold access (resp. adversary) structures, which contain all the subsets with a minimum (resp. maximum) number of users.

A usual strategy to design distributed cryptographic schemes is to use a secret sharing scheme (or a similar technique) to distribute shares of the secret key of a known individual cryptographic scheme. However, this is not always as easy or direct as it can seem. Furthermore, depending on the access and adversary structures to be considered, one must use the appropriate secret sharing scheme.

In this chapter we study distributed signature schemes, where an authorized subset of users must cooperate in order to compute a valid signature on a message. We give the basic definitions and the security requirements for these schemes, and we cite some existing works (most of them considering only threshold structures). We also recall the basic concepts related to secret sharing schemes.

Later, we focus on RSA distributed signature schemes. We extend an existing RSA threshold signature scheme to a scenario where the access and the adversary structures can be more general families. We state the algebraic and combinatorial conditions that these structures must satisfy; when this is

the case, the proposed scheme is proved to be robust and secure.

2.1 Distributed Signature Schemes

In some situations, the power to generate valid signatures is shared among a set of users or computers (we will refer to them also as *players* or *participants*). For example, in a big company, it is not convenient that a single player has the knowledge of the secret key and signs all the messages on behalf of the company. This fact decreases security, because an adversary must attack only a single point to obtain the full secret information. It also decreases reliability, because the signature system remains inoperative if the player has some technical problem.

A solution to solve these problems is to distribute the power of signing among a set $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ of ℓ players, where a monotone increasing family of authorized or qualified subsets $\Gamma \subset 2^{\mathcal{P}}$ must be defined. This family will be called the access structure of the system.

Each player will have a share of the secret key of the set. To compute a signature, each player of an authorized subset will use his secret share to compute a partial signature. Finally, a combining algorithm will convert the partial signatures from the subset into a valid standard signature of the message, that can be verified by using the single public key which matches with the shared secret key. In next section we define distributed signature schemes a bit more formally.

2.1.1 Basic Definitions and Security Requirements

Definition 2.1. *A $(\mathcal{P}, \Gamma)$-distributed signature scheme consists of three polynomial time and probabilistic protocols:*

- *Dist-Key-Gen: this protocol can be executed jointly by the ℓ players themselves, or by a trusted and external authority. The input is a security parameter. The public outputs are pk (the public key of the scheme) and some verification key vk, whereas the private output of each player P_i is a share sk_i of the secret key sk related to pk.*

- *Dist-Sig: if m is the message to be signed, each player P_i uses his private information to compute and broadcast his partial signature $\theta_i(m)$. The correctness of the partial signatures can be verified using the verification key vk. Finally, a combiner algorithm takes valid partial signatures corresponding to an authorized subset $A \in \Gamma$ and produces from $\{\theta_i(m)\}_{P_i \in A}$ a valid standard signature $\theta(m)$.*

- *Ver: this protocol is executed by the recipient of the signature. The inputs are the public key pk, the message m and the signature $\theta(m)$. The output will be "true" or "false".*

Usually, distributed signature schemes are constructed by starting from a standard (individual) signature scheme, such as RSA or Schnorr. In that cases, it is desirable that a final distributed signature has the same form as an individual signature (and therefore, the verification protocols must be identical, as well). Thus, the recipient of the signature cannot distinguish how the signature has been generated; this is an interesting property in some scenarios requiring privacy and anonymity.

A distributed signature scheme must be secure even in the presence of an adversary who corrupts and controls the behaviour of some subset of dishonest players. The family of subsets of players that the system tolerates to be corrupted by an adversary is commonly known as the *adversary structure* and denoted by $\Lambda \subset 2^{\mathcal{P}}$. It must be monotone decreasing (if the scheme remains secure when an attacker corrupts a subset $B_1 \in \Lambda$, then it must also remain secure if an attacker corrupts players of B_2, for any subset $B_2 \subset B_1$). This implies that the adversary structure Λ is determined by its basis $\Lambda_0 = \{B \in \Lambda \mid B \cup \{P_i\} \notin \Lambda, \text{ for all } P_i \notin B\}$.

Informally, we say that a $(\mathcal{P}, \Gamma)$-distributed signature scheme is Λ-secure if it is *robust* and *existentially unforgeable under adaptive chosen message attacks*, considering an adversary allowed to corrupt players of any subset in the structure Λ. Now we explain with more detail these two security properties.

By robustness we refer to the fact that the scheme provides mechanisms to detect corrupted players who do not follow the protocol correctly. Furthermore, the protocol must always produce a valid signature from the partial signatures of the honest players.

On the other hand, existential unforgeability under adaptive chosen message attacks is defined now in relation to the following game (or challenge) $\mathcal{G}_1$:

1. If a hash function is assumed to behave as a random function, the adversary can make Q queries to the random oracle which models this function.

2. The adversary is given a set of players $\mathcal{P}$, a monotone increasing access structure $\Gamma \subset 2^{\mathcal{P}}$ and an adversary structure $\Lambda \subset 2^{\mathcal{P}}$.

3. The adversary chooses a subset of players $B \in \Lambda$ to corrupt.

4. The *Dist-Key-Gen* protocol is executed. The adversary obtains all the information that is made public in the execution of this protocol, as well as private information corresponding to the corrupted players $P_j \in B$ (in particular, their shares sk_j of the secret key sk).

5. The adversary can adaptively choose Q_s messages to be signed. For these messages, the *Dist-Sig* protocol is executed. The adversary sees all the information that is made public in the required executions of this protocol, as well as private information corresponding to the corrupted players.

Definition 2.2. *Such an adversary is a $(\mathcal{P}, \Gamma, \Lambda, T, \varepsilon, Q, Q_s)$-forger against a distributed signature scheme if its total running time is at most T and it obtains, with probability ε, a valid (message,signature) pair, different from the ones that it has received during the game $\mathcal{G}_1$.*

Definition 2.3. *(Exact unforgeability of distributed signature schemes). A distributed signature scheme is $(\mathcal{P}, \Gamma, \Lambda, T, \varepsilon, Q, Q_s)$-unforgeable if there does not exist any $(\mathcal{P}, \Gamma, \Lambda, T, \varepsilon, Q, Q_s)$-forger against it.*

Note that we consider only *static* adversaries: they must choose the subset of players $B \in \Lambda$ they want to corrupt before the signature scheme is initialized, and this corrupted set does not vary throughout the life of the system. To achieve security against *adaptive* adversaries [19], one can use known techniques such as proactive schemes [66, 48, 84].

Usually, unforgeability of a distributed signature scheme is proved by reducing it to the unforgeability of the regular signature scheme from which the distributed one has been constructed. The most delicate point is to simulate the secret information that the adversary obtains during the execution of the *Dist-Key-Gen* protocol and the Q_s executions of the *Dist-Sig* protocol.

2.1.2 State of the Art

First works on distributed cryptography only considered threshold access structures, where all the players have the same power inside the set, and the same susceptibility of being corrupted by an adversary.

With respect to RSA threshold signatures, the first schemes [39, 38, 52] used cyclotomic extensions of the integers to solve the main problem: to share an RSA secret key $d \in \mathbb{Z}_{\phi(n)}$ in such a way that the ring $\mathbb{Z}_{\phi(n)}$ remains secret to the players. Shoup proposed in [91] a more efficient and simple solution, which has been studied and extended in [31, 47].

In all these schemes, the key generation phase is executed by an external and trusted dealer. There are some works which propose protocols where the own players jointly execute the algorithm *Dist-Key-Gen* for the RSA case, generating the public key (n, e) and secret shares of the secret key (p, q, d) for each player. In these protocols (see [12, 49, 21]), players must compute shares of the product of two shared secret values. This problem is known as the *problem of the multiplication*, and is efficiently solved only in the case of threshold structures and a few more particular families of access structures (see [33], for example).

With respect to the distributed versions of other signature schemes, for example those whose security is based on the Discrete Logarithm problem, the solutions are not so complicated, because the secret keys that must be shared belong to a field (in Schnorr's scheme, $\mathbb{Z}_q$) which is public. A threshold version of Schnorr's signature scheme can be found in [93], and a threshold version of Digital Standard Signature (DSS) scheme can be found in [53]. For this kind of schemes, the generation of the keys can be jointly executed by the players, by following the protocol explained in [54] for the threshold case.

Another example of threshold signature scheme can be found in [10]. Its security is based on the Computational Diffie-Hellman Assumption.

2.2 RSA Distributed Signature Schemes for General Access Structures

Most of the proposals of distributed cryptographic schemes (in particular, distributed signature schemes) consider threshold structures: the access structure is $\Gamma = \{A \subset \mathcal{P} : |A| \geq t\}$, whereas the adversary structure is $\Lambda = \{B \subset \mathcal{P} : |B| < t\}$. However, this corresponds to a very particular situation. In the real life, players of a distributed cryptographic scheme usually have different levels of importance: they can have different privileges or computational resources, and enjoy different levels of protection against possible attacks, for example.

For this reason, it is important to design distributed schemes that work properly in the case of general access and adversary structures, not only in the threshold case. A logical strategy consists in trying to extend to the general case some existing threshold scheme, whenever it is possible. This is what we have done with the RSA threshold signature scheme proposed by Shoup in [91]. In this section, we study the conditions that must be satisfied in order to extend this protocol to the case of general structures, in such a

way that the resulting scheme is secure.

We will consider a general access structure $\Gamma \subset 2^{\mathcal{P}}$, where $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, for the collection of subsets of participants that will be able to generate a valid signature. And we will consider a general adversary structure $\Lambda \subset 2^{\mathcal{P}}$, containing the subsets of players that an adversary could corrupt without breaking the security of the scheme.

We introduce some notation: for any family $\Theta \subset 2^{\mathcal{P}}$ of subsets of $\mathcal{P}$, we define the family $\overline{\Theta} = 2^{\mathcal{P}} - \Theta$ (containing those sets which are not in Θ) and the family $\Theta^c = \{\mathcal{P} - B \ : \ B \in \Theta\}$ (containing the complementary sets of the sets in Θ).

An obvious necessary condition if we want a distributed signature scheme to achieve $(\mathcal{P}, \Gamma, \Lambda)$-unforgeability, is that any subset in Λ cannot be in Γ; that is $\Lambda \cap \Gamma = \emptyset$. This condition is the same as $\Gamma \subset \overline{\Lambda}$, or equivalently, $\Lambda \subset \overline{\Gamma}$, where $\overline{\Gamma}$ is the family of non-authorized subsets. Since $\overline{\Gamma}$ is monotone decreasing, we can consider the family $(\overline{\Gamma})_0 \subset \overline{\Gamma}$ of maximal non-authorized subsets.

To achieve robustness, for any possible set of corrupted players $B \in \Lambda$, the subset formed by the rest of players of $\mathcal{P}$ must be able to sign; that is, $\mathcal{P} - B \in \Gamma$. In other words, $\Lambda^c \subset \Gamma$.

In conclusion, to design a secure distributed signature scheme it is necessary that the structures Λ and Γ satisfy $\Gamma \subset \overline{\Lambda}$ and $\Lambda^c \subset \Gamma$, which can be written as $\Lambda^c \subset \Gamma \subset \overline{\Lambda}$. In particular, this implies that Λ must be a $\mathcal{Q}^2$ structure.

Definition 2.4. *($\mathcal{Q}^2$ structures, [67]). A monotone decreasing structure $\Lambda \subset 2^{\mathcal{P}}$ is said to be $\mathcal{Q}^2$ in $\mathcal{P}$ if $\Lambda^c \subset \overline{\Lambda}$ (or equivalently, if there are not two subsets in Λ that cover all the set $\mathcal{P}$).*

For example, in the threshold case the $\mathcal{Q}^2$ condition is equivalent to $\ell \geq 2t + 1$.

2.2.1 Previous Considerations

Before describing the protocols of the resulting RSA distributed signature scheme for general access structures, we must explain some useful tricks. Namely, we will use a modification of standard secret sharing schemes to work with secrets that belong to groups which are unknown to the players. Furthermore, we must introduce an integer Δ that will be necessary to ensure the proper and secure running of the scheme.

In RSA distributed signature schemes, if $n = pq$ is the public modulus, the secret d to be shared will belong to a group $\mathbb{Z}_{\phi(n)}$ (or $\mathbb{Z}_\eta$ in our scheme, where $\phi(n) = 4\eta$) that must remain secret to the participants. One possible

approach to solve this problem is to use a *black-box* secret sharing scheme, which allows to share secrets belonging to any (possibly unknown for the participants) Abelian group. However, efficient constructions of such schemes are only known for the threshold case (see [29]).

Our approach, following the idea introduced by Shoup in [91], consists in modifying the framework of standard linear secret sharing schemes in order to adapt them to our needs. For simplicity, we will consider the vector space case, with a map ψ that assigns one vector to each player; but everything can be easily extended to the (general) linear case where players can receive more than one vector.

Definition 2.5. *We say that a function $\psi : \mathcal{P} \cup \{D\} \longrightarrow \mathbb{Z}^r$ realizes Γ over the rationals when $A \in \Gamma$ if and only if there exists $\{c_i^A\}_{i \in A}$, with $c_i^A \in \mathbb{Q}$, such that $\psi(D) = \sum_{P_i \in A} c_i^A \psi(P_i)$. Furthermore, we say that such a function ψ is Λ-independent if for all subset of players B in the adversary structure Λ, the vectors in $\{\psi(P_j)\}_{P_j \in B}$ which are different are linearly independent.*

Note that using the technique presented by Simmons, Jackson and Martin in [92] we can find, for any access structure Γ, a Λ-independent function realizing Γ, where $\Lambda = \overline{\Gamma}$. So the scheme that we will propose can be used with any access structure. The construction of [92], however, can result in a very inefficient scheme in some cases, in terms of the number of vectors (and so the number of shares) assigned to each participant.

Let us return to RSA distributed signatures. If a dealer would use such a function ψ, projected into the corresponding ring $\mathbb{Z}_\eta$, to distribute shares of the secret key $d \in \mathbb{Z}_\eta$, then the following problem would appear: if an authorized subset would like to compute a valid RSA signature $H(m)^d \bmod n$ of a message m (where H is a hash function), its members would have to recover the secret key d in the exponent of an expression in $\mathbb{Z}_n$. If the coefficients of the combination that allows to compute d from their shares were rational, this step could not be efficiently done, because the participants would not be able to compute roots modulo n. The solution is to find a factor Δ that cancels all these denominators in the coefficients, for all the authorized subsets of players. Each player will multiply his share by this public factor Δ. Note that, then, an authorized subset will obtain in the exponent a multiple of the secret key d. But this fact does not affect the computation of a valid RSA signature, as we will see later.

For the security proofs of the resulting scheme, we also need that the factor Δ cancels the denominators in the coefficients of the following linear combinations: the vector of a player P_i written as a combination of the vector of the dealer and the vectors corresponding to a subset of players $B \in (\overline{\Gamma})_0$ which does not contain player P_i. Note that this linear combination always

exists: $B \cup \{P_i\} \in \Gamma$ by definition, which means that $\psi(D) = \lambda_i \psi(P_i) + \sum_{P_j \in B} \lambda_j \psi(P_j)$, and $\lambda_i \neq 0$ because $B \notin \Gamma$, so we can write $\psi(P_i)$ as a linear combination of the vectors in $\{\psi(D), \{\psi(P_j)\}_{P_j \in B}\}$.

Here we explain a method to obtain such a public value Δ. This method is in general inefficient. However, there can be access structures for which an appropriate Δ can be found in a more efficient way (for example, in the threshold case studied by Shoup, this factor is simply $\Delta = \ell!$). The general method is the following.

For each minimal authorized subset $A \in \Gamma_0$, we will associate to it a factor M_A such that $\tilde{c}_i^A = M_A c_i^A$ is an integer, for all $P_i \in A$. This factor M_A is the determinant of some non-zero minor, with maximal order, of the matrix G_A whose columns are the vectors $\{\psi(P_i)\}_{P_i \in A}$. We define $Min_A = \{$non-zero minors of matrix G_A with maximal order$\}$. Then we define the value M_A in the following way:

$$M_A = \operatorname*{lcm}_{g_A \in Min_A} \{|\det\ g_A|\}$$

It is clear that this factor M_A cancels all the denominators in all the possible solutions $\{c_i^A\}_{P_i \in A}$. But we are looking for a factor that cancels all the denominators, for all the minimal authorized subsets $A \in \Gamma_0$. Hence we define

$$\Delta_1 = \operatorname*{lcm}_{A \in \Gamma_0} \{M_A\}$$

With respect to the second condition that the value Δ must satisfy, we have to consider systems of equations with the following form: the columns of the matrix of this system are the vectors $\psi(D)$ and $\{\psi(P_j)\}_{P_j \in B}$, where $B \in (\overline{\Gamma})_0$ is a maximal non-authorized subset. We note this matrix $G_{D,B}$. As before, we define $Min_{D,B} = \{$non-zero minors of matrix $G_{D,B}$ with maximal order$\}$, and then the values

$$M_{D,B} = \operatorname*{lcm}_{g_{D,B} \in Min_B} \{|\det\ g_{D,B}|\} \text{ and } \Delta_2 = \operatorname*{lcm}_{B \in (\overline{\Gamma})_0} \{M_{D,B}\}.$$

Finally, the factor Δ is $\Delta = \mathrm{lcm}\{\Delta_1, \Delta_2\}$.

We provide a simple example to see how this method works. Consider a set of five participants, $\mathcal{P} = \{1, 2, 3, 4, 5\}$. Any subset formed by three participants will be able to sign a message. Furthermore, the participants 1 and 2 have more power than the rest, and they can jointly compute valid signatures if they agree. Therefore, the corresponding access structure Γ is defined by its basis

$$\Gamma_0 = \{\ \{1,2\}, \{1,3,4\}, \{1,3,5\}, \{1,4,5\}, \{2,3,4\}, \{2,3,5\}, \{2,4,5\}, \{3,4,5\}\ \}$$

Let us define the adversary structure as the whole family of non-authorized subsets. Therefore, it is determined by the basis

$$\mathcal{A}_0 = (\overline{\Gamma})_0 = \{\ \{1,3\},\{1,4\},\{1,5\},\{2,3\},\{2,4\},\{2,5\},\{3,4\},\{3,5\},\{4,5\}\ \}$$

These structures satisfy the necessary conditions $\mathcal{A}^c \subset \Gamma \subset \overline{\mathcal{A}}$. The access structure Γ can be realized over the rationals by a vector space secret sharing scheme defined by the following vectors: $\psi(D) = (1,1,0)$, $\psi(1) = (1,0,0)$, $\psi(2) = (0,1,0)$, $\psi(3) = (0,0,1)$, $\psi(4) = (1,2,1)$, $\psi(5) = (2,1,1)$. The vectors of the five participants are pairwise linearly independent, so the function ψ is $\mathcal{A}$-independent.

For the computation of the value Δ, we must first consider authorized subsets in Γ_0. For example, if we consider the subset $\{1,2\}$, with related vectors $(1,0,0)$ and $(0,1,0)$, then there is only one non-zero minor with maximal order, which is

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1\ .$$

If we consider the authorized subset $\{3,4,5\}$, then there is only one non-zero minor with maximal order, which is the whole matrix formed by the three corresponding vectors:

$$\begin{vmatrix} 0 & 0 & 1 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{vmatrix} = -3\ .$$

Once all the subsets in Γ_0 have been considered, one finds the value $\Delta_1 = 6$.

Then we must consider maximal non-authorized subsets (those in $\mathcal{A}_0$) and study the matrix formed by the related vectors along with the vector of the dealer. For example, for the subset $\{4,5\}$, the resulting matrix is

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix}$$

and the only non-zero minor with maximal order is the whole matrix, which has determinant equal to 2. Repeating this process with all the subsets in $\mathcal{A}_0$, one finds the value $\Delta_2 = 2$. Therefore, in this case we would obtain $\Delta = 6$.

2.2.2 The Proposal

Let Λ and Γ be adversary and access structures defined on a set of participants $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, satisfying $\Lambda^c \subset \Gamma \subset \overline{\Lambda}$. Suppose, for simplicity,

that we have an appropriate function $\psi : \mathcal{P} \cup \{D\} \longrightarrow \mathbb{Z}^r$ that realizes the access structure Γ over the rationals, and that is Λ-independent.

If these conditions hold, we can construct a RSA distributed signature scheme secure against the action of an adversary who corrupts players in any subset of Λ, and in which participation of players of some subset of Γ is required in order to compute a valid RSA signature. The construction can also be applied in the more general case where Γ is realized by a linear secret sharing scheme; that is, when ψ can assign more than one vector to each participant.

The *RSA-Dist-Key-Gen* protocol. We assume that a trusted dealer executes this protocol. The structures Γ and Λ, the function ψ and the value Δ (that depends only on ψ) are made public. The dealer receives as input a security parameter k, and chooses two k-bit integers $p = 2p' + 1$ and $q = 2q' + 1$ in such a way that:

(i) p, q, p' and q' are primes;

(ii) p' and q' are larger than Δ and larger than the absolute values of all the components of $\psi(D)$.

We denote $\eta = p'q'$. The value $n = pq$ is made public. However, note that η, and so $\phi(n) = 4\eta$, remain secret to the participants. The dealer chooses and makes public a hash function $H : \{0,1\}^* \to \mathbb{Z}_n^*$. He chooses the public exponent e as a positive odd integer such that $\gcd(e, \eta\Delta) = 1$. He also computes $d = e^{-1} \bmod \eta$. Then he performs the following protocol to compute and distribute shares of d among the players:

1. The dealer chooses a random vector $\mathbf{w} \in (\mathbb{Z}_\eta)^r$, such that $\mathbf{w} \cdot \psi(D) = d \bmod \eta$. (This vector exists because of the requirement (ii) above.)

2. He sends to each player P_i his secret share $s_i = \mathbf{w} \cdot \psi(P_i) \bmod \eta$, for $1 \leq i \leq \ell$.

3. The dealer also chooses a random $v \in Q_n$, where Q_n is the subgroup of squares in $\mathbb{Z}_n^*$, and computes $v_i = v^{s_i} \bmod n$, for $1 \leq i \leq \ell$. He makes public the verification keys v and $\{v_i\}_{1 \leq i \leq \ell}$.

We note that v generates Q_n with overwhelming probability, and that the order of Q_n is exactly η.

The *RSA-Dist-Sig* protocol. If the players want to sign a message m, they first compute $x = H(m) \in \mathbb{Z}_n^*$. Then they execute the following protocol:

1. Each player P_i, for $1 \leq i \leq \ell$, computes and broadcasts his partial signature $x_i = x^{4\Delta s_i} \bmod n$.

2. He also makes public a "proof of correctness" of this share, which is a zero-knowledge and non-interactive proof of knowledge that the discrete logarithm of x_i^2 to the base $\tilde{x} = x^{8\Delta}$ is the same as the discrete logarithm of v_i to the base v. To compute such a proof, player P_i proceeds as follows, if $H' : \{0,1\}^* \to \{0,1\}^{L_1}$ is a hash function (taking for example $L_1 = 128$), and $L(n)$ is the bit-length of n:

   2.1. He chooses at random $r \in \{0, 1, \ldots, 2^{L(n)+2L_1} - 1\}$.

   2.2. He computes the values $v' = v^r$, $x' = \tilde{x}^r$, $c = H'(v, \tilde{x}, v_i, x_i, v', x')$ and $z = s_i c + r$.

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

   This non-interactive protocol was also proposed by Shoup in [91], by following some well-known techniques which combine some previous works [23, 45].

3. Each player P_j, for $1 \leq j \leq \ell$, verifies the proof of correctness (z, c) of player P_i, by checking if

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

   If the proof is not correct, P_j complains against P_i.

4. Players who get complaints from a subset which is not in Λ are rejected.

5. Once the dishonest players have been rejected, and because of the requirement $\mathcal{A}^c \subset \Gamma$, there is at least one authorized subset $A \in \Gamma$ in which all the partial signatures are valid. We can consider, for simplicity, that A is a minimal authorized subset, that is, $A \in \Gamma_0$.

6. Each player in A can obtain rational numbers $\{c_i^A\}_{P_i \in A}$ such that $\psi(D) = \sum_{P_i \in A} c_i^A \psi(P_i)$; because of the definition of Δ, we have that the values $\tilde{c}_i^A = \Delta c_i^A$ are integers, and so the users can compute $\sigma = \prod_{P_i \in A} x_i^{\tilde{c}_i^A} = x^{4\Delta^2 d} \bmod n$.

7. Since $\gcd(e, \eta\Delta) = 1$, we have that $\gcd(e, 4\Delta^2) = 1$ because e is odd. Then each player in A can obtain integers a and b such that $4\Delta^2 a + eb = 1$, using the Euclidian algorithm.

8. The signature for the message x is $y = \sigma^a x^b$.

The *RSA-Ver* protocol. The recipient of a signature y on a message m under public key (n, e) verifies its correctness by checking if $y^e = H(m) \bmod n$. If the signature has been generated by using the *RSA-Dist-Sig* protocol, then it is correct, because

$$y^e = \sigma^{ea} x^{be} = x^{4\Delta^2 a + eb} = x = H(m) \mod n.$$

2.2.3 Security Analysis

First we explain why we require the function ψ to be Λ-independent. Suppose that there exists some subset $B \in \Lambda$ of corruptible players such that the vectors in the set $\{\psi(P_j)\}_{P_j \in B}$ are linearly dependent over the rationals. If we use the scheme explained in the previous section, then each player P_j of B will receive the share $s_j = \mathbf{w} \cdot \psi(P_j) \bmod \eta$. If there exists a non-trivial linear combination of the vectors in $\{\psi(P_j)\}_{P_j \in B}$ that is equal to zero, over the rationals, then the same combination applied to the shares $\{s_j\}_{P_j \in B}$ will be equal to zero, in $\mathbb{Z}_\eta$. That is, players in B could obtain a multiple of η, and so break the system.

Now we prove that the way in which the dealer distributes the shares s_i among the players, in the execution of *RSA-Dist-Key-Gen*, is perfectly secure, provided the primes p' and q' are larger than the value Δ.

In effect, let $B = \{P_{j_1}, \ldots, P_{j_s}\} \in \Lambda$ be a corruptible subset of players, and let $\{s_i\}_{1 \le i \le \ell}$ be a sharing of some secret $d \in \mathbb{Z}_\eta$, obtained with a random vector $\mathbf{w} \in (\mathbb{Z}_\eta)^r$. Because of the properties satisfied by the value Δ (in particular, by Δ_2), we know that the system of equations

$$\begin{pmatrix} \cdots & \psi(D) & \cdots \\ \cdots & \psi(P_{j_1}) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \psi(P_{j_s}) & \cdots \end{pmatrix} \begin{pmatrix} \kappa_1 \\ \vdots \\ \vdots \\ \kappa_r \end{pmatrix} = \begin{pmatrix} \Delta \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

has at least one integer solution $\kappa = (\kappa_1, \ldots, \kappa_r) \in \mathbb{Z}^r$. Since p' and q' are larger than Δ, we have $\gcd(\Delta, \eta) = 1$, and so there exist integers α and β such that $\alpha\Delta + \beta\eta = 1$.

For every possible value of the secret $d' \in \mathbb{Z}_\eta$ and for every solution $\kappa \in \mathbb{Z}^r$ of the system above, consider the vector $\mathbf{w}' = \mathbf{w} + (d' - d)\alpha\kappa$ with its entries reduced modulo η. We have then that the vector $\mathbf{w}' \in (\mathbb{Z}_\eta)^r$ satisfies the two following equations:

(i) $\mathbf{w}' \cdot \psi(D) = (\ \mathbf{w} + (d' - d)\alpha\kappa\ ) \cdot \psi(D) = d + (d' - d)\alpha\Delta = d' \mod \eta,$

(ii) $\mathbf{w}' \cdot \psi(P_j) = (\ \mathbf{w} + (d' - d)\alpha\kappa\ ) \cdot \psi(P_j) = s_j \mod \eta$, for all $P_j \in B$.

So each possible secret $d \in \mathbb{Z}_\eta$ is equally likely from the point of view of an adversary who knows the shares $\{s_j\}_{P_j \in B}$. Therefore, an adversary who corrupts players in B does not obtain any information about the secret from the shares that it knows.

Robustness

To show that the RSA distributed signature scheme is robust, we must prove that corrupted players who do not follow the protocol properly are detected with overwhelming probability. This is due to the soundness property satisfied by the non-interactive protocol which is used for proving the correctness of the signature shares (step 2 of the *RSA-Dist-Sig* protocol). The proof of this property, which is valid in the random oracle model for the hash function H', has been taken from [91].

Proposition 2.1. *Except with negligible probability, it is impossible to construct a valid proof of correctness for an incorrect partial signature, in the random oracle model for the hash function H'.*

Proof. Consider a given pair (x, x_i), where $x = H(m)$ and x_i is the partial signature broadcast by some player P_i. Let us suppose that the corresponding proof of correctness broadcast by P_i is (z, c).

If the proof is valid, then we have $c = H'(v, \tilde{x}, v_i, x_i, v', x')$, where

$$\tilde{x} = x^{8\Delta},\ v' = v^z v_i^{-c},\ x' = \tilde{x}^z x_i^{-2c}. \tag{2.1}$$

Since we are assuming that the hash function H' behaves as a random function, the value c is uniform and independent of the inputs of the hash function.

It is easy to see that $\tilde{x}, v_i, x_i^2, v', x'$ all belong to Q_n, the subgroup of squares in $\mathbb{Z}_n^*$, which is assumed to be generated by v, of order η. Therefore, we have that

$$\tilde{x} = v^\alpha,\ v_i = v^{s_i},\ x_i^2 = v^\beta,\ v' = v^\gamma,\ x' = v^\delta\ , \tag{2.2}$$

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

From equations 2.1 and 2.2, we conclude that $z - cs_i = \gamma \mod \eta$ and $z\alpha - c\beta = \delta \mod \eta$. This implies:

$$c(\alpha s_i - \beta) = \delta - \gamma\alpha \mod \eta\ . \tag{2.3}$$

On the other hand, the partial signature x_i is correct if and only if $\tilde{x}^{s_i} = x_i^2$ This condition is equivalent to $v^{\alpha s_i} = v^\beta$, which happens if and only if

$$\beta = \alpha s_i \mod \eta\ . \tag{2.4}$$

Summing up, if the proof of correctness is valid but the partial signature is incorrect, then equation 2.4 must fail to hold modulo p' or modulo q'. In this case, equation 2.3 uniquely determines the value c modulo one of these primes.

Remember, however, that the distribution of c was uniform and independent of the rest of values. Therefore, the probability that the value c determined by equation 2.3 coincides with the output of the hash value H' is negligible. And this is exactly the probability that an incorrect partial signature results in a valid proof of correctness.

□

Once this result is proven, then it is straightforward to see that the scheme is robust. Corrupted players who broadcast incorrect partial signatures are detected and expelled from the protocol. Because of the required combinatorial conditions on the structures Γ and Λ, the remaining honest players always form an authorized subset, so they can finish the protocol and compute a valid RSA signature.

Unforgeability

We reduce the security of the proposed RSA distributed scheme to the security of standard FDH-RSA signature scheme, introduced in Section 1.2.1 and analyzed in Section 1.3.4. The proof is again in the random oracle model, for the hash functions H and H'. Therefore, an adversary will be allowed to make Q queries to the random oracle which models the hash function H, and Q' queries to the random oracle which models H'.

Theorem 2.1. *If there exists a $(\mathcal{P}, \Gamma, \Lambda, T, \varepsilon, Q, Q', Q_s)$-forger against our RSA distributed signature scheme, then there exists a $(T', \varepsilon', Q, Q_s)$-forger against standard FDH-RSA signature scheme, with $T' \leq T + \ell T_{exp} Q_s + Q'$ and $\varepsilon' \geq \varepsilon - \delta$, where δ is negligible in the security parameter k of the scheme (the length of the employed RSA primes).*

Proof. Let $\mathcal{F}_{Dist}$ denote the $(\mathcal{P}, \Gamma, \Lambda, T, \varepsilon, Q, Q', Q_s)$-forger against the proposed RSA distributed signature scheme. This adversary plays game $\mathcal{G}_1$, described in Section 2.1.1. We will construct a $(T', \varepsilon', Q, Q_s)$-forger against the standard FDH-RSA signature scheme, that we will denote $\mathcal{F}_{RSA}$, and which will use the forger $\mathcal{F}_{Dist}$ as a subroutine.

By definition, the forger $\mathcal{F}_{RSA}$ receives a RSA public key (n, e). It can query Q answers to the random oracle model H (the hash function of FDH-RSA signature scheme) and can ask for Q_s valid RSA standard signatures. Its goal is to obtain a new valid RSA standard signature.

Once $\mathcal{F}_{RSA}$ receives the input (n, e), it executes the forger $\mathcal{F}_{Dist}$, simulating the environment of $\mathcal{F}_{Dist}$ in game $\mathcal{G}_1$. This idea is captured in Figure 2.1 below. The details are as follows:

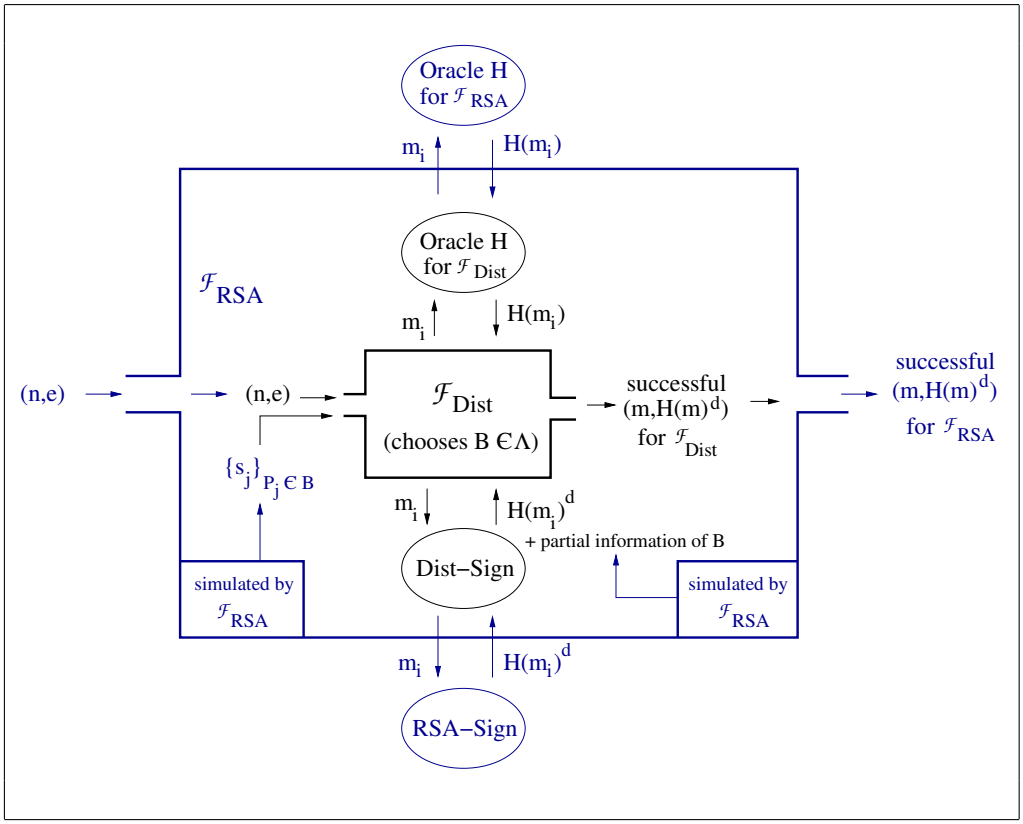

Figure 2.1: $\mathcal{F}_{Dist}$ simulates the environment of $\mathcal{F}_{RSA}$

1. Since H and H' are assumed to behave as random functions, the adversary $\mathcal{F}_{Dist}$ can make Q queries to the random oracle for H and Q' queries to the random oracle for H'. When it asks to the random oracle H, the forger $\mathcal{F}_{RSA}$ can query its oracle for the same hash function H, and sends to $\mathcal{F}_{Dist}$ the obtained output. To answer the queries to the random oracle H', forger $\mathcal{F}_{RSA}$ manages the table TAB' as usual: if an input is already stored in the table, it answers the corresponding output; if not, he chooses an output uniformly at random, gives it as answer and stores the new relation in TAB'.

2. The forger $\mathcal{F}_{Dist}$ is given a set of players $\mathcal{P}$, a monotone increasing access structure $\Gamma \subset 2^{\mathcal{P}}$ and an adversary structure $\Lambda \subset 2^{\mathcal{P}}$. We can

assume that $\mathcal{F}_{Dist}$ also receives here the function $\psi : \mathcal{P} \cup \{D\} \longrightarrow \mathbb{Z}^r$ realizing Γ over the rationals, which must be Λ-independent, and the value Δ.

3. The adversary chooses a subset of players $B \in \Lambda$ to corrupt.

4. The *RSA-Dist-Key-Gen* protocol is executed. The adversary obtains all the information that is made public in the execution of this protocol, as well as private information corresponding to the corrupted players $P_j \in B$.

   Let us see how $\mathcal{F}_{RSA}$ can provide $\mathcal{F}_{Dist}$ with this information. The resulting RSA public key will be (n, e). Because of the necessary combinatorial condition $\mathcal{A} \subset \overline{\Gamma}$, there exists a maximal non-authorized subset $B' \in (\overline{\Gamma})_0$ such that $B \subset B'$. The real share of a player P_j in B would be $s_j = \mathbf{w} \cdot \psi(P_j) \bmod \eta$, where $\mathbf{w}$ is a random vector in $(\mathbb{Z}_\eta)^r$. To simulate these shares, $\mathcal{F}_{RSA}$ chooses a random vector $\tilde{\mathbf{w}}$ in $(\mathbb{Z}_{\tilde{\eta}})^r$, where $\tilde{\eta} = \lfloor n/4 \rfloor - 1$. Then, it computes $\tilde{s}_j = \tilde{\mathbf{w}} \cdot \psi(P_j) \bmod \tilde{\eta}$, for all $P_j \in B'$.

   $\mathcal{F}_{RSA}$ must also simulate consistent values for the verification keys. It chooses at random an element $\rho \in Q_n$ and computes $\tilde{v} = \rho^{4e\Delta} \bmod n$. For players $P_j \in B'$, it computes $\tilde{v}_j = \tilde{v}^{\tilde{s}_j}$.

   It remains to compute the simulated values $\tilde{v}_i$, for the rest of players $P_i \in \mathcal{P} - B'$. Since $B' \in (\overline{\Gamma})_0$, we have that $B' \cup \{P_i\} \in \Gamma$, for all $P_i \in \mathcal{P} - B'$. This condition is equivalent, over the rationals, to $\psi(P_i) \in \langle \psi(D), \{\psi(P_j)\}_{P_j \in B'} \rangle$. That is, there exist rational numbers $f_0^{B',i}, \{f_j^{B',i}\}_{P_j \in B'}$ such that

$$\psi(P_i) = f_0^{B',i} \psi(D) \ + \ \sum_{P_j \in B'} f_j^{B',i} \psi(P_j)$$

   Due to the second condition that the value Δ must satisfy, the values $\Delta f_0^{B',i}$ and $\{\Delta f_j^{B',i}\}_{P_j \in B'}$ are all integers.

   Note that, ideally, the shares must correspond to the secret $d = e^{-1} \bmod \eta$, where $\phi(n) = 4\eta$. Therefore, the equality

$$\tilde{s}_i = f_0^{B',i} d \ + \ \sum_{P_j \in B'} f_j^{B',i} \tilde{s}_j \ \bmod \eta \tag{2.5}$$

   should be satisfied. We cannot compute $\tilde{s}_i$, however we are interested in computing only $\tilde{v}^{\tilde{s}_i}$. Multiplying equality 2.5 with e, we obtain

$$e\tilde{s}_i = f_0^{B',i} \ + \ e \sum_{P_j \in B'} f_j^{B',i} \tilde{s}_j \ \bmod \eta. \tag{2.6}$$

Therefore, $\mathcal{F}_{RSA}$ can produce the simulated verification keys $\tilde{v}_i$, for all players $P_i \in \mathcal{P} - B'$, as follows:

$$\rho^{4\Delta f_0^{B',i} + 4e \sum_{P_j \in B'} \Delta f_j^{B',i} \tilde{s}_j} = \rho^{4\Delta e \tilde{s}_i} = \tilde{v}^{\tilde{s}_i} = \tilde{v}_i \mod n \ .$$

Forger $\mathcal{F}_{RSA}$ provides forger $\mathcal{F}_{Dist}$ with all this information: n, e, $\{\tilde{s}_j\}_{P_j \in B}$, $\tilde{v}$ and $\{\tilde{v}_i\}_{P_i \in \mathcal{P}}$.

Since the statistical distance between the uniform distribution on $\{0, \ldots, \eta\}$ and the uniform distribution on $\{0, \ldots, \tilde{\eta}\}$ is $\mathcal{O}(n^{-1/2})$ (which is negligible in the security parameter), we can deduce that the distribution of real values (in particular, $\{s_j\}_{j \in B}$) and the distribution of simulated values (in particular, $\{\tilde{s}_j\}_{j \in B}$) are computationally indistinguishable, as desired.

5. The adversary can adaptively choose Q_s messages m^* to be signed. For these messages, the *RSA-Dist-Sig* protocol is executed. The adversary sees all the information that is made public in the required executions of this protocol. Let us see how $\mathcal{F}_{RSA}$ can simulate this information.

   First, $\mathcal{F}_{RSA}$ is allowed to ask the random oracle model which models H for the value $H(m^*)$, obtaining the output x^*. Then, it can query its signing oracle to obtain a valid standard FDH-RSA signature for the message m^*; it obtains as answer the value y^* such that $(y^*)^e = x^* = H(m^*)$.

   The partial signatures corresponding to the players P_j in the set B' would be $x_j^* = (x^*)^{4\Delta \tilde{s}_j}$. If the forger $\mathcal{F}_{Dist}$, which manages corrupted players in B, forces some of them to broadcast an incorrect partial share, then $\mathcal{F}_{RSA}$ broadcasts complaints against this corrupted player from all the non-corrupted ones.

   To simulate consistent partial signatures x_i^* for the rest of players $P_i \in \mathcal{P} - B'$, the forger $\mathcal{F}_{RSA}$ can use the same strategy as before, when it simulated the values $\tilde{v}_i$. In effect, by using equality 2.6, it can compute

$$(y^*)^{4(\Delta f_0^{B',i} + e \sum_{j \in B'} \Delta f_j^{B',i} \tilde{s}_j)} = (y^*)^{4e\Delta \tilde{s}_i} = (x^*)^{4\Delta \tilde{s}_i} = x_i^* \mod n \ .$$

   Forger $\mathcal{F}_{RSA}$ must also simulate the proofs of correctness of the partial signatures corresponding to non-corrupted players (remember that corrupted players are managed by $\mathcal{F}_{Dist}$, so it is not necessary to provide it with this information). Forger $\mathcal{F}_{RSA}$ will construct and maintain a

table TAB' to store the definition of the random oracle which models the hash function H'. It will manage this table in the usual way: when it needs an output of the hash function, it looks for the input in the table. If it is already there, it returns the stored output; otherwise, it chooses at random a value different from the rest of stored values, returns it as the input, and stores the new relation in the table.

For non-corrupted players P_j in $B' - B$, forger $\mathcal{F}_{RSA}$ can compute valid proofs of correctness for the partial signatures x_j^*, by using knowledge of the shares $\tilde{s}_j$, applying the non-interactive protocol explained in step 2 of *RSA-Dist-Sig* (see Section 2.2.2).

For the rest of players $P_i \in \mathcal{P} - B'$, forger $\mathcal{F}_{RSA}$ simulates a consistent proof of correctness for the pair (x^*, x_i^*) as follows: it chooses at random $c \in \{0, 1, \ldots, 2^{L_1} - 1\}$ and $z \in \{0, 1, \ldots, 2^{L(n)+2L_1} - 1\}$. With overwhelming probability, the input $\left(\tilde{v}, \tilde{x^*}, \tilde{v}_i, x_i^*, \tilde{v}^z \tilde{v}_i^{-c}, (\tilde{x^*})^z (x_i^*)^{-2c}\right)$, where $\tilde{x^*} = (x^*)^{8\Delta}$, has not been still asked to the random oracle H'; therefore, $\mathcal{F}_{RSA}$ can define the output of the random oracle for this input to be exactly c, and store this new relation in the table TAB'. This makes the proof (c, z) consistent.

In this way, $\mathcal{F}_{RSA}$ can provide the forger $\mathcal{F}_{Dist}$ with all the information it would see in a real execution of the *RSA-Dist-Sig* protocol for message m^*.

In time at most T and with probability greater than ε, forger $\mathcal{F}_{Dist}$ obtains a valid RSA signature (m, y), satisfying $y^e = H(m) \bmod n$, such that $(m, y) \neq (m^*, y^*)$ for all the pairs (m^*, y^*) obtained during game $\mathcal{G}_1$.

Therefore, forger $\mathcal{F}_{RSA}$ has performed a valid chosen-message forgery against standard FDH-RSA signature scheme, because it has obtained from its signing oracle the same pairs (m^*, y^*) as the forger $\mathcal{F}_{Dist}$, and so the final valid RSA signature (m, y) is original. The success probability ε' of $\mathcal{F}_{RSA}$ is exactly the same as the one of $\mathcal{F}_{Dist}$, except the negligible probability that there is some collision in the simulation of the outputs of the random oracle H'. We can thus write $\varepsilon' \geq \varepsilon - \delta$, where δ is a negligible function in the security parameter k of the scheme.

With respect to the execution time T' of $\mathcal{F}_{RSA}$, it consists of the execution time of $\mathcal{F}_{Dist}$ plus the time to answer the queries to the signing oracle plus the management of the table TAB' modeling the random oracle H'. We can thus write $T' \leq T + \ell T_{exp} Q_s + Q'$.

Note that forger $\mathcal{F}_{RSA}$ makes exactly the same queries Q_s and Q to the signing oracle and the random oracle H than forger $\mathcal{F}_{Dist}$ does.

□

Putting together both Theorem 1.1 and Theorem 2.1, we immediately obtain the following result, which relates the unforgeability of our RSA distributed signature scheme with the RSA Assumption (see Definition 1.7).

Theorem 2.2. *If there exists a* $(\mathcal{P}, \Gamma, \Lambda, T, \varepsilon, Q, Q', Q_s)$*-forger against the proposed RSA distributed signature scheme, with security parameter* k*, then the RSA problem can be solved in time* $T' \leq T + 2\ell T_{exp} Q_s + T_{exp} Q + Q'$ *and with probability* $\varepsilon' \geq \frac{\varepsilon - \delta}{\mathbf{e} \cdot (Q_s + 1)}$ *(in the random oracle model), where* δ *is negligible in* k *and* $\mathbf{e}$ *is the Euler's value* $\mathbf{e} = 2,718281828459...$

2.3 Other Extensions

In the last section of this chapter, we discuss some topics related to the RSA distributed signature scheme that we have proposed for general access structures.

2.3.1 Eliminating the Trusted Dealer

Our proposed RSA distributed signature scheme for general structures, as well as the original threshold scheme by Shoup [91], require the use of safe primes. That is, the public key must be $n = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$, such that p, q, p' and q' are all prime numbers. For the generation of these parameters, a trusted third party (dealer) is required, who computes the prime numbers and distributes the shares of the secret key among the players.

In [31, 47], some modifications of the threshold scheme of Shoup are proposed, in such a way that the prime numbers p and q must not be safe primes. In this way, one can use some known protocols [12, 49, 21] which enable the own players to jointly generate the public parameters of the scheme and their shares of the secret key, without the help of any trusted dealer. The resulting RSA threshold signature schemes are therefore fully distributed.

Later, a different protocol for the joint generation of RSA public keys has been proposed in [3], which produces a public key $n = pq$ and shares of the secret key, and such that p and q are safe primes. Therefore, this protocol can be directly combined with Shoup's scheme to obtain a fully distributed RSA threshold signature scheme, as well.

Using the techniques that we have employed in this chapter to extend the scheme of Shoup, the schemes in [31, 47] can also be extended to work with general access structures. But we are still unable to design a fully distribute RSA signature scheme for general structures, because all the protocols

for the joint generation of RSA keys [12, 49, 21, 3] require the computation of shares of the product of two shared secrets. This is known as the *multiplication problem*, which is known to have an efficient solution, in the presence of active adversaries, only for the threshold case and a few more particular access structures. Finding a protocol for the joint generation of RSA keys which works efficiently for the case of general access structures remains therefore as an interesting open problem.

2.3.2 Distributed Paillier's Cryptosystem

In [82], Paillier proposed a new public key encryption scheme which enjoys some interesting properties. One of them is that the scheme is *homomorphic*: roughly speaking, if c_1 is an encryption of a message m_1 and c_2 is an encryption of a message m_2, then $c = c_1 \cdot c_2$ is an encryption of the message $m = m_1 + m_2$. This property makes Paillier's cryptosystem specially useful for being applied in e-voting schemes, digital auctions, etc.

In such situations, it is desirable that the power to decrypt a ciphertext is distributed, and not held by a single person or machine. In this sense, a threshold version of the encryption scheme of Paillier was proposed in [46]: the secret key of the cryptosystem is distributed in shares among a set of ℓ players. The messages can be encrypted by anyone by using the matching public key, but in order to decrypt the resulting ciphertexts, the cooperation of at least t of the ℓ players is necessary.

The threshold decryption scheme proposed in [46] basically follows the ideas of the RSA threshold signature scheme by Shoup [91]. Therefore, we could apply to their decryption scheme the same techniques that we have introduced in this chapter. As a result, we would obtain a distributed decryption scheme for more general access structures Γ and $\mathcal{A}$: in order to decrypt a ciphertext, all the members of some authorized subset of players in Γ must collaborate; the scheme is secure if an adversary is restricted to corrupt only the players of some subset in the family $\mathcal{A}$. In this way, the new scheme would fit in with more real situations, for example when the authorities that decrypt the votes in an election or the bids in an auction have different rights, different computational resources or different susceptibilities of being corrupted.

Chapter 3

Some Applications of Distributed Signatures

In the present chapter, we use distributed signature schemes as a primitive to construct some distributed protocols, which have applications on the Internet: metering schemes and distributed key distribution schemes.

The employed distributed signature schemes must be secure (robust and unforgeable under chosen message attacks), and they must also satisfy some extra property: for constructing metering schemes, the distributed signature scheme must be *non-interactive*; for constructing distributed key distribution schemes, the distributed signature scheme must be *deterministic*.

Definition 3.1. *A distributed signature scheme is non-interactive if each player P_i can compute its partial signature $\theta_i(m)$ of a message m without interacting with the rest of players.*

Definition 3.2. *A distributed signature scheme is deterministic if every message m has a unique valid signature $\theta(m)$.*

Note that the notion of determinism can also be applied to standard signature schemes. For example, RSA signature scheme (see Section 1.2.1) is deterministic, whereas Schnorr's signature scheme (see Section 1.2.2) is non-deterministic: every message has $q-1$ valid signatures, one for each value of the element $a \in \mathbb{Z}_q^*$ that the signer chooses at random.

With respect to existing distributed signature schemes in the literature, some of the proposals based on RSA [91, 31, 47], including the scheme that we propose in Section 2.2.2 for the case of general structures, are both non-interactive and deterministic. The threshold signature scheme proposed by Boldyreva in [10] is also non-interactive and deterministic. Therefore, all

these distributed signature schemes could be used to construct both new metering schemes and distributed key distribution schemes, by applying the general constructions that we propose in this chapter.

The threshold version of the Gennaro-Halevi-Rabin signature scheme [51], which is proposed in [21], is deterministic but interactive. On the other hand, distributed versions of some discrete logarithm based signature schemes, like Schnorr's one in [93] or DSS in [53], are neither non-interactive nor deterministic, so they do not provide neither metering schemes nor distributed key distribution schemes.

Remember that the adversary structure Λ contains the subsets of players that an adversary is allowed to corrupt without breaking the distributed signature scheme.

3.1 Constructing Metering Schemes

Metering schemes were introduced in [75] to measure the interactions between servers and clients. Each client visiting a server must give some secret information to that server. When a server has been visited by some specific subset of clients in a period of time, he can compute a valid proof of this fact from the received secret information. These proofs can be taken into account to decide on advertisement fees for web servers on the Internet, for example, since the most usual application of metering schemes is as a counter of the web accesses to servers on the Internet.

We will show how any secure and non-interactive distributed signature scheme can be used to construct a computationally secure metering scheme.

3.1.1 Review of Metering Schemes

In this section we review the protocols that take part in a metering scheme, and the current state of the art in this topic.

In the *initialization phase* of a metering scheme, an *audit agency* distributes some secret information c_i to each client C_i in $\mathcal{C} = \{C_1, \ldots, C_\ell\}$. There are an access structure $\Gamma_{\mathcal{C}} \subset 2^{\mathcal{C}}$ and an adversary structure $\Lambda_{\mathcal{C}} \subset 2^{\mathcal{C}}$, both defined on the set of clients.

In the *regular operation*, when a client C_i visits some server S_j in $\mathcal{S} = \{S_1, \ldots, S_r\}$ during a time frame $\lambda = 1, \ldots, \tau$, he gives some piece of information c_{ij}^{λ} to him.

Once a server S_j has been visited by an authorized subset $A \in \Gamma_{\mathcal{C}}$ of clients, S_j can compute the proof p_j^{λ}. This is the *proof computation phase*.

With this proof, the server can demonstrate to the audit agency that clients of an authorized subset have visited him in time frame λ.

Many proposals of metering schemes consider only the threshold case for $\Gamma_{\mathcal{C}}$; that is, a proof can be computed when a server receives the visit of t clients, where t is the threshold. In this work we consider more general structures, because we think that more general situations can make sense. For example, in order to decide on advertisement fees for a web page dealing with economic products, the visit of a manager of a bank is more important than the visit of a lawyer. Therefore, considering access structures where users play different roles is important.

Some unconditionally secure metering schemes have been proposed [75, 73, 8]. On the other hand, only a few schemes in a computational setting have been proposed. In [75] Naor and Pinkas propose a computationally secure scheme under the computational Diffie-Hellman assumption using bivariable polynomials. Ogata and Kurosawa propose in [79] a computationally secure scheme based on the same assumption, using polynomials in three variables, which repairs some minor security fails in the proposal of [75]. All these proposals consider only threshold access structures for the set of clients.

3.1.2 Security Requirements for Metering Schemes

Roughly speaking, to guarantee the security of a computational metering scheme, we require a server not to be able to compute a valid proof for a time frame if he has not been visited by an authorized subset of clients during this period of time, even if he receives information from the other servers. Our definition of security is more complete than the one in [79], because we allow for a collusion of all the servers, whereas in [79] only a certain number of servers can be corrupted at the same time.

To formalize this idea, we consider an adversary which plays the following game $\mathcal{G}_2$:

1. The adversary is given the public parameters of the system: the set of servers $\mathcal{S} = \{S_1, \ldots, S_r\}$, the set of clients $\mathcal{C} = \{C_1, \ldots, C_\ell\}$, the access and adversary structures $\Gamma_{\mathcal{C}} \subset 2^{\mathcal{C}}$ and $\Lambda_{\mathcal{C}} \subset 2^{\mathcal{C}}$, and the number of time frames, τ.

2. The adversary chooses a subset of clients $B \in \Lambda_{\mathcal{C}}$ to be corrupted.

3. The initialization phase of the metering scheme is executed. The adversary obtains all the public information and the secret information of the corrupted clients $C_b \in B$.

4. The regular operation is executed for all clients $C_i \in \mathcal{C}$, for all servers $S_j \in \mathcal{S}$ and for all the time frames $\lambda \leq t$, where $t \in \{1, \ldots, \tau\}$, except for the case $(j, \lambda) = (j_0, t)$ (that is, some server S_{j_0} is not visited during time frame t).

   The adversary obtains from these executions all the information received from all the servers, in particular the pieces of information c_{ij}^{λ}.

5. The goal of the adversary is to obtain a valid proof $p_{j_0}^t$ for the server S_{j_0} and the time frame t.

Definition 3.3. *Such an adversary $(\mathcal{S}, \mathcal{C}, \Gamma_{\mathcal{C}}, \Lambda_{\mathcal{C}}, T, \varepsilon)$-breaks the metering scheme if its running time is at most T, and the probability that it obtains a valid proof $p_{j_0}^t$ for server S_{j_0} and time frame t, after executing game $\mathcal{G}_2$, is at least ε.*

3.1.3 The New Construction: Σ-Metering Schemes

Now we present a new method to construct a computationally secure metering scheme from any non-interactive distributed signature scheme. The resulting metering scheme, like other computationally secure ones, allows servers to check the correctness of the information received from clients in a regular operation, provided the distributed signature scheme is robust.

We consider a non-interactive distributed signature scheme Σ, defined on the set of clients $\mathcal{C}$ with access structure $\Gamma_{\mathcal{C}}$. We will call such a scheme a $(\mathcal{C}, \Gamma_{\mathcal{C}})$-distributed signature scheme. We will assume Σ is secure (robust and unforgeable) with respect to an adversary structure $\Lambda_{\mathcal{C}}$. We denote as Σ-*metering scheme* the metering scheme constructed as follows.

Initialization phase. In this phase, the audit agency executes the *Dist-Key-Gen* protocol of Σ to generate public outputs pk and vk_i (the public and the verification keys of the signature scheme), and private outputs sk_i. Then the audit agency sends privately the information $c_i = sk_i$ to each client C_i.

Regular operation. The idea behind this protocol is that clients produce partial signatures on message $h(S_j, \lambda)$ when they visit server S_j in a time frame λ. Here h must be a public and collision-resistant hash function whose outputs are valid messages for the signature scheme Σ.

Since Σ is non-interactive, the client C_i does not need the other clients to compute the partial signature $c_{ij}^{\lambda} = \theta_i(h(S_j, \lambda))$ and send it privately to server S_j, together with some proof of correctness. Server S_j checks if the

partial signature is correct using the public verification key (recall that Σ is assumed to be a robust distributed signature scheme); if not, he denies the access to C_i.

Proof computation phase. Assume that server S_j has been visited, during a time frame λ, by all clients of some authorized subset $A \in \Gamma_{\mathcal{C}}$. Server S_j uses the combiner algorithm of the *Dist-Sign* protocol of Σ to produce a valid standard signature $p_j^\lambda = \theta(h(S_j, \lambda))$ from valid partial signatures $\{\theta_i(h(S_j, \lambda))\}_{C_i \in A}$. The audit agency (or anyone) uses the public key pk to verify the validity of the proof p_j^λ using the *Ver* protocol of Σ.

One of the main advantages of the resulting metering scheme with respect to previously known ones, is that the audit agency is necessary only in the initialization phase. After this phase, it can disappear from the system. In particular, due to the fact that the proofs are digital signatures, anyone can verify their correctness; for example, the own companies which must pay web servers for having their advertisements on the server's web pages.

In some previous metering schemes (for example the one in [79]), the audit agency uses a secret sharing scheme to distribute some secret information among the set of servers, as well. This implies that an adversary is allowed to corrupt only some specific subsets of servers without compromising the security of the protocol. In Σ-metering schemes, this does not happen, and so they are secure even if the adversary corrupts all the servers of the system.

If the distributed signature schemes Σ works only for threshold access structures, then the authorized subsets of clients of the resulting Σ-metering scheme are those with at least t clients, where t is the threshold. As we have said before, this may be a limitation in some scenarios, for example if an advertisement company thinks that the visit of some clients to the web page is more interesting than other visits. For this reason it is important to consider general access structures in the set of clients. In our construction, this can be achieved by using a distributed signature scheme which works for general access structures, for example the RSA-based scheme that we propose in Chapter 2.

3.1.4 Security of Σ-Metering Schemes

The following result relates the security of Σ-metering schemes with the unforgeability of the employed distributed signature scheme Σ. Therefore, if we use a secure distributed signature scheme, we obtain a computationally secure metering scheme.

Theorem 3.1. *If there exists an adversary which $(\mathcal{S}, \mathcal{C}, \Gamma_{\mathcal{C}}, \Lambda_{\mathcal{C}}, T, \varepsilon)$-breaks a Σ-metering scheme with ℓ clients, r servers and τ time frames, then there exits a $(\mathcal{C}, \Gamma_{\mathcal{C}}, \Lambda_{\mathcal{C}}, T', \varepsilon', Q, Q_s)$-forger against the distributed signature scheme Σ, with $Q = 0$, $Q_s = r\tau$, $T' \leq T + r\tau$ and $\varepsilon' \geq \varepsilon - \delta$, where δ is a negligible function of the security parameter of Σ.*

Proof. Let $\mathcal{F}_{Met}$ denote the adversary which $(\mathcal{S}, \mathcal{C}, \Gamma_{\mathcal{C}}, \Lambda_{\mathcal{C}}, T, \varepsilon)$-breaks the Σ-metering scheme. This adversary plays game $\mathcal{G}_2$, described in Section 3.1.2. We will construct a $(\mathcal{C}, \Gamma_{\mathcal{C}}, \Lambda_{\mathcal{C}}, T', \varepsilon', 0, r\tau)$-forger against the distributed signature scheme Σ, that we will denote $\mathcal{F}_{\Sigma}$, and which will use the forger $\mathcal{F}_{Met}$ as a subroutine.

The goal of the forger $\mathcal{F}_{\Sigma}$ is to play game $\mathcal{G}_1$ (see Section 2.1.1) and obtain a new valid signature. In the first steep, $\mathcal{F}_{\Sigma}$ receives the set of ℓ players $\mathcal{P}$, the access structure Γ and the adversary structure Λ.

At this moment, $\mathcal{F}_{\Sigma}$ executes the forger $\mathcal{F}_{Met}$, simulating the environment of $\mathcal{F}_{Met}$ in game $\mathcal{G}_2$, by using the information it obtains from its game $\mathcal{G}_1$.

First of all, $\mathcal{F}_{\Sigma}$ gives to $\mathcal{F}_{Met}$ the public parameters: it chooses a set of r servers $\mathcal{S} = \{S_1, \ldots, S_r\}$, imposes the set of ℓ clients to be $\mathcal{C} = \mathcal{P}$, and the access and adversary structures to be $\Gamma_{\mathcal{C}} = \Gamma$ and $\Lambda_{\mathcal{C}} = \Lambda$. It also sends to $\mathcal{F}_{Met}$ the total number of time frames, τ.

By definition, in step 2 of game $\mathcal{G}_2$ the adversary $\mathcal{F}_{Met}$ chooses a subset of clients $B \in \Lambda_{\mathcal{C}}$ to be corrupted. For its game $\mathcal{G}_1$, forger $\mathcal{F}_{\Sigma}$ chooses the same set B of corrupted players.

In step 3 of game $\mathcal{G}_2$, the initialization phase of the metering scheme is executed, and the adversary obtains all the public information and the secret information of the corrupted clients $C_b \in B$. To provide $\mathcal{F}_{Met}$ with this information, forger $\mathcal{F}_{\Sigma}$ goes into step 4 of its game $\mathcal{G}_1$: *Dist-Key-Gen* protocol of Σ is executed, and the forger obtains all the public information and the shares of the secret key which correspond to the corrupted players, in B. Forger $\mathcal{F}_{\Sigma}$ gives to the adversary $\mathcal{F}_{Met}$ all this information.

In step 4 of game $\mathcal{G}_2$, the regular operation is executed for all clients $C_i \in \mathcal{C}$, for all servers $S_j \in \mathcal{S}$ and for all the time frames $\lambda \leq t$, where $t \in \{1, \ldots, \tau\}$, except for the case $(j, \lambda) = (j_0, t)$. To provide $\mathcal{F}_{Met}$ with the information that it obtains from these executions, the forger $\mathcal{F}_{\Sigma}$ executes step 5 of its game $\mathcal{G}_1$: it chooses the messages $h(S_j, \lambda)$, for $j = 1, \ldots, r$, $\lambda = 1, \ldots, t$, and $(j, \lambda) \neq (j_0, t)$, to be signed by its signing oracle. For these messages, the *Dist-Sig* protocol of Σ is executed. The adversary $\mathcal{F}_{\Sigma}$ sees all the information that is made public in these executions of *Dist-Sig*, as well as private information corresponding to the corrupted players. In particular, it obtains all the partial signatures which are the pieces of

information $c_{ij}^{\lambda} = \theta_i(h(S_j, \lambda))$, and the final signatures, which are the proofs $p_j^{\lambda} = \theta(h(S_j, \lambda))$, for all clients $C_i \in \mathcal{C}$, for all servers $S_j \in \mathcal{S}$ and for all the time frames $\lambda \leq t$, where $t \in \{1, \ldots, \tau\}$, except for the case $(j, \lambda) = (j_0, t)$. Forger $\mathcal{F}_\Sigma$ gives to the adversary $\mathcal{F}_{Met}$ all this information.

Note that the number of times that $\mathcal{F}_\Sigma$ queries the signing oracle is at most $Q_s = r\tau$.

By hypothesis, the adversary $\mathcal{F}_{Met}$ obtains with probability ε a valid proof for the server S_{j_0} and the time frame t. That is, it obtains a valid signature $p_{j_0}^t = \theta(h(S_{j_0}, t))$. Since the hash function h is assumed to be collision-resistant, the probability δ that $h(S_{j_0}, t) = h(S_j, \lambda)$, for some pair $(j, \lambda) \neq (j_0, t)$ is negligible in the security parameter of Σ, if the outputs of h belong to a set whose size is exponential in this security parameter.

Therefore, with probability $\varepsilon' \geq \varepsilon - \delta$, the forger $\mathcal{F}_\Sigma$ has obtained a valid signature $p_{j_0}^t$ for some message $h(S_{j_0}, t)$ which is different from the pairs (message,signature) that $\mathcal{F}_\Sigma$ has asked to its signing oracle in the execution of game $\mathcal{G}_1$. That is, it has obtained a successful forgery against the distributed signature scheme Σ.

The forger $\mathcal{F}_\Sigma$ has made no queries to any random oracle (therefore, $Q = 0$), and its running time T' consists of the running time of $\mathcal{F}_{Met}$ plus the time of making the $r\tau$ queries to its signing oracle. We can thus write $T' \leq T + r\tau$.

□

Obviously, if we use a distributed signature scheme Σ whose security is proved in the random oracle model, then the security of the resulting Σ-metering scheme is proved in the same model. But, on the positive side, if the scheme Σ is secure in the standard model, then Σ-metering scheme also achieves this property, because the construction does not add any ideal assumption on the behaviour of the hash functions. Unfortunately, this will not happen in the construction of Σ-DKDS that we present in Section 3.2.

3.1.5 Different Constructions of Metering Schemes

As suggested in [50], different metering schemes can be constructed by following the idea that we have exploited in Section 3.1.3, but by using other kinds of signature schemes different from distributed signatures: multisignatures and aggregate signatures.

In a *multisignature* scheme [68, 74] (see also [10] for a recent and efficient proposal), any user has his own secret and public keys. If many users agree in signing the same message, they can compute a partial signature each one, and then these partial signatures are combined to produce a multisignature. The

goal is that the final multisignature must be shorter than the concatenation of all the signatures. Taking the multisignature and the list of the involved users, a receiver verifies that these users have actually cooperated to sign the message.

The idea of *aggregate* signatures [14, 72] is very similar, but now the messages signed by different users can be different. Again, it is possible to combine the signatures of the different users on the different messages to produce a unique aggregate signature. The receiver takes this signature, along with the list of messages and users and verifies the correctness of all the signatures at the same time.

The resulting metering schemes would be as follows: when a client C_i visits a server S_j in a period of time λ, he signs the message $m = (S_j, \lambda)$ in the case of multisignatures, or the message $m_i = (C_i, S_j, \lambda)$ in the case of aggregate signatures. The server verifies these signatures and provides access to the honest clients. Once the server has received enough visits, he combines the signatures to produce a multisignature or an aggregate-signature, which will be the proof delivered to the audit agency who pays for advertisement, for example.

The use of multisignatures or aggregate-signatures instead of distributed signatures to construct metering schemes has two main advantages. On the one hand, any client can use his own secret and public keys to compute the signature that he sends to the servers; therefore, the key generation process of a distributed signature scheme, where each client receives a share of the secret key, must not be executed. On the other hand, with these two different approaches the audit agency can establish different criteria for the required visits, depending on the server; when using distributed signature schemes, the family $\Gamma_{\mathcal{C}}$ of subsets of clients that must visit a server was the same for all servers.

However, the use of multisignatures or aggregate signatures implies some disadvantages. On the one hand, the verification of a multisignature or an aggregate signature is usually costly (and depends on the specific signing users), whereas the verification of a distributed signature is the same as the verification of a standard signature. On the other hand, and perhaps more importantly, the fact that the identities of the signing users must be included in a multisignature or an aggregate signature would make the proofs of the servers longer and the resulting metering schemes not anonymous: anyone who would see the proof that a server sends to the audit agency (in particular, the own agency) would know what clients have visited this server. This is usually not desirable in a scenario such as the Internet.

3.2 Constructing Distributed Key Distribution Schemes

Another important task in cryptographic protocols is the distribution and management of secret keys for groups of users, when there are not secure channels among these users. The resulting group (in these scenarios they are usually known as *conferences*) key can have different purposes: it can allow the access to some restricted resource (for example on the Internet), or it can be used as the secret key of a symmetric encryption scheme to provide secure communication among the members of the group.

In [77] a single server responsible for distributing keys was introduced. This model has some drawbacks. First of all, the server must be trusted because it knows all the group keys. Furthermore, if the single server crashes, the system totally breaks down. And last, the server is a bottle-neck for the efficiency of the system, because all the key requests are directed to the same point.

To avoid the main problems of a single server, a new model was presented in [76], where the task of a single server is distributed among a set of servers. These are the so-called *distributed key distribution schemes*. Both information theoretic [9, 32] and computational [34] models have also been studied in literature.

We construct distributed and computationally secure key distribution schemes from any secure and deterministic distributed signature scheme. We consider the same framework as in [76]. Our schemes achieve the highest possible level of security, i.e. semantic security, in the random oracle model.

3.2.1 Review of Distributed Key Distribution Schemes

Let $\mathcal{S} = \{S_1, \ldots, S_\ell\}$ be a set of servers and $\mathcal{U} = \{U_1, \ldots, U_r\}$ be a group of users. Subsets of users which want to share a common key are called *conferences*, and the resulting common key is usually called *conference key*. Let $\Gamma_{\mathcal{S}} \subset 2^{\mathcal{S}}$ be an access structure defined on the set of servers, containing those subsets of servers which are able to provide conference keys to users. There will also be an adversary structure $\Lambda_{\mathcal{S}} \subset 2^{\mathcal{S}}$ containing the subsets of servers that can be corrupted by an attacker. Secure channels between users and servers are necessary.

Naor, Pinkas and Reingold [76] proposed the following model for distributed key distribution schemes: each server $S_i \in \mathcal{S}$ receives in the *initialization phase* a share of a secret value; this phase does not involve the users.

Later, the *key request phase* takes place when a user U in a conference $C \subset \mathcal{U}$ needs the conference key κ_C. He requests it to some authorized subset of servers. These servers use their secret shares to compute some partial information that they privately send to the user.

Finally, in the *key computation phase*, the conference key κ_C is computed by U from the partial information he has received from the contacted servers.

This model have been considered in other works dealing with unconditionally secure distributed key distribution schemes (see [9], for example). Nevertheless, for computationally secure schemes, a different model has been proposed [34] in order to reduce the amount of computations the user must perform to obtain the conference key. The usefulness of this model relies on those situations where servers have the main computational power.

3.2.2 Security Requirements for Distributed Key Distribution Schemes

The security level required for distributed key distribution schemes depends on the scenario where the conference members use the generated key, and basically on the type of attacks the application must be protected against. The highest possible level of security is the equivalent of *semantic security* for encryption schemes [55]. Roughly speaking, an adversary should not be able to distinguish a conference key κ_C from a value taken at random from the set of possible keys, even if it corrupts some subset of servers (in the adversary structure $\Lambda_{\mathcal{S}}$) and the protocol is executed for other conferences.

To formalize this intuitive idea, we describe here a game $\mathcal{G}_3$, played by an adversary against a distributed key distribution scheme. Note that this game is appropriate to study the security of schemes which follow the model introduced in [76] and sketched in Section 3.2.1 (for example, in this model servers send the partial information to users throughout secure point-to-point channels). For distributed key distribution schemes following other models, maybe the following game $\mathcal{G}_3$ should be modified.

1. The adversary is given the public parameters of the system: the set of servers $\mathcal{S} = \{S_1, \ldots, S_\ell\}$, the access and adversary structures $\Gamma_{\mathcal{S}}$ and $\Lambda_{\mathcal{S}}$ and the set of users $\mathcal{U} = \{U_1, \ldots, U_r\}$.

2. The adversary chooses a subset of servers $B \in \Lambda_{\mathcal{S}}$ to be corrupted.

3. The initialization phase of the distributed key distribution scheme, which involves only servers, is executed. The adversary obtains all the

public information and the secret information of the corrupted servers $S_b \in B$.

4. The adversary chooses Q_{conf} different conferences $C' \subset \mathcal{U}$. Request phase and key computation phase of the scheme are executed for all the users in these conferences. As a result, the adversary obtains all the information produced in these executions (one can think that the adversary has corrupted the members of this conference C'): the partial information provided by all the servers, the resulting conference keys $\kappa_{C'}$, etc.

5. The adversary chooses a conference C different from the conferences it has queried in the previous step. For this conference, request and key computation phases are executed, and the adversary obtains the information that the corrupted servers in B produce in these executions.

6. A random bit $b^* \in \{0,1\}$ is chosen. If $b^* = 1$, the adversary is given the real key κ_C. If $b^* = 0$, the adversary is given an element taken uniformly at random from the set of possible conference keys.

7. The adversary outputs a bit $b' \in \{0,1\}$. It wins the game if $b' = b^*$.

We define the *advantage* of such an adversary in breaking the semantic security of the distributed key distribution scheme as

$$\varepsilon = \Pr[b' = b^*] - \frac{1}{2}.$$

Definition 3.4. *Such an adversary $(\mathcal{U}, \mathcal{S}, \Gamma_{\mathcal{S}}, \Lambda_{\mathcal{S}}, T, \varepsilon, Q_{conf})$-breaks the distributed key distribution scheme if its running time is at most T, and its advantage in breaking the semantic security of the scheme is at least ε.*

3.2.3 The New Construction: Σ-DKDS's

Next, we present a new method to construct computationally secure distributed key distribution schemes, from a distributed signature scheme. We follow the model introduced of [76], explained in Section 3.2.1. This construction is possible provided the considered signature scheme is deterministic.

Let Σ be a deterministic $(\mathcal{S}, \Gamma_{\mathcal{S}})$-distributed signature scheme defined on the set of servers $\mathcal{S}$, with access structure $\Gamma_{\mathcal{S}}$, and secure with respect to an adversary structure $\Lambda_{\mathcal{S}}$. We will assume that these structures satisfy the necessary condition for robustness, $\Lambda^c_{\mathcal{S}} \subset \Gamma_{\mathcal{S}}$ (see Section 2.2). We can

define the family of subsets of servers $\Omega = \Omega(\Gamma_{\mathcal{S}}, \Lambda_{\mathcal{S}}) = \{R \subset \mathcal{S}$ such that $R - B \in \Gamma_{\mathcal{S}}$ for all $B \in \Lambda_{\mathcal{S}}\}$. Note that Ω is not empty because, in particular, $\mathcal{S} \in \Omega$. This monotone increasing family is known as the family of *robust* subsets.

We refer to the distributed key distribution scheme constructed below as Σ-*DKDS*.

Initialization phase. In this phase, the *Dist-Key-Gen* protocol of the distributed signature scheme Σ is performed, taking as players the set of servers $\mathcal{S}$. At the end of this phase, each server S_i has a share sk_i of the secret key. Furthermore, a hash function $\hat{H} : \{0,1\}^* \to \mathcal{K}$ is publicly chosen, where $\mathcal{K}$ is the set of possible conference keys (also known as *key-space*). In the security analysis, we will assume that the hash function $\hat{H}$ behaves as a totally random function.

Key request phase. The total set of users is denoted as $\mathcal{U}$. A user U in a conference $C \subset \mathcal{U}$ contacts with all the servers of some robust subset of servers $R \in \Omega$, requiring the key κ_C of the conference C. After checking for membership of U in C, every server $S_i \in R$ performs the first part of the *Dist-Sign* protocol of the scheme Σ. In this way, he uses his private information to compute the partial signature $\theta_i(h(C))$ and sends it privately to the user. Here h is a public and collision-resistant hash function whose outputs are valid messages for the scheme Σ. The server must also send to the user a proof of correctness of his partial signature.

Key computation phase. Once having received the answers from the servers, user U checks the proof of correctness of partial signatures he has received in the previous phase. Note that here we are assuming that the scheme Σ is robust: the user detects incorrect partial signatures, but the correct ones are enough to finish the protocol, because $R \in \Omega$. Using the combiner algorithm of the scheme Σ, user U takes valid partial signatures from an authorized subset of servers and produces from them a standard signature $\theta(h(C))$ on the message $h(C)$. Finally, the conference key for C is $\kappa_C = \hat{H}\Big(\theta(h(C))\Big) \in \mathcal{K}$.

Note that all users in the conference C obtain the same conference key κ_C since the distributed signature scheme Σ is deterministic.

Although the scheme Σ does not need to be non-interactive to enable the construction of Σ-DKDS's, it is very recommendable to use a non-interactive distributed signature scheme Σ. This means that servers must not contact with the other servers before computing the partial signature and sending this information to the users. In this way, the system is more

reliable and efficient.

Again, if we use our RSA distributed signature scheme for general access structures, explained in Chapter 2, we can consider general access structures for the subsets of servers allowed to provide conference keys, and general adversary structures for the sets of servers that can be corrupted by an attacker.

3.2.4 Security of Σ-DKDS's

We base the semantic security of the scheme Σ-DKDS on the unforgeability of the yielding distributed signature scheme Σ under adaptive chosen-message attacks, assuming as well that the hash function $\hat{H}$ behaves as a random oracle.

Theorem 3.2. *If there exists an adversary which $(\mathcal{U}, \mathcal{S}, \Gamma_{\mathcal{S}}, \Lambda_{\mathcal{S}}, T, \varepsilon, Q_{conf})$-breaks a Σ-DKDS, and the hash function $\hat{H}$ behaves as a random oracle, then there exists a $(\mathcal{S}, \Gamma_{\mathcal{S}}, \Lambda_{\mathcal{S}}, T', \varepsilon', Q, Q_s)$-forger against the distributed signature scheme Σ, with $Q = 0$, $Q_s = Q_{conf}$, $T' \leq T + Q_{conf}$ and $\varepsilon' \geq \varepsilon - \delta$, where δ is a negligible function of the security parameter of Σ.*

Proof. We denote as $\mathcal{F}_{DKDS}$ the adversary which $(\mathcal{U}, \mathcal{S}, \Gamma_{\mathcal{S}}, \Lambda_{\mathcal{S}}, T, \varepsilon, Q_{conf})$-breaks the Σ-DKDS. This adversary plays game $\mathcal{G}_3$, described in Section 3.2.2. We are going to construct a $(\mathcal{S}, \Gamma_{\mathcal{S}}, \Lambda_{\mathcal{S}}, T', \varepsilon', Q, Q_s)$-forger $\mathcal{F}_\Sigma$ against the distributed signature scheme Σ, which will use the forger $\mathcal{F}_{DKDS}$ as a subroutine.

The goal of the forger $\mathcal{F}_\Sigma$ is to play game $\mathcal{G}_1$ (see Section 2.1.1) and obtain a new valid signature. In the first steep, $\mathcal{F}_\Sigma$ receives the set of ℓ players $\mathcal{P}$, the access structure Γ and the adversary structure Λ.

$\mathcal{F}_\Sigma$ executes then the forger $\mathcal{F}_{DKDS}$, simulating the environment of $\mathcal{F}_{DKDS}$ in game $\mathcal{G}_3$, by using the information it obtains from its game $\mathcal{G}_1$.

First of all, $\mathcal{F}_\Sigma$ gives to $\mathcal{F}_{DKDS}$ the public parameters: it chooses a set of users $\mathcal{U}$, imposes the set of ℓ servers to be $\mathcal{S} = \mathcal{P}$ and the access and adversary structures to be $\Gamma_{\mathcal{S}} = \Gamma$ and $\Lambda_{\mathcal{S}} = \Lambda$.

By definition, in step 2 of game $\mathcal{G}_3$ the adversary $\mathcal{F}_{DKDS}$ chooses a subset of servers $B \in \Lambda_{\mathcal{S}}$ to be corrupted. For its game $\mathcal{G}_1$, forger $\mathcal{F}_\Sigma$ chooses the same set B of corrupted players.

In step 3 of game $\mathcal{G}_3$, the initialization phase of the distributed key distribution scheme, which involves only servers, is executed. The adversary $\mathcal{F}_{DKDS}$ obtains all the public information and the secret information of the corrupted servers $S_b \in B$. To simulate this information, forger $\mathcal{F}_\Sigma$ goes into step 4 of its game $\mathcal{G}_1$: the *Dist-Key-Gen* protocol of Σ is executed, and the

forger obtains all the public information and the shares of the secret key which correspond to the corrupted players, in B. Forger $\mathcal{F}_\Sigma$ gives to the adversary $\mathcal{F}_{DKDS}$ all this information.

In step 4 of game $\mathcal{G}_3$, the adversary $\mathcal{F}_{DKDS}$ chooses Q_{conf} different conferences $C' \subset \mathcal{U}$. Request phase and key computation phase of the scheme are executed for these conferences. As a result, the adversary obtains all the information produced in these executions: the partial information $\theta_i(h(C'))$ provided by all the servers, and the resulting conference keys $\kappa_{C'} = \hat{H}\Big(\theta(h(C'))\Big)$. The forger $\mathcal{F}_\Sigma$ executes step 5 of its game $\mathcal{G}_1$: it chooses the messages $h(C')$ to be signed by its signing oracle. For these messages, the *Dist-Sig* protocol of Σ is executed. The adversary $\mathcal{F}_\Sigma$ sees all the information that is made public in these executions of *Dist-Sig*, as well as private information corresponding to the corrupted players. In particular, it obtains all the partial signatures $\theta_i(h(C'))$, and the final signatures $\theta(h(C'))$, from which it can compute the conference keys $\kappa_{C'} = \hat{H}\Big(\theta(h(C'))\Big)$. Forger $\mathcal{F}_\Sigma$ gives to the adversary $\mathcal{F}_{DKDS}$ all this information.

Note that the number of times that $\mathcal{F}_\Sigma$ queries the signing oracle is at most $Q_s = Q_{conf}$.

In step 5 of game $\mathcal{G}_3$, the adversary chooses a conference C different from the conferences it has queried in the previous step, and obtains the partial information produced by the corrupted server in an execution of the computational phase for this conference. The adversary $\mathcal{F}_\Sigma$ can execute step 5 of its game $\mathcal{G}_1$ with input message $h(C)$. Then $\mathcal{F}_\Sigma$ sends the values $\theta_b(h(C))$ to the adversary $\mathcal{F}_{DKDS}$, for all the corrupted servers $S_b \in B$.

In step 6 of game $\mathcal{G}_3$, the adversary $\mathcal{F}_{DKDS}$ must receive either the conference key $\kappa_C = \hat{H}\Big(\theta(h(C))\Big)$ or a value κ^* taken at random from the key-space $\mathcal{K}$. We are assuming, however, that $\hat{H}$ behaves as a random function; therefore, the real conference key κ_C is also a uniform and random value in $\mathcal{K}$. To simulate this step, forger $\mathcal{F}_\Sigma$ chooses uniformly at random an element $\tilde{\kappa}$ from the key-space $\mathcal{K}$, and sends it to the adversary $\mathcal{F}_{DKDS}$.

Finally, in step 7 of game $\mathcal{G}_3$, $\mathcal{F}_{DKDS}$ tries to guess if the received element $\tilde{\kappa}$ is equal to $\kappa_C = \hat{H}\Big(\theta(h(C))\Big)$ or not. By hypothesis, and taking off the case where it guesses at random (with probability 1/2), the adversary $\mathcal{F}_{DKDS}$ guesses which is the case with probability, or advantage, at least ε.

Since the probability distribution of both $\kappa_C = \hat{H}\Big(\theta(h(C))\Big)$ and κ^* are the same, the only way $\mathcal{F}_{DKDS}$ has to distinguish between these two random values in $\mathcal{K}$ is by obtaining the pre-image $\theta(h(C))$ of κ_C by the hash function $\hat{H}$.

As long as the hash function h is assumed to be collision-resistant, the probability δ that $h(C) = h(C')$, for some conference $C' \neq C$ is negligible in the security parameter k of Σ, provided the outputs of h belong to a set whose size is exponential in k.

Summing up, with probability $\varepsilon' \geq \varepsilon - \delta$, the adversary $\mathcal{F}_{DKDS}$, and therefore the forger $\mathcal{F}_\Sigma$, they both have obtained a valid signature $\theta(h(C))$ for some message $h(C)$ which is different from the messages that $\mathcal{F}_\Sigma$ has asked to its signing oracle in the execution of game $\mathcal{G}_1$. That is, it has obtained a successful forgery against the distributed signature scheme Σ.

The running time T' of forger $\mathcal{F}_\Sigma$ consists of the running time of $\mathcal{F}_{DKDS}$ plus the time of making the Q_{conf} queries to its signing oracle. We can thus write $T' \leq T + Q_{conf}$.

□

Note the importance of the hash function $\hat{H}$ in the description of our Σ-DKDS. For example, if we do not use it in order to avoid the assumption of the random oracle model, then the schemes do not achieve semantic security. In effect, if we define the resulting key as $\kappa_C = \theta(h(C))$, then any adversary can use the publicly available verification protocol of the signature scheme Σ, to distinguish which between κ_C and a random value κ^* is actually a valid signature of the message $h(C)$. In this case, the schemes would achieve a weaker level of security, against an adversary whose goal is limited to compute the key κ_C.

Chapter 4

New Ring Signature Schemes for Disc-Log Scenarios

In this chapter, we study ring signature schemes. First of all, we introduce a family of ring signature schemes that we call *generic*. For this family, we prove a theoretic result, the Ring Forking Lemma, which will be useful later to prove the security of generic ring signature schemes.

Then, we propose a specific generic ring signature scheme, which follows the idea of Schnorr's signature scheme. In this scheme, all the users must generate their pair of secret and public keys by using the same parameters: two large prime numbers p and q such that $q|p-1$, and an element $g \in \mathbb{Z}_p^*$ with order q. The secret key of a user is a random element $x \in \mathbb{Z}_q^*$, whereas the matching public key is $y = g^x \bmod p$. This is usually known as *Discrete Logarithm scenario*, or Disc-Log scenario for short, because the Discrete Logarithm problem in $\langle g \rangle$ must be hard to solve in order to ensure security.

Finally, we extend this particular ring signature scheme to distributed situations, where a subset of users want to cooperate to compute a distributed anonymous signature on a message. As we have said before, we prove the security of the two specific proposals by using the Ring Forking Lemma for generic schemes.

Since all users of the system must share the same Disc-Log security parameters in our proposals, the resulting schemes can be specially useful in situations such as companies, small organizations, forums on the Internet, etc.

4.1 Ring Signature Schemes

The idea of a ring signature is the following: a user wants to compute a signature on a message, on behalf of a set (or ring) of users $\mathcal{U} = \{U_1, \ldots, U_n\}$ which includes himself. He wants the verifier of the signature to be convinced that the signer of the message is in effect some of the members of this ring. But he wants to remain completely anonymous. That is, nobody will know which member of the ring is the actual author of the signature.

4.1.1 Definitions and Applications

A ring signature scheme consists of three protocols:

1. **Key generation.** This protocol is executed individually by each user U_i of the system. The input is a security parameter and (possibly) some public parameters, common to all the users of the system. The output consists of a public key PK_i, that the users makes public, and a secret key SK_i, that U_i keeps secret.

2. **Ring signature generation.** Suppose a user U_s wants to compute a ring signature on a message m on behalf of a ring $\mathcal{U} = \{U_1, \ldots, U_n\}$ which contains himself. Then U_s must execute this protocol, taking as input the message m, the public keys $PK_1, \ldots, PK_n$ and his own secret key SK_s. The output is a signature θ.

3. **Ring signature verification.** The recipient of a ring signature checks its validity by running this protocol. It takes as input the message m, the signature θ and all the public keys $PK_1, \ldots, PK_n$ of the ring $\mathcal{U}$. The output is 1 if the signature is valid, and 0 if it is invalid.

In order to properly achieve their functionality, ring signature schemes must satisfy three properties, that we informally describe below.

1. **Correctness:** if a ring signature is generated by following the protocol correctly, then the result of the verification is always 1.

2. **Anonymity:** any verifier should not have probability greater than $1/n$ to guess the identity of the real signer who has computed a ring signature on behalf of a ring of n members.

3. **Unforgeability:** among all the proposed definitions of unforgeability, we consider the strongest one, which is existential unforgeability against chosen message-ring attacks. This means that any attacker must have

> negligible probability of success in forging a valid ring signature for some message on behalf of a ring that does not contain himself, even if he knows valid ring signatures for messages and rings, different from the pair message-ring of the forged signature, that he can adaptively choose.

Note that the anonymity property could be extended to require that even a collusion of t members in a ring of n members have probability at most $1/n - t$ to guess which other member of the ring has computed a signature on behalf of this ring. All the constructions that we present in this thesis satisfy this strongest notion of anonymity, as well.

The anonymity property must hold unconditionally, whereas the unforgeability property usually holds computationally. This means that an adversary with unlimited computational and time resources could forge a signature, but could not extract any information about the author of a ring signature.

The reason of this difference is the following: one can avoid computational attacks against the unforgeability of the scheme, by modifying the keys of the users once a year, for example. However, identities do not change. Therefore, if an unbounded adversary could break the anonymity in, for example, 3 years, then the authority of a message would be unmasked, even if the keys have been modified during these 3 years. To avoid this, we require unconditional anonymity.

Ring signatures are a useful tool to provide anonymity in some scenarios. For example, if a member of a group wants to leak to the media a secret information about the group, he can sign this information using a ring scheme. Everybody will be convinced that the information comes from the group itself, but anybody could accuse him of leaking the secret.

A different application is the following: if the signer A of a message wants that the authorship of a message could be entirely verified only by some specific user B, he can sign the message with respect to the ring $\{A, B\}$. The rest of users could not know who between A and B is the author of the signature, but B will be convinced that the author is A.

Recently, ring signature schemes have been also used as a primitive to construct a different kind of signatures, *concurrent* signatures [24].

4.1.2 State of the Art

Although the first proposals of ring signature schemes can be found in the scenario of group signatures (see [22, 28, 18]), the concept was formally introduced by Rivest, Shamir and Tauman in [86]. They propose a scheme

that they prove unforgeable, in the ideal cipher model (which is a different paradigm for proving security of cryptographic protocols), assuming the hardness of the RSA problem. This scheme also uses a symmetric encryption scheme and the notion of combining functions.

Bresson, Stern and Szydlo show in [16] that the scheme of [86] can be modified in such a way that the new scheme is proved to achieve the same level of security, but under the strictly weaker assumption of the random oracle model. They also introduce the concept of threshold ring signature schemes, which has been the focus of some other works, for example [94].

In [1], Abe, Ohkubo and Suzuki design some general ring signature schemes where the public keys of the users can be totally independent: different sizes, and different types of keys (RSA keys, discrete logarithm keys,...).

Recently, Dodis, Kiayias, Nicolosi and Shoup [41] have proposed the first ring signature scheme where the length of the final signature is not linear with respect to the number of members of the ring (the result is valid if the same ring is used by the same anonymous signer for many different messages).

Finally, the only identity-based ring signature scheme proposed until now is the one by Zhang and Kim [95]. A formal proof of the security of this scheme has been given in [57]. We will review the concept of identity-based cryptography in Chapter 5.

4.2 Generic Ring Signature Schemes

We define a family of ring signature schemes that we call *generic* (influenced by the work of Pointcheval and Stern [83], where they give this name to a family of signature schemes which includes Schnorr's one), and for which the results in this section are valid. Consider a security parameter k, a hash function which outputs k-bit long elements, and a ring $\mathcal{U} = \{U_1, \ldots, U_n\}$ of n members. Given the input message m, a generic ring signature scheme produces a tuple $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$, where $R_1, \ldots, R_n$ take their values randomly in a large set G in such a way that $R_i \neq R_j$ for all $i \neq j$, h_i is the hash value of $(\mathcal{U}, m, R_i)$, for $1 \leq i \leq n$, and the value σ is fully determined by $R_1, \ldots, R_n, h_1, \ldots, h_n$ and the message m.

Another required condition is that no R_i can appear in a signature with probability greater than $1/2^{k-1}$, where k is the security parameter. This condition can be achieved by choosing the set G as large as necessary, and is required in order to reduce the probability of collisions in the security proofs, when the behaviour of hash functions is modeled with random oracles [5]. In the framework that we consider, the outputs of the random oracle will be

k-bit long elements.

The basic idea of the Forking Lemmas in [83] and the Ring Forking Lemma that we are going to introduce is the following: assuming that an adaptive chosen message attack can forge a generic ring signature, another attacker could obtain, by replaying the first attacker with randomly chosen hash functions (i.e. random oracles), two forged ring signatures of the same message m, the same ring $\mathcal{U}$ and with the same randomness $R_1, \ldots, R_n$, but with different values of the other components of the signature. Then, these two forged signatures could be used to solve some computational problem which is assumed to be intractable. In this way, the corresponding ring signature scheme is proved to be existentially unforgeable under adaptive chosen message attacks.

4.2.1 A Security Result for Generic Ring Signature Schemes

We first state a well-known lemma (known as *Splitting Lemma* in [83] and known as *Heavy-Row Lemma* in [80], for example) that we will need in the proof of our Ring Forking Lemma and in the security proof of a scheme that we propose in Chapter 5. Figure 4.1 below gives an intuitive (although simple) idea of this result (in particular, the sets X and Y in the figure are not discrete!).

Lemma 4.1. *(The Splitting Lemma). Let X and Y two finite sets where two probability distributions are considered. Let $A \subset X \times Y$ be a set such that $\Pr[A] \geq \epsilon$, where the probability distribution in $X \times Y$ is the joint probability distribution induced by the distributions in X and Y. For any $\alpha < \epsilon$, define*

$$B = \{(x, y) \in X \times Y | \Pr_{y' \in Y}[(x, y') \in A] \geq \epsilon - \alpha\} \text{ and } \bar{B} = X \times Y \ - B,$$

then the following statements hold:

1. $\Pr[B] \geq \alpha$.
2. *for any $(x, y) \in B$, $\Pr_{y' \in Y}[(x, y') \in A] \geq \epsilon - \alpha$.*
3. $\Pr[B|A] \geq \alpha/\epsilon$.

Proof. The definition of the sets B and $\bar{B}$ implies that $\Pr\left[A|\bar{B}\right] < \epsilon - \alpha$. In effect, if this would not be the case, then we would have

$$\epsilon - \alpha \leq \Pr\left[A|\bar{B}\right] = \sum_{x_0 \in \bar{B}} \Pr[x = x_0] \cdot \Pr_y[(x_0, y) \in A] < (\epsilon - \alpha) \cdot \sum_{x_0 \in \bar{B}} \Pr[x = x_0] \leq \epsilon - \alpha,$$

a contradiction. We can prove now the three statements of the lemma:

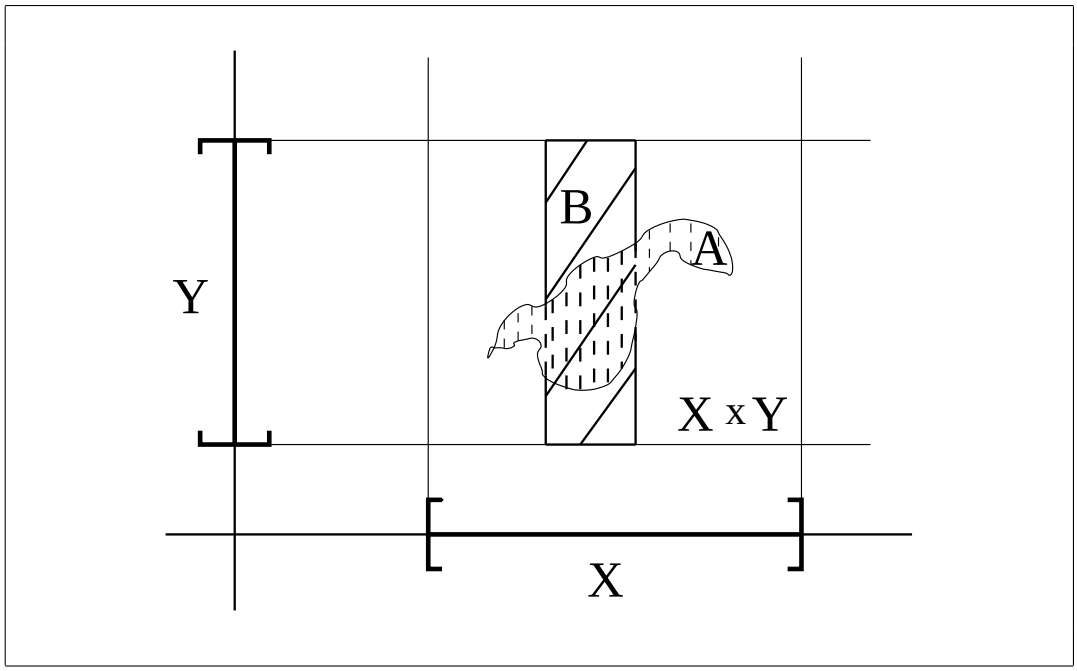

Figure 4.1: The idea behind the Splitting Lemma

1. For the sake of contradiction, we assume that $\Pr[B] < \alpha$. Then we have

$$\epsilon \leq \Pr[A] = \Pr[B] \cdot \Pr[A|B] + \Pr\left[\bar{B}\right] \cdot \Pr\left[A|\bar{B}\right] < \alpha \cdot 1 + 1 \cdot (\epsilon - \alpha) = \epsilon.$$

2. Trivial, by definition of the set B.

3. We can use Bayes' law to see that:

$$\Pr[B|A] = 1 - \Pr\left[\bar{B}|A\right] = 1 - \frac{\Pr\left[A|\bar{B}\right] \cdot \Pr\left[\bar{B}\right]}{\Pr[A]} \geq 1 - \frac{(\epsilon - \alpha) \cdot 1}{\epsilon} = \alpha/\epsilon\,.$$

□

Now we are ready to state and prove the Ring Forking Lemma, that will be useful to prove the unforgeability of generic ring signature schemes. For integers Q and n such that $Q \geq n \geq 1$, we denote as $V_{Q,n}$ the number of different n-permutations of Q elements (without repetitions); that is, $V_{Q,n} = Q(Q-1) \cdot \ldots \cdot (Q-n+1)$.

Theorem 4.1. *(The Ring Forking Lemma). Consider a generic ring signature scheme with security parameter $k \geq 6$. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data and which can ask Q queries to the random oracle that models the hash function of the scheme.*

We assume that $\mathcal{A}$ produces, within time bound T and with probability of success at least ε, a valid ring signature $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ for a ring $\mathcal{U}$ of n users.

Then, within time $T' \leq 2T$, and with probability $\varepsilon' \geq \frac{\varepsilon^2}{65 V_{Q,n}}$, by executing $\mathcal{A}$ with random instantiations of the hash function we can obtain two valid ring signatures $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ and $(\mathcal{U}, m, R_1, \ldots, R_n, h'_1, \ldots, h'_n, \sigma')$ such that $h_j \neq h'_j$, for some $j \in \{1, \ldots, n\}$ and $h_i = h'_i$ for all $i = 1, \ldots, n$ such that $i \neq j$.

Proof. The Turing machine $\mathcal{A}$, with random tape ω, can ask Q queries to the random oracle H. We denote by $\mathcal{Q}_1, \ldots, \mathcal{Q}_Q$ the Q distinct queries and by $\rho = (\rho_1, \ldots, \rho_Q)$ the list of the Q answers of the random oracle H. So we can see a random choice of the random oracle H as a random choice of such a vector ρ.

Now, for a random choice of (ω, H) and with probability ε, the machine $\mathcal{A}$ outputs a valid ring signature $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$. Since H is a random oracle and its outputs are k-bit long elements, the probability that there exists some index i such that $\mathcal{A}$ has not asked the query $(\mathcal{U}, m, R_i)$ to the random oracle is less than $n/2^k$. Therefore, with probability at least $1 - n/2^k$, $\mathcal{A}$ has asked all the queries $(\mathcal{U}, m, R_i)$ to the oracle, for $1 \leq i \leq n$, and so we have that $Q \geq n$ is necessary (this will be the case when considering any forger against any generic ring signature scheme).

With probability at least $\varepsilon\left(1 - n/2^k\right)$, the machine $\mathcal{A}$ is successful in forging a ring signature $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ and besides it has asked to the random oracle all the queries $(\mathcal{U}, m, R_i)$, for $i = 1, \ldots, n$. In this case, for all index i there exists an integer $\ell_i \in \{1, 2, \ldots, Q\}$ such that the query $\mathcal{Q}_{\ell_i}$ is precisely $(\mathcal{U}, m, R_i)$. Then, we define $L(\omega, H) = (\ell_1, \ell_2, \ldots, \ell_n)$ and $\beta(\omega, H) = \max\{\ell_i \mid (\ell_1, \ell_2, \ldots, \ell_n) = L(\omega, H)\}$. Note that, since the forged ring signature is a valid generic one, we have that all the R_i's are different, and so the integers ℓ_i are also all different.

In the unlikely case where $\mathcal{A}$ has not asked for some of the queries $(\mathcal{U}, m, R_i)$ to the random oracle, then we say that $\ell_i = \infty$, and so $\beta(\omega, H) = \infty$. Now we define the sets

$$\mathcal{S} = \{(\omega, H) \mid \mathcal{A}(\omega, H) \text{ succeeds and } \beta(\omega, H) \neq \infty\} ,$$
$$\mathcal{S}_{\vec{\ell}} = \{(\omega, H) \mid \mathcal{A}(\omega, H) \text{ succeeds and } L(\omega, H) = \vec{\ell}\} ,$$

for all the vectors $\vec{\ell} \in L_n = \{(\ell_1, \ell_2, \ldots, \ell_n) \mid \ell_i \in \{1, 2, \ldots, Q\}$ and $\ell_i \neq \ell_j$ for all $i \neq j\}$. Note that the cardinality of the set L_n is the number of n-permutations of Q elements, that is, $V_{Q,n} = Q(Q-1) \cdot \ldots \cdot (Q-n+1)$. Furthermore, the set $\{\mathcal{S}_{\vec{\ell}} \mid \vec{\ell} \in L_n\}$ is a partition of $\mathcal{S}$. The pairs (ω, H) in the set $\mathcal{S}$ are called the successful pairs. We can find the lower bound $\nu = \Pr[\mathcal{S}] \geq \varepsilon\left(1 - n/2^k\right)$. With this probability ν, the machine $\mathcal{A}$ gets one pair (ω, H) in $\mathcal{S}$.

Let I be the set formed by the most likely vectors, $I = \{\vec{\ell} \in L_n \mid \Pr[\mathcal{S}_{\vec{\ell}} \mid \mathcal{S}] \geq \frac{1}{2V_{Q,n}}\}$. Probabilities are taken over the random choice of (ω, H). The following lemma asserts that, in case of success, the corresponding vector of indexes lies in I with probability at least $\frac{1}{2}$.

Lemma 4.2. $\Pr[L(\omega, H) \in I \mid \mathcal{S}] \geq \frac{1}{2}$.

Proof. Since the sets $\mathcal{S}_{\vec{\ell}}$ are disjoint, we have that $\Pr[L(\omega, H) \in I \mid \mathcal{S}] = \sum_{\vec{\ell} \in I} \Pr[\mathcal{S}_{\vec{\ell}} \mid \mathcal{S}]$. This probability is equal to $1 - \sum_{\vec{\ell} \notin I} \Pr[\mathcal{S}_{\vec{\ell}} \mid \mathcal{S}]$. Since the complement of I contains fewer than $V_{Q,n}$ vectors, this probability is at least $1 - V_{Q,n} \cdot \frac{1}{2V_{Q,n}} = \frac{1}{2}$. □

Let us return to the proof of Theorem 4.1. For each vector $\vec{\ell} \in I$, if we denote by $\beta_{\vec{\ell}}$ the maximum of the coordinates of $\vec{\ell}$, we can apply the Splitting Lemma (Lemma 4.1). Following the notation of this lemma, and if we see the oracle H as a random vector $(\rho_1, \ldots, \rho_Q)$, then $A = \mathcal{S}_{\vec{\ell}}$, $X = \{(\omega, \rho_1, \ldots, \rho_{(\beta_{\vec{\ell}}-1)})\}$ and $Y = \{(\rho_{\beta_{\vec{\ell}}}, \ldots, \rho_Q)\}$. We also refer to $(\rho_1, \ldots, \rho_{(\beta_{\vec{\ell}}-1)})$ as $H_{\beta_{\vec{\ell}}}$, the restriction of H to queries of index strictly less than $\beta_{\vec{\ell}}$. Since $\Pr[\mathcal{S}_{\vec{\ell}}] = \Pr[\mathcal{S}] \cdot \Pr[\mathcal{S}_{\vec{\ell}} \mid \mathcal{S}] \geq \frac{\nu}{2V_{Q,n}}$, we take $\epsilon = \frac{\nu}{2V_{Q,n}}$ and $\alpha = \frac{\nu}{4V_{Q,n}}$, and the Splitting Lemma proves that there exists a subset $\Omega_{\vec{\ell}}$ of executions (ω, H) such that, for any $(\omega, H) \in \Omega_{\vec{\ell}}$,

$$\Pr_{H'}[(\omega, H') \in \mathcal{S}_{\vec{\ell}} \mid H_{\beta_{\vec{\ell}}} = H'_{\beta_{\vec{\ell}}}] \geq \frac{\nu}{4V_{Q,n}} \tag{4.1}$$

$$\text{and} \quad \Pr[\Omega_{\vec{\ell}} \mid \mathcal{S}_{\vec{\ell}}] \geq \frac{1}{2}$$

Using again that the sets $\mathcal{S}_{\vec{\ell}}$ are all disjoint, along with the result in Lemma 4.2, we have that

$$\Pr[\text{there exists } \vec{\ell} \in I \text{ such that } (\omega, H) \in \Omega_{\vec{\ell}} \cap \mathcal{S}_{\vec{\ell}} \mid \mathcal{S}] = \Pr\left[\bigcup_{\vec{\ell} \in I} (\Omega_{\vec{\ell}} \cap \mathcal{S}_{\vec{\ell}}) \mid \mathcal{S}\right] =$$

$$= \sum_{\vec{\ell} \in I} \Pr[\Omega_{\vec{\ell}} \cap \mathcal{S}_{\vec{\ell}} \mid \mathcal{S}] = \sum_{\vec{\ell} \in I} \Pr[\Omega_{\vec{\ell}} \mid \mathcal{S}_{\vec{\ell}}] \cdot \Pr[\mathcal{S}_{\vec{\ell}} \mid \mathcal{S}] \geq \left(\sum_{\vec{\ell} \in I} \Pr[\mathcal{S}_{\vec{\ell}} \mid \mathcal{S}]\right)/2 \geq \frac{1}{4}\,.$$

For simplicity, we denote by $\vec{\ell}$ the vector $L(\omega, H)$ corresponding to the successful pair (ω, H) obtained by the attack $\mathcal{A}$ with probability ν, and by β the corresponding index $\beta(\omega, H)$. As we have seen, with probability at least $1/4$, we have that $\vec{\ell} \in I$ and $(\omega, H) \in \Omega_{\vec{\ell}} \cap \mathcal{S}_{\vec{\ell}}$. Therefore, with probability greater than $\nu/4$, the attack $\mathcal{A}$ has provided a successful pair (ω, H), with $\vec{\ell} = L(\omega, H) \in I$ and $(\omega, H) \in \Omega_{\vec{\ell}} \cap \mathcal{S}_{\vec{\ell}}$.

If now we replay the attack, with fixed random tape ω but randomly chosen oracle H' such that $H'_\beta = H_\beta$, we can use inequality (4.1) and thus we obtain that

$$\Pr_{H'}[(\omega, H') \in \mathcal{S}_{\vec{\ell}} \text{ and } \rho_\beta \neq \rho'_\beta \mid H'_\beta = H_\beta] =$$

$$= \Pr_{H'}[(\omega, H') \in \mathcal{S}_{\vec{\ell}} \mid H'_\beta = H_\beta] \left(1 - \Pr_{H'}[\rho_\beta = \rho'_\beta]\right) \geq \frac{\nu}{4V_{Q,n}} \cdot \frac{2^k - 1}{2^k} \ ,$$

where $\rho_\beta = H(\mathcal{Q}_\beta)$ and $\rho'_\beta = H'(\mathcal{Q}_\beta)$.

Summing up, after executing twice the attack $\mathcal{A}$, we obtain two valid ring signatures $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ and $(\mathcal{U}', m', R'_1, \ldots, R'_n, h'_1, \ldots, h'_n, \sigma')$, with total probability

$$\varepsilon' \geq \frac{\nu}{4} \cdot \frac{\nu}{4V_{Q,n}} \cdot \frac{2^k - 1}{2^k} = \frac{\varepsilon^2}{16V_{Q,n}} \cdot \frac{2^k - n}{2^k} \cdot \frac{2^k - n}{2^k} \cdot \frac{2^k - 1}{2^k} \geq$$

$$\geq \frac{\varepsilon^2}{64V_{Q,n}} \cdot \frac{2^k - 1}{2^k} \geq \frac{\varepsilon^2}{65V_{Q,n}}.$$

We have used the fact that n must be polynomial in the security parameter k, so $n \leq 2^{k/2}$, and the fact that $\frac{2^k-1}{2^k} \geq \frac{64}{65}$ if $k \geq 6$.

The two executions of $\mathcal{A}$ have the same random tape ω and the answers of the two random oracles H and H' are the same until the query $\mathcal{Q}_\beta$, where β is the index $\beta(\omega, H)$ corresponding to the first successful forgery performed by $\mathcal{A}$. Since all the values of $\mathcal{U}, m, R_i$ have been chosen before this query, we have that $\mathcal{U}' = \mathcal{U}$, $m' = m$ and $R'_i = R_i$, for all $i = 1, \ldots, n$. The two oracles H and H' verify, furthermore, that $H(\mathcal{Q}_\beta) \neq H'(\mathcal{Q}_\beta)$.

Therefore, if we denote as j the index such that $(\mathcal{U}, m, R_j)$ was the query $\mathcal{Q}_\beta$, then we have that $h_j = H(\mathcal{Q}_\beta) \neq H'(\mathcal{Q}_\beta) = h'_j$. However, the rest of queries $(\mathcal{U}, m, R_i)$, for $i = 1, \ldots, n$, $i \neq j$, have been asked before the query $\mathcal{Q}_\beta$, and so the values obtained from the oracles H and H' have been the same. That is, $h_i = h'_i$, for all $i = 1, \ldots, n$ with $i \neq j$.

□

4.3 A Specific Generic Ring Signature Scheme

In this section we propose a ring signature scheme which runs in Discrete Logarithm scenarios where all users have the same common parameters. The scheme is inspired by Schnorr's signature scheme. The point is that it is actually a generic ring signature scheme, so we will be able to apply the Ring Forking Lemma to prove its unforgeability. The different protocols of the scheme are described below.

Key generation. Given a security parameter k, let p and q be large primes such that $q|p-1$ and $q \geq 2^k + \hat{n}$, where $\hat{n}$ is the maximum possible number of users in a ring. Let g be an element of $\mathbb{Z}_p^*$ with order q, and let H be a collision resistant hash function which outputs elements in $\mathbb{Z}_q$.

Consider a set, or ring, of potential signers $\mathcal{U} = \{U_1, \ldots, U_n\}$. Every potential signer U_i has a private key $x_i \in \mathbb{Z}_q^*$ and the corresponding public key $y_i = g^{x_i} \bmod p$.

Ring signature generation. To sign a message m on behalf of the ring $\mathcal{U}$, a signer U_s, where $s \in \{1, \ldots, n\}$, acts as follows:

1. For every user $U_i \in \mathcal{U}$, $i \in \{1, \ldots, n\}$, $i \neq s$, choose a_i at random in $\mathbb{Z}_q^*$, pairwise different. Compute $R_i = g^{a_i} \bmod p$, for all $i \neq s$.

2. Choose a random $a \in \mathbb{Z}_q$.

3. Compute $R_s = g^a \prod_{i \neq s} y_i^{-H(\mathcal{U}, m, R_i)} \bmod p$. If $R_s = 1$ or $R_s = R_i$ for some $i \neq s$, then go to step 2.

4. Compute $\sigma = a + \sum_{i \neq s} a_i + x_s H(\mathcal{U}, m, R_s) \bmod q$.

5. Define the signature of the message m made by the ring $\mathcal{U} = \{U_1, \ldots, U_n\}$ to be $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$, where $h_i = H(\mathcal{U}, m, R_i)$, for all $1 \leq i \leq n$.

   Note that the values $h_1, \ldots, h_n$ can be derived from the rest of values in the signature, and so they do not need to be part of the final signature in practice. We include them for clarity in the security proofs.

Ring signature verification. The validity of the signature is verified by the recipient of the message by checking that $h_i = H(\mathcal{U}, m, R_i)$ and that

$$g^\sigma = R_1 \cdot \ldots \cdot R_n \cdot y_1^{h_1} \cdot \ldots \cdot y_n^{h_n} \bmod p.$$

4.3.1 Analysis of the Scheme

Note that the proposed scheme is in effect generic: by construction, we have that $R_i \neq R_j$ for every pair $i \neq j$, and that no R_i appears with probability greater than $\frac{1}{q-\hat{n}} \leq \frac{2}{2^k}$. Now we show that it satisfies the three required conditions for ring signature schemes.

Correctness

If the ring signature has been correctly generated, following the steps of the protocol above, then the verification result is always "True". In effect:

$$R_1 \cdot \ldots \cdot R_n \cdot y_1^{h_1} \cdot \ldots \cdot y_n^{h_n} = g^a y_s^{h_s} \prod_{i \neq s} R_i = g^{a + x_s h_s + \sum_{i \neq s} a_i} = g^\sigma \mod p.$$

Anonymity

In order to prove that our ring signature scheme is unconditionally anonymous, we must show that any attacker outside a ring of n possible users has probability $1/n$ to guess which member of the ring has actually computed a given signature on behalf of this ring.

Let $Sig = (\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ be a valid ring signature of a message m. That is, $h_i = H(\mathcal{U}, m, R_i)$ and $g^\sigma = R_1 \cdot \ldots \cdot R_n \cdot y_1^{h_1} \cdot \ldots \cdot y_n^{h_n}$. Let U_s be a member of the ring. We now find the probability that U_s computes exactly the ring signature Sig, when he produces a ring signature of message m on behalf of the ring $\mathcal{U}$, by following the method explained in Section 4.3.

The probability that U_s computes all the values $R_i \neq 1$ of Sig, pairwise different for $1 \leq i \leq n$, $i \neq s$, is $\frac{1}{q-1} \cdot \frac{1}{q-2} \cdot \ldots \cdot \frac{1}{q-n+1}$. Then, the probability that U_s chooses exactly the only value $a \in \mathbb{Z}_q$ that leads to the value R_s of Sig, among all possible values for R_s different from 1 and different from all R_i with $i \neq s$, is $\frac{1}{q-n}$.

Summing up, the probability that U_s generates exactly the ring signature Sig is

$$\frac{1}{q-1} \cdot \frac{1}{q-2} \cdot \ldots \cdot \frac{1}{q-n+1} \cdot \frac{1}{q-n} = \frac{1}{V_{q-1,n}}$$

and this probability does not depend on who U_s is, so it is the same for all the members of the ring. This fact proves the unconditional anonymity of the scheme.

Unforgeability

First of all, we formally define the exact unforgeability of a ring signature scheme, by stating the capabilities and the goals of an adversary which tries to break the scheme.

Such an adversary is given a list of $\hat{n}$ identities (or public keys), corresponding to the total set of considered users. He is allowed to corrupt up to Q_e users, obtaining their secret keys. The adversary can also make Q queries to the random oracle. Finally, the adversary can require the execution of the signing algorithm for Q_s pairs of messages and rings that he adaptively chooses, obtaining a valid ring signature.

We say that this adversary is $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-successful if he obtains in polynomial time T and with non-negligible probability at least ε a valid ring signature for some message m and some ring of users $\mathcal{U}$, such that:

(i) the pair formed by the message m and the ring $\mathcal{U}$ has not been asked to the signing oracle during the attack; and

(ii) none of the users in the ring $\mathcal{U}$ has been corrupted by the adversary.

Finally, we say that a ring signature scheme is $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-unforgeable if there does not exist any $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-successful adversary against it.

Before stating and proving the theorem that ensures the unforgeability of our scheme, we recall and prove a well-known result of elementary probability (the birthday problem is a particular case of it), which will be used in many unforgeability proofs throughout the rest of this thesis.

Fact 4.1. *The probability that at least two of K values uniformly and independently chosen from a set of q elements are equal, where $K \leq q$, is less than $\frac{K^2}{2q}$.*

Proof. We denote as A the fact that at least two of these K chosen values are equal. Let us compute the probability of the complementary of A, denoted as $\bar{A}$, that is the probability that the K elements are all different:

$$\Pr[\bar{A}] = 1 \cdot \frac{q-1}{q} \cdot \frac{q-2}{q} \cdot \ldots \cdot \frac{q-(K-1)}{q} = \left(1 - \frac{1}{q}\right) \cdot \left(1 - \frac{2}{q}\right) \cdot \ldots \cdot \left(1 - \frac{K-1}{q}\right).$$

Note now that if we have K non-negative values $\alpha_1, \ldots, \alpha_K$, then

$$\prod_{i=1}^{K} (1 - \alpha_i) \geq 1 - \sum_{i=1}^{K} \alpha_i.$$

This inequality can be easily proved by induction, for example. Applying it to $\Pr[\bar{A}]$, we obtain

$$\Pr[\bar{A}] = \prod_{i=1}^{K-1} \left(1 - \frac{i}{q}\right) \geq 1 - \sum_{i=1}^{K-1} \frac{i}{q} = 1 - \frac{K(K-1)}{2q}.$$

And finally we have $\Pr[A] = 1 - \Pr[\bar{A}] \leq \frac{K(K-1)}{2q} \leq \frac{K^2}{2q}$, as desired. □

The following theorem states that the ring signature scheme in Section 4.3 is unforgeable in the random oracle model, assuming that the Discrete Logarithm problem is hard to solve in groups of prime order.

Theorem 4.2. *Let $\mathcal{A}$ be a $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-successful attacker against the ring signature scheme presented in Section 4.3, with security parameter $k \geq 6$ such that $Q_s \leq \frac{2^{k/2}}{4}$ and $Q \leq \frac{2^{k/2}}{3}$.*

Then the Discrete Logarithm problem in $\langle g \rangle$ can be solved within time $T' \leq 2T + 2Q + 4\hat{n}T_{exp}Q_s$ and with probability $\varepsilon' \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q,\hat{n}}}$.

Proof. Let (p, q, g, y) be an instance of the Discrete Logarithm problem in the subgroup $< g >$ of $\mathbb{Z}_p$ of order q, where q is a prime that divides $p - 1$. We will use the attacker $\mathcal{A}$ against the ring signature scheme to solve this instance of the Discrete Logarithm problem.

Now we are going to construct a probabilistic polynomial time Turing machine $\mathcal{F}$ which will solve the given instance of the Discrete Logarithm problem. We will apply to $\mathcal{F}$ the result of Theorem 4.1; therefore, $\mathcal{F}$ will be allowed to make Q queries to the random oracle which models the behaviour of the hash function H. This machine $\mathcal{F}$ will use the attacker $\mathcal{A}$ as a subroutine; therefore, $\mathcal{F}$ must perfectly simulate the environment of the attacker $\mathcal{A}$. This idea is captured by Figure 4.2 below.

The machine $\mathcal{F}$ receives the public data (p, q, g, y). Then $\mathcal{F}$ runs the attacker $\mathcal{A}$ against the ring signature scheme, answering to all the queries that $\mathcal{A}$ makes.

Let us define $\mu = (5/6)^{1/Q_e}$. If we consider attacks where $Q_e = 0$, then we fix $\mu = 0$.

First $\mathcal{F}$ creates a table TAB_* in the following way. For each possible user U_i of the ring signature scheme, $\mathcal{F}$ proceeds as follows: with probability μ, it chooses the bit $c_i = 0$ and an element $x_i \in \mathbb{Z}_q^*$ at random, and defines $y_i = g^{x_i} \bmod p$. The entry (U_i, x_i, y_i, c_i) is stored in the table TAB_*. On the other hand, with probability $1 - \mu$, it chooses the bit $c_i = 1$ and an element $\alpha_i \in \mathbb{Z}_q^*$ at random, and defines $y_i = y^{\alpha_i} \bmod p$. A new entry is stored in the table TAB_*, with the values $(U_i, \alpha_i, y_i, c_i)$.

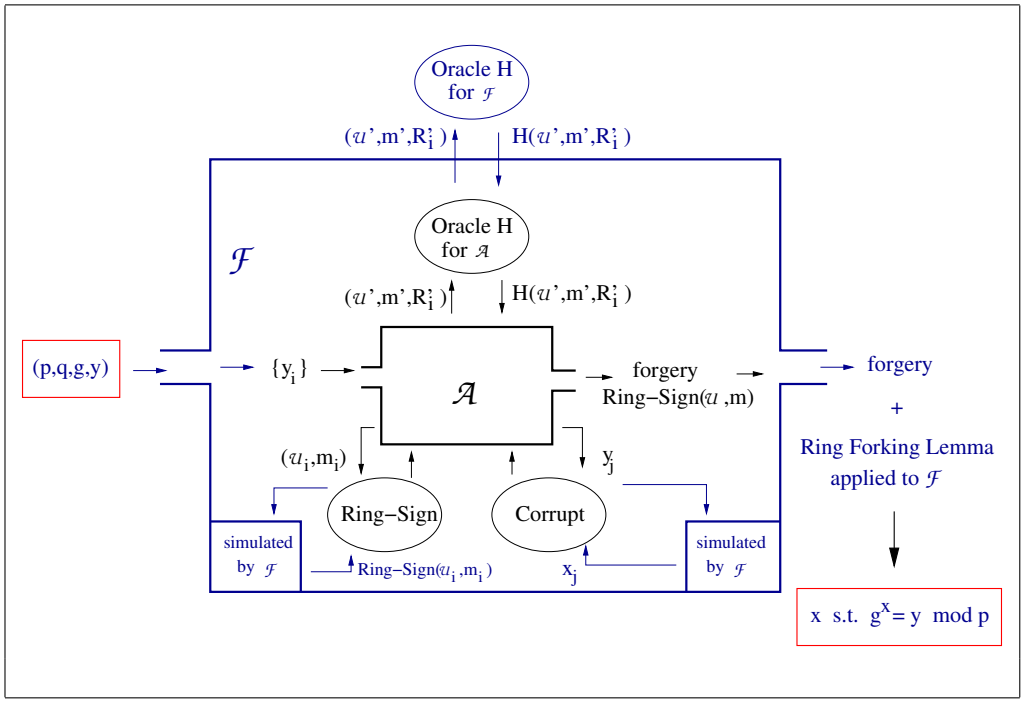

Figure 4.2: $\mathcal{F}$ executes $\mathcal{A}$ and solves the Disc-Log problem

The attacker $\mathcal{A}$ is given the list with the publics keys of the $\hat{n}$ considered users, computed as explained just above. Every time $\mathcal{A}$ asks for the secret key of a user U_i, the machine B looks for U_i in the table TAB_*. If $c_i = 0$, then $\mathcal{F}$ sends x_i to $\mathcal{A}$. If $c_i = 1$, the machine $\mathcal{F}$ cannot answer and halts. Note that the probability that $\mathcal{F}$ halts in this process is less than $1 - \mu^{Q_e} \leq 1/6$.

To answer the queries of $\mathcal{A}$ to the random oracle, we recall that $\mathcal{F}$ has access to the random oracle which models the hash function H; therefore, $\mathcal{F}$ must only send to $\mathcal{A}$ the answers that it obtains from its oracle.

The attacker $\mathcal{A}$ can ask Q_s times for a valid ring signature for messages m' and rings $\mathcal{U}'$ of $n \leq \hat{n}$ members (for simplicity, we denote $\mathcal{U}' = \{U'_1, \ldots, U'_n\}$). We assume that $\mathcal{A}$ has not asked for any of the secret keys of the ring $\mathcal{U}'$ (otherwise, $\mathcal{A}$ could obtain a valid ring signature by itself, by using the signing algorithm explained in Section 4.3). To answer such a query, the machine $\mathcal{F}$ proceeds as follows:

1. Choose at random an index $s \in \{1, \ldots, n\}$.

2. For all $i \in \{1, \ldots, n\}$, $i \neq s$, choose a_i at random in $\mathbb{Z}_q^*$, pairwise different. Compute $R'_i = g^{a_i} \mod p$, for all $i \neq s$.

3. Choose at random $h'_s \in \mathbb{Z}_q$. For $i \neq s$, compute $h'_i = H(\mathcal{U}', m', R'_i)$ (asking to the random oracle for H).

4. Choose at random $\sigma' \in \mathbb{Z}_q$.

5. Compute $R'_s = g^{\sigma' - \sum_{i \neq s} a_i} y_1^{-h'_1} \dots y_n^{-h'_n}$. If $R'_s = 1$ or $R'_s = R'_i$ for some $i \neq s$, then go to step 4.

6. Now $\mathcal{F}$ "falsifies" the random oracle H, by imposing the relation $H(\mathcal{U}', m', R'_s) = h'_s$. Later, if $\mathcal{A}$ asks to the random oracle H for this input, then $\mathcal{F}$ will answer with h'_s. Since h'_s is a random value and we are in the random oracle model for H, this relation seems consistent to $\mathcal{A}$.

7. Return the tuple $(\mathcal{U}', m', R'_1, \dots, R'_n, h'_1, \dots, h'_n, \sigma')$.

In each simulation, $\mathcal{F}$ must perform $2n$ modular exponentiations. There is some risk of "collisions" in the management of the random oracle, because of the falsification performed by $\mathcal{F}$ in step 6 of the simulation above. Remember that no R_i can appear with probability greater than $1/2^{k-1}$ in a ring signature.

Two kinds of collisions can occur:

- A tuple $(\mathcal{U}', m', R'_s)$ that $\mathcal{F}$ outputs, as part of a simulated ring signature, has been asked before to the random oracle by $\mathcal{A}$. In this case, it is quite unlikely that the relation $H(\mathcal{U}', m', R'_s) = h'_i$ corresponding to the values output in the simulation of the signature coincides with the relation previously stored in TAB_H. The probability of such a collision is, however, less than $Q \cdot Q_s \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$.

- Two tuples that $\mathcal{F}$ outputs inside two different simulated ring signatures are exactly equal. Using Fact 4.1, we have that the probability of this collision is less than $\frac{Q_s^2}{2} \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$.

Altogether, the probability of collisions is less than $1/3$. Now we can compute the success probability of the machine $\mathcal{F}$; that is, the probability that $\mathcal{F}$ obtains a valid ring signature:

$$\varepsilon_{\mathcal{F}} = \Pr[\mathcal{F} \text{ succeeds}] =$$

$$\Pr[\mathcal{F} \text{ does not halt AND no-collisions in the simulations AND } \mathcal{A} \text{ succeeds}] \geq$$

$$\geq \Pr[\mathcal{A} \text{ succeeds} \mid \mathcal{F} \text{ does not halt AND no-collisions in the simulations }] \cdot$$

$$\cdot (1 - \Pr[\mathcal{F} \text{ halts OR collisions in the simulations}]) \geq \varepsilon \left(1 - \frac{1}{6} - \frac{1}{3}\right) = \frac{\varepsilon}{2}.$$

On the other hand, the execution time of the machine $\mathcal{F}$ is $T_{\mathcal{F}} \leq T + Q + 2\hat{n}T_{exp}Q_s$. Summing up, we have a Turing machine $\mathcal{F}$ that forges a generic ring signature scheme in time $T_{\mathcal{F}}$ and with probability $\varepsilon_{\mathcal{F}} \geq \varepsilon/2$. We can apply Theorem 4.1 to the machine $\mathcal{F}$.

This means that, by executing twice the machine $\mathcal{F}$, we will obtain in time $T' \leq 2T_{\mathcal{F}}$ and with probability $\tilde{\varepsilon}' \geq \frac{\varepsilon_{\mathcal{F}}^2}{65V_{Q,\hat{n}}}$ two valid ring signatures $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ and $(\mathcal{U}, m, R_1, \ldots, R_n, h'_1, \ldots, h'_n, \sigma')$ such that $h_j \neq h'_j$, for some $j \in \{1, \ldots, n\}$ and $h_i = h'_i$ for all $i = 1, \ldots, n$ such that $i \neq j$.

Then we have that

$$g^{\sigma} = R_1 \cdot \ldots \cdot R_n \cdot y_1^{h_1} \cdot \ldots \cdot y_j^{h_j} \cdot \ldots \cdot y_n^{h_n}$$

$$g^{\sigma'} = R_1 \cdot \ldots \cdot R_n \cdot y_1^{h'_1} \cdot \ldots \cdot y_j^{h'_j} \cdot \ldots \cdot y_n^{h'_n}$$

Dividing these two equations, we obtain $g^{\sigma-\sigma'} = y_j^{h_j - h'_j}$. Now we look again at the table TAB_*. With probability $1 - \mu$, we have that $c_j = 1$, and therefore $y_j = y^{\alpha_j}$. In this case, we get $g^{\sigma-\sigma'} = y^{\alpha_j(h_j - h'_j)}$, and so we have that

$$y = g^{\frac{\sigma-\sigma'}{\alpha_j(h_j - h'_j)}} \mod p.$$

Therefore, we have found the discrete logarithm of y in base g. The total success probability is

$$\varepsilon' = (1-\mu)\tilde{\varepsilon}' \geq (1-\mu)\frac{\varepsilon_{\mathcal{F}}^2}{65V_{Q,\hat{n}}} \geq (1-\mu)\frac{(\varepsilon/2)^2}{65V_{Q,\hat{n}}} \geq \frac{(1-\mu)\varepsilon^2}{260V_{Q,\hat{n}}} \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q,\hat{n}}}.$$

We have used that $1 - \mu = 1 - (5/6)^{1/Q_e} \geq 1/6Q_e$ (applying Taylor's series methodology to the function $f(x) = 1-(1-x)^{1/Q_e}$ and then fixing $x = 1/6$).

Finally, the total time needed to solve the Discrete Logarithm problem has been $T' \leq 2T_{\mathcal{F}} \leq 2T + 2Q + 4\hat{n}T_{exp}Q_s$.

□

Note that if we restrict $Q_e = 0$, meaning that the adversary is not able to corrupt any user, then we obtain a success probability $\varepsilon' \geq \frac{\varepsilon^2}{147V_{Q,\hat{n}}}$ of solving the Discrete Logarithm problem.

This theorem gives the exact security of the ring signature scheme explained in Section 4.3, i.e. the exact relation between the successful probabilities and the execution times of both attacks against the scheme and the

discrete logarithm problem. A negative point is the presence of the factor $V_{Q,\hat{n}}$ in the reduction coefficients, because this factor is exponential in the number $\hat{n}$ of members of the ring. Therefore, the security result is really practical only for small rings; but this is what actually happens in some of the applications of ring signatures, such as concurrent signatures [24] or signatures with one designated verifier [86].

4.4 Distributed Ring Signature Schemes

Consider the following extension of the concept of ring signature. Suppose that a set of users $\mathcal{U}_s$ want to anonymously sign some message, in such a way that the verifier of the signature will be convinced that at least the members of some set have all agreed in signing this message, but he could not know which set has actually computed the signature, among the sets of a certain family of possible signing sets.

Members of $\mathcal{U}_s$ can freely choose the rest of users and the family of possible signing sets (in an *ad-hoc* way). We denote as $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_d\}$ the family of possible signing sets, such that the actually signing set $\mathcal{U}_s$ must be in $\mathcal{U}$.

The resulting signature will be a ring signature, taking as ring the set $\mathcal{U}$. In this way, the verifier will be convinced that at least *all* the members of some set in $\mathcal{U}$ have cooperated to compute the signature, but he will not have any information about which set in $\mathcal{U}$ is the actual author of the signature.

This extension of ring signature schemes was first considered in [16]. Their specific RSA-based scheme, however, runs only when the ad-hoc families of possible signing sets are necessarily threshold (that is, they contain all the sets with a fixed number of users). Recently more general proposals, which allow the use of different types of keys, have appeared in [94, 2]; but again these schemes are valid only for the threshold case.

In this section we propose a distributed ring signature scheme for general families of possible signing sets. This scheme follows the ideas of the ring signature scheme that has been presented in Section 4.3. In particular, it can be defined as a generic distributed ring signature scheme, so we could use similar security results than those introduced in Section 4.2.1 to prove the unforgeability of this new scheme.

4.4.1 A Proposal for General Families

We will assume that any specific set of users can always have access to an authenticated broadcast channel, while the information in this channel remains

secret to the rest of users. This can be achieved using different cryptographic techniques (for example, broadcast encryption schemes [44]).

The scheme that we propose consists of the following protocols.

Key generation. Let p and q be large primes such that $q|p-1$ and $q \geq 2^k + \hat{d}$, where k is the security parameter of the scheme and $\hat{d}$ is the maximum possible number of subsets in a family $\mathcal{U}$ of signing subsets. Let g be an element of $\mathbb{Z}_p^*$ with order q, and let H be a collision resistant hash function which outputs elements in $\mathbb{Z}_q$.

Each user U_j of the system has as secret key a random element $x_j \in \mathbb{Z}_q^*$ and the matching public key is $y_j = g^{x_j} \bmod p$.

Distributed ring signature generation. To sign a message m, some users choose a family $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_d\}$ of possible signing sets, such that the signing users form one of the sets in $\mathcal{U}$, say $\mathcal{U}_s$. Note that a specific user can be in more than one of the sets in $\mathcal{U}$ (maybe in all of them).

For each of the sets $\mathcal{U}_i \in \mathcal{U}$, we consider the public value

$$Y_i = \prod_{U_j \in \mathcal{U}_i} y_j = g^{\sum_{U_j \in \mathcal{U}_i} x_j} \bmod p.$$

Members of the signing set $\mathcal{U}_s \in \mathcal{U}$ jointly perform the following algorithm:

1. Each user $U_j \in \mathcal{U}_s$ chooses at random $a_{sj} \in \mathbb{Z}_q^*$ and computes $R_{sj} = g^{a_{sj}} \bmod p$. He broadcasts the value R_{sj}.

2. One of the users in $\mathcal{U}_s$, for example U_1, chooses, for all $i = 1, \ldots, d$, $i \neq s$, random values $a_i \in \mathbb{Z}_q^*$, pairwise different, and computes $R_i = g^{a_i} \bmod p$. He broadcasts these values R_i, and therefore all the members of $\mathcal{U}_s$ can compute $h_i = H(\mathcal{U}, m, R_i)$, for all $i = 1, \ldots, d$, $i \neq s$.

3. Members of $\mathcal{U}_s$ compute the value

$$R_s = \left(\prod_{U_j \in \mathcal{U}_s} R_{sj} \right) \left(\prod_{1 \leq i \leq d, i \neq s} Y_i^{-h_i} \right) \bmod p.$$

   If $R_s = 1$ or $R_s = R_i$ for some $i = 1, \ldots, d$, $i \neq s$, they return to step 1. Members of $\mathcal{U}_s$ can then compute $h_s = H(\mathcal{U}, m, R_s)$.

4. User U_1 computes and broadcasts the value $\sigma_1 = a_{s1} + x_1 h_s + \sum_{1 \leq i \leq d, i \neq s} a_i \bmod q$.

5. For $j = 2, \ldots, n_s$, player U_j computes and broadcasts the value $\sigma_j = a_{sj} + x_j h_s + \sigma_{j-1} \bmod q$.

6. Define $\sigma = \sigma_{n_s}$. The resulting signature is $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$.

Verification of a distributed ring signature. The recipient of the signature checks that $h_i = H(\mathcal{U}, m, R_i)$, for all $i = 1, \ldots, d$, and that the following verification equation is satisfied:

$$g^{\sigma} = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_d^{h_d} \mod p.$$

Some Remarks

- This scheme allows to detect any possible misbehaviour of some of the signers in $\mathcal{U}_s$, because the correctness of the values σ_j can be verified by the rest of the signers, by using known information. Namely, for $j = 1$ the equation

$$g^{\sigma_1} = R_{s1}\ y_1^{h_s} \prod_{1 \leq i \leq d, i \neq s} R_i \mod p$$

must be satisfied. For the rest of values of j, the equation that must be checked is

$$g^{\sigma_j} = R_{sj}\ y_j^{h_s} g^{\sigma_{j-1}} \mod p.$$

- We consider the case where the signing users form an ad-hoc family of signing sets. But the scheme runs as well if the family is fixed. In this case the resulting scheme would be in fact a distributed signature scheme (or threshold signature scheme), taking as access structure the closure of the fixed family.

- If we consider a family where all the possible signing sets are individual users, then we recover the ring signature scheme proposed in Section 4.3. If we consider a family formed by a unique set with a unique user, then we recover the individual Schnorr's signature scheme [87].

- The efficiency of the scheme depends on the total number of users and the number of sets in the family. Therefore, it is a good solution for situations where the number of sets is small. If the family of possible signing sets is a threshold one, then the number of sets is very large (it is exactly $\begin{pmatrix} n \\ t \end{pmatrix}$, if n is the total number of users and t is the threshold). For these cases where the number of sets in the family is very large, we strongly recommend other schemes like the ones presented in [16, 94], for the threshold case, or the scheme that we propose in Section 5.4 for more general cases.

4.4.2 Analysis of the Scheme

Now we show that the proposed distributed ring signature scheme satisfies the three required properties for this kind of schemes: correctness, anonymity and unforgeability.

First of all, we define a *generic distributed ring signature scheme* as the one whose resulting signatures, for a message m and a family $\mathcal{U}$ of d possible signing subsets, have the form $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$, where:

1. the random values R_i are taken from a set whose size is greater than 2^k, where k is the security parameter;

2. $R_i \neq R_j$ for all $i \neq j$;

3. $h_i = H(\mathcal{U}, m, R_i)$ for some hash function $H : \{0,1\}^* \rightarrow \{0, 1, \ldots, 2^k - 1\}$;

4. the value σ is fully determined from the rest of elements in the signature.

It is easy to verify that the scheme proposed in the previous section is actually a generic distributed ring signature scheme. For this kind of schemes, we can state a result very similar to that of Theorem 4.1. We skip the proof because it is almost identical to that of Theorem 4.1.

Theorem 4.3. *(The Distributed Ring Forking Lemma). Consider a generic distributed ring signature scheme with security parameter $k \geq 6$. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data and which can ask Q queries to the random oracle that models the hash function of the scheme.*

We assume that $\mathcal{A}$ produces, within time bound T and with probability of success ε, a valid distributed ring signature $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ for a message m and a family $\mathcal{U}$ of d possible signing sets.

Then, within time $T' \leq 2T$, and with probability $\varepsilon' \geq \frac{\varepsilon^2}{65 V_{Q,n}}$, by executing $\mathcal{A}$ with random instantiations of the hash function we can obtain two valid distributed ring signatures $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ and $(\mathcal{U}, m, R_1, \ldots, R_d, h'_1, \ldots, h'_d, \sigma')$ such that $h_j \neq h'_j$, for some $j \in \{1, \ldots, d\}$ and $h_i = h'_i$ for all $i = 1, \ldots, d$ such that $i \neq j$.

This result will be used in the proof of the unforgeability of our scheme, as well as in Chapter 5.

Correctness

A signature $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ computed by following the protocol of our scheme satisfies the verification equation $g^\sigma = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_d^{h_d} \mod p$. In effect:

$$g^\sigma = g^{\sigma_{n_s}} = g^{\left(\sum_{U_j \in \mathcal{U}_s} a_{sj} + x_j h_s\right) + \left(\sum_{1 \le i \le d, i \ne s} a_i\right)} = \left(\prod_{U_j \in \mathcal{U}_s} R_{sj}\ y_j^{h_s}\right) \left(\prod_{1 \le i \le d, i \ne s} R_i\right) =$$

$$= R_s \left(\prod_{1 \le i \le d, i \ne s} Y_i^{h_i}\right) Y_s^{h_s} \prod_{1 \le i \le d, i \ne s} R_i = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_d^{h_d} \mod p\ .$$

Anonymity

Since we are assuming that signing users have a private broadcast channel, the only information obtained by an external verifier is the ring signature itself. This signature can be seen as a ring signature produced by the (standard) ring scheme proposed in Section 4.3, where the members of the ring are now the sets of users $\mathcal{U}_i$ in the family $\mathcal{U}$, with public keys $Y_i = \prod_{U_j \in \mathcal{U}_i} y_j \mod p$.

Therefore, the unconditional anonymity of the new distributed ring scheme directly infers from the anonymity property of the original ring scheme (see Section 4.3.1 for the proof of this property). Roughly speaking, the verifier has no information about which set is the actual author of a given signature, because all the sets have the same probability of having computed it.

Unforgeability

In order to show that the new scheme is unforgeable, we must first define what unforgeability means in this kind of schemes, namely which are the capabilities and goals of an adversary who tries to successfully attack a distributed ring signature scheme.

The adversary is given a list of $\hat{n}$ identities (or public keys), corresponding to the total set of considered users. He is allowed to corrupt a set of up to Q_e users, obtaining all the secret information owned by these users during the life of the system. The adversary can also make Q queries to the random oracle. Finally, the adversary can require the execution of the signing algorithm for Q_s messages and families that he adaptively chooses; the adversary obtains a valid ring signature, as well as all the information

(secret and public) seen by the corrupted users during the computation of this signature.

We will assume that the adversary only requires executions of the signing algorithm for families where all the sets contain at least one non-corrupted user. Otherwise, if some of the sets was formed only by corrupted users, the adversary would have enough information to compute by himself a valid ring signature for this family.

Such an adversary is $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-successful if he obtains in polynomial time T and with non-negligible probability greater than ε a valid ring signature for some message m and some family $\mathcal{U}$, such that:

(i) the pair formed by message m and family $\mathcal{U}$ has not been asked to the signing oracle during the attack; and

(ii) all the sets of the family $\mathcal{U}$ contain at least one non-corrupted user.

Finally, we say that a distributed ring signature scheme is $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-unforgeable if there does not exist any $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-successful adversary against it.

Theorem 4.4. *Let $\mathcal{A}$ be a $(T, \varepsilon, Q, Q_e, Q_s, \hat{n})$-successful adversary against the distributed ring signature scheme proposed in Section 4.4.1, with security parameter $k \geq 6$ such that $Q_s \leq \frac{2^{k/2}}{4}$ and $Q \leq \frac{2^{k/2}}{3}$.*

We denote by $\hat{d}$ the maximum number of subsets that can form the families for which $\mathcal{A}$ asks for a valid signature.

Then the Discrete Logarithm problem in $\langle g \rangle$ can be solved in time $T' \leq 2T + 2Q + 4(\hat{d} + \hat{n})T_{exp}Q_s$ and with probability $\varepsilon' \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q,\hat{d}}}$.

Proof. Let (p, q, g, Y) be an instance of the Discrete Logarithm problem in the subgroup $< g >$ of $\mathbb{Z}_p$ of order q, where q is a prime that divides $p-1$. The goal is to find the only integer $x \in \{0, 1, \ldots, q-1\}$ such that $g^x = Y \bmod p$.

We will design a probabilistic polynomial time Turing machine $\mathcal{F}$ which will use the adversary $\mathcal{A}$ as a sub-routine, in order to solve the given instance of the Discrete Logarithm problem. Therefore, $\mathcal{F}$ must perfectly simulate the environment of the machine $\mathcal{A}$, answering all the queries that $\mathcal{A}$ makes. Later, we will apply Theorem 4.3 to the machine $\mathcal{F}$, so this machine will be allowed to make Q queries to the random oracle which models the hash function H.

Let us define $\mu = (5/6)^{1/Q_e}$. If we consider attacks where $Q_e = 0$, then we fix $\mu = 0$. Applying a reasoning using Taylor's series, we can bound $1 - \mu \geq \frac{1}{6Q_e}$.

We initialize the machine $\mathcal{F}$, providing it with the public data (p, q, g, Y). The machine $\mathcal{F}$ constructs a table TAB_*, with one entry for any possible user U_j of the scheme. The machine fills each entry of this table as follows: with probability μ, it chooses the bit $c_j = 0$ and an element $x_j \in \mathbb{Z}_q^*$ at random and defines $y_j = g^{x_j} \bmod p$. The entry (U_j, x_j, y_j, c_j) is stored in the table TAB_*. On the other hand, with probability $1 - \mu$, it chooses the bit $c_j = 1$ and an element $\alpha_j \in \mathbb{Z}_q^*$ at random and defines $y_j = Y^{\alpha_j} \bmod p$. A new entry, with the values $(U_j, \alpha_j, y_j, c_j)$, is stored in the table TAB_*.

The list with the computed public keys of the $\hat{n}$ considered users is provided to the attacker $\mathcal{A}$. For any possible set of users $\mathcal{U}_i$, we consider the value $Y_i = \prod_{U_j \in \mathcal{U}_i} y_j \mod p$. Because of the way in which the machine $\mathcal{F}$ has computed the values y_j, we have that

$$Y_i = g^{\gamma_i}\ Y^{\delta_i} \bmod p$$

for some values $\gamma_i, \delta_i \in \mathbb{Z}_q$ that the machine $\mathcal{F}$ knows.

Every time $\mathcal{A}$ asks for the secret key of a user U_j, the machine $\mathcal{F}$ looks for U_j in the table TAB_*. If $c_j = 0$, then $\mathcal{F}$ sends x_i to $\mathcal{A}$. If $c_j = 1$, the machine $\mathcal{F}$ cannot answer and halts. Note that the probability that $\mathcal{F}$ halts in this process is less than $1 - \mu^{Q_e} \leq \frac{1}{6}$.

To answer the queries of $\mathcal{A}$ to the random oracle, since $\mathcal{F}$ has access to the random oracle which models the hash function H, it must only send to $\mathcal{A}$ the answers that it obtains from its oracle.

Now we show how $\mathcal{F}$ can simulate the information that $\mathcal{A}$ obtains from an execution of the signing algorithm. Let B be the set of the users for whom $\mathcal{A}$ has asked for their secret keys (we call them corrupted users). Suppose that $\mathcal{A}$ asks for a valid signature for a message m' and a family $\mathcal{U}' = \{\mathcal{U}'_1, \ldots, \mathcal{U}'_d\}$ of possible signing sets. The machine $\mathcal{F}$ chooses at random one of the sets of $\mathcal{U}'$; for simplicity, we denote this set as $\mathcal{U}'_s = \{U'_1, U'_2, \ldots, U'_{n_s}\}$, which will be the "real" author of the ring signature. The information that $\mathcal{A}$ would obtain from such a real computation consists of all the information broadcast in the private broadcast channel of $\mathcal{U}'_s$ (because we can assume that some of the users in $\mathcal{U}'_s$ is corrupted, and so $\mathcal{A}$ has access to this channel), as well as the secret information generated by the corrupted players in $B \cap \mathcal{U}'_s$. The following algorithm must be executed by the machine $\mathcal{F}$ to obtain this simulated information:

1. For each user $U'_\ell \in \mathcal{U}'_s \cap B$, choose at random $a_{s\ell} \in \mathbb{Z}_q^*$ and compute $R'_{s\ell} = g^{a_{s\ell}} \bmod p$.

2. Choose, for all $i = 1, \ldots, d$, $i \neq s$, random values $a_i \in \mathbb{Z}_q^*$, pairwise different, and compute $R'_i = g^{a_i} \bmod p$ and $h'_i = H(\mathcal{U}', m', R'_i)$ (by asking to the random oracle for H).

3. Choose at random $h'_s \in \mathbb{Z}_q$.

4. For user U'_1:

   - if $U'_1 \in B$ (since $\mathcal{F}$ has not halted, this means that the machine $\mathcal{F}$ knows the secret key x_1 of this corrupted user, as well as the value a_{s1}), compute $\sigma_1 = a_{s1} + x_1 h'_s + \sum_{1 \le i \le d, i \ne s} a_i \mod q$;
   - if $U'_1 \notin B$, choose at random $\sigma_1 \in \mathbb{Z}_q$ and compute

$$R'_{s1} = g^{\sigma_1} y_1^{-h'_s} \prod_{1 \le i \le d, i \ne s} (R'_i)^{-1} \bmod p.$$

5. For user U'_j, for $j = 2, \ldots, n_s$:

   - if $U'_j \in B$ (since $\mathcal{F}$ has not halted, this means that the machine $\mathcal{F}$ knows the secret key x_j of this corrupted user, as well as the value a_{sj}), compute $\sigma_j = a_{sj} + x_j h'_s + \sigma_{j-1} \bmod q$;
   - if $U'_j \notin B$, choose at random $\sigma_j \in \mathbb{Z}_q$ and compute

$$R'_{sj} = g^{\sigma_j - \sigma_{j-1}} y_j^{-h'_s} \bmod p.$$

6. Compute the value

$$R'_s = \left(\prod_{U'_j \in \mathcal{U}'_s} R'_{sj} \right) \left(\prod_{1 \le i \le d, i \ne s} Y_i^{-h'_i} \right) \bmod p.$$

   If $R'_s = 1$ or $R'_s = R'_i$ for some $i = 1, \ldots, d$, $i \ne s$, then return to step 1.

7. Now $\mathcal{F}$ must "falsify" the random oracle H, by imposing the relation $H(\mathcal{U}', m', R'_s) = h'_s$. Later, if $\mathcal{A}$ asks the random oracle H for this input, then $\mathcal{F}$ will answer with h'_s. Since h'_s is a random value, this relation seems consistent to $\mathcal{A}$, provided we assume that H behaves as a random oracle.

8. Define $\sigma' = \sigma_{n_s}$ and return the tuple $(\mathcal{U}', m', R'_1, \ldots, R'_d, h'_1, \ldots, h'_d, \sigma')$.

This simulation produces values which are indistinguishable from those $\mathcal{A}$ would obtain in a real execution of the signing protocol.

In each simulation, $\mathcal{F}$ must perform $2(d+n_s)$ modular exponentiations. There is some risk of "collisions" in the management of the random oracle,

because of the falsification performed by $\mathcal{F}$ in step 7. Such a collision happens if the query $(\mathcal{U}', m', R'_s)$ has been previously made to the random oracle, or if the same tuple $(\mathcal{U}', m', R'_s)$ is produced two times in two different runs of the signature simulation algorithm.

Since no R'_i can appear with probability greater than $1/2^{k-1}$ in a simulated ring signature, we have that:

- The probability that a tuple $(\mathcal{U}', m', R'_s)$ that $\mathcal{F}$ outputs, as part of a simulated ring signature, has been asked before to the random oracle by $\mathcal{A}$ is less than $Q \cdot Q_s \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$.
- The probability that the same tuple $(\mathcal{U}', m', R'_s)$ is output two times by $\mathcal{F}$ in two different signature simulations is less than $\frac{Q_s^2}{2} \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$ (using again Fact 4.1).

Altogether, the probability of collisions is less than $1/3$. Now we can compute the success probability of the machine $\mathcal{F}$; that is, the probability that $\mathcal{F}$ obtains a valid ring signature:

$$\varepsilon_{\mathcal{F}} = \Pr[\mathcal{F} \text{ succeeds}] =$$

$$\Pr[\mathcal{F} \text{ does not halt AND no-collisions in the simulations AND } \mathcal{A} \text{ succeeds}] \geq$$

$$\geq \Pr[\mathcal{A} \text{ succeeds} \mid \mathcal{F} \text{ does not halt AND no-collisions in the simulations }] \cdot$$

$$\cdot (1 - \Pr[\mathcal{F} \text{ halts OR collisions in the simulations}]) \geq \varepsilon \left(1 - \frac{1}{6} - \frac{1}{3}\right) = \frac{\varepsilon}{2}.$$

The execution time of the machine $\mathcal{F}$ is $T_{\mathcal{F}} \leq T + Q + 2(\hat{d} + \hat{n})T_{exp}Q_s$. Summing up, we have constructed a Turing machine $\mathcal{F}$ that forges a generic distributed ring signature scheme in time $T_{\mathcal{F}}$ and with probability $\varepsilon_{\mathcal{F}} \geq \frac{\varepsilon}{2}$. We now apply Theorem 4.3 to the machine $\mathcal{F}$.

The theorem says that, by executing twice the machine $\mathcal{F}$, we will obtain in time $T' \leq 2T_{\mathcal{F}}$ and with probability $\tilde{\varepsilon}' \geq \frac{\varepsilon_{\mathcal{F}}^2}{65 V_{Q,\hat{d}}}$ two valid ring signatures $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ and $(\mathcal{U}, m, R_1, \ldots, R_d, h'_1, \ldots, h'_d, \sigma')$ such that $h_j \neq h'_j$, for some $j \in \{1, \ldots, d\}$ and $h_i = h'_i$ for all $i = 1, \ldots, d$ such that $i \neq j$.

Furthermore, since the ring signatures have been correctly forged by $\mathcal{A}$, this means that there exists at least one non-corrupted user in each subset $\mathcal{U}_i \in \mathcal{U}$; in particular there exists a non-corrupted user $U_z \in \mathcal{U}_j - B$ in the subset $\mathcal{U}_j$. Note that $Y_j = g^{\gamma_j}\ Y^{\delta_j} \bmod p$. With probability $1 - \mu$, we

have $c_z = 1$ and therefore $y_z = Y^{\alpha_z} \bmod p$. So the value α_z is one of the terms added in the exponent δ_j. If this is the case, then with overwhelming probability we will have that $\delta_j \neq 0 \bmod q$.

Let us consider then the two corresponding verification equations, satisfied by the two forged ring signatures:

$$g^{\sigma} = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_j^{h_j} \cdot \ldots \cdot Y_d^{h_d} \mod p$$

$$g^{\sigma'} = R_1 \cdot \ldots \cdot R_d \cdot Y_1^{h_1} \cdot \ldots \cdot Y_j^{h'_j} \cdot \ldots \cdot Y_d^{h_d} \mod p.$$

Dividing these two equations, we obtain the equality $g^{\sigma-\sigma'} = Y_j^{h_j - h'_j} = g^{\gamma_j(h_j-h'_j)} Y^{\delta_j(h_j-h'_j)} \bmod p$. Therefore, we have found the discrete logarithm x of Y with respect to the base g, which is:

$$x = \frac{\sigma - \sigma' - \gamma_j(h_j - h'_j)}{\delta_j(h_j - h'_j)}.$$

The inverse of $\delta_j(h_j - h'_j)$ is taken modulo q, and it exists because $h_j \neq h'_j$ and $\delta_j \neq 0 \bmod q$ with overwhelming probability.

We have thus constructed a probabilistic algorithm that solves the Discrete Logarithm problem. Its running time is $T' \leq 2T_{\mathcal{F}} \leq 2T + 2Q + 4(\hat{d} + \hat{n})T_{exp}Q_s$. On the other hand, its success probability is

$$\varepsilon' \geq (1-\mu)\tilde{\varepsilon}' \geq \frac{(1-\mu)\varepsilon_{\mathcal{F}}^2}{65V_{Q,\hat{d}}} \geq (1-\mu)\frac{(\varepsilon/2)^2}{65V_{Q,\hat{d}}} \geq \frac{(1-\mu)\varepsilon^2}{260V_{Q,\hat{d}}} \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q,\hat{d}}}.$$

□

Again, if we do not allow the adversary to corrupt any user, which means $Q_e = 0$, then the obtained success probability of solving the Discrete Logarithm problem would be $\varepsilon' \geq \frac{\varepsilon^2}{147V_{Q,\hat{d}}}$.

Chapter 5

New Ring Signature Schemes for Identity-Based Scenarios

The cryptographic protocols that we have studied until now work in a standard public key scenario: each user U has a secret key SK_U, and usually the matching public key PK_U is computed from SK_U. This happens in RSA and Schnorr's signature schemes, and also in the ring signature schemes that we have proposed in Chapter 4. In these scenarios, a serious problem appears: how can one be sure that PK_U is actually the public key of user U, or in other words, that the only person who knows SK_U is user U? For example, a different user U' can generate $SK_{U'}$, compute the matching public key and broadcast it as if it was the public key of user U.

To solve this problem, the public keys of the users are authenticated via a Public Key Infrastructure (PKI) based on digital certificates: a user who wants to use a public key cryptosystem turns to a certification authority, who signs a message linking the public key PK_U with the identity of user U. Later, a user who must use public key PK_U (to encrypt a message or to verify a signature) must first verify that the certificate which links U and PK_U is still valid. Other problems appear when revocation of some certificate is necessary, because a secret key SK_U corresponding to a certified PK_U has been compromised, for example.

All these facts make the use of cryptographic protocols less efficient in the real life. A dramatic example are ring signature schemes in PKI scenarios, such as the ones proposed in Chapter 4: the anonymous signer must verify that all the public keys that he includes in the ring signatures are still certified, and the verifier of the signature must do the same. If the number of involved users is very large, this can result in very slow protocols, in practice.

Thus, any possible alternative which avoids the necessity of digital cer-

tificates is welcome in order to design efficient ring signature schemes, and in general more efficient public key cryptosystems.

One of these alternatives is identity-based cryptography. In this chapter we propose a standard ring signature scheme and two distributed ring signature schemes which work in identity-based scenarios.

5.1 Identity-Based Cryptography from Bilinear Pairings

Shamir introduced in 1984 the concept of *identity-based* cryptography [89] (from now on, ID-based cryptography). The idea is that the public key of a user can be publicly computed from his identity (for example, from a complete name, an e-mail or an IP address). Then, the secret key is derived from the public key. In this way, digital certificates are not needed, because anyone can easily verify that some public key PK_U corresponds in fact to user U.

The process that generates secret keys from public keys must be executed by an external entity, known as the *master entity*. This entity knows the secret keys of all the users of the system, so it must be completely trusted. A way to relax this negative point could be to consider a set of master entities which share the secret information.

Most of the proposed cryptographic protocols for ID-based scenarios, specially since the proposal of [13], use as a tool a mathematical object known as *bilinear pairing*. The protocols that we will propose throughout this chapter also use this concept, so we explain it in this section.

Let $\mathbb{G}_1$ be an additive group of prime order q, generated by some element P. Let $\mathbb{G}_2$ be a multiplicative group with the same order q.

Definition 5.1. *A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following three properties:*

1. *It is bilinear, which means that given elements $A_1, A_2, A_3 \in \mathbb{G}_1$, we have that $e(A_1 + A_2, A_3) = e(A_1, A_3) \cdot e(A_2, A_3)$ and $e(A_1, A_2 + A_3) = e(A_1, A_2) \cdot e(A_1, A_3)$.*

2. *The map e can be efficiently computed for any possible input pair. We use the notation T_{bp} for a bound on the necessary time to evaluate a bilinear pairing.*

3. *The map e is non-degenerate: there exist elements $A_1, A_2 \in \mathbb{G}_1$ such that $e(A_1, A_2) \neq 1_{\mathbb{G}_2}$.*

In particular, property 1 implies that $e(aP, bP) = e(P, P)^{ab} = e(P, abP) = e(abP, P)$, for all $a, b \in \mathbb{Z}_q$. This implies $e(A_1, A_2) = e(A_2, A_1)$, for all $A_1, A_2 \in \mathbb{G}_1$, and also implies $e(0, A) = 1_{\mathbb{G}_2}$, for all $A \in \mathbb{G}_1$.

Combining properties 1 and 3, it is easy to see that $e(P, P) \neq 1_{\mathbb{G}_2}$ and that the equality $e(A_1, P) = e(A_2, P)$ implies that $A_1 = A_2$.

The typical way of obtaining such pairings is by deriving them from the Weil or the Tate pairing on an elliptic curve over a finite field. The interested reader is referred to [96] for a complete bibliography of cryptographic works based on pairings.

Let $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ be a public hash function, where $\mathbb{G}_1^* = \mathbb{G}_1 - \{0\}$. The most usual way to design an ID-based cryptosystem is the following. The master has a secret key $x \in \mathbb{Z}_q^*$, and he publishes the value $Y = xP \in \mathbb{G}_1$.

Every user U of the ID-based system has an identifier $ID_U \in \{0,1\}^*$, that can be an IP address, a telephone number, an e-mail address, etc. The public key of U is then defined to be $PK_U = H_1(ID_U) \in \mathbb{G}_1^*$. In this way, everybody can verify the authenticity of a public key without the necessity of certificates. The user U needs to contact the master entity to obtain his secret key $SK_U = xPK_U \in \mathbb{G}_1$.

For simplicity, we consider throughout this chapter bilinear pairings as defined above. But the constructions can be extended to the case where the pairings are $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, for three different groups $\mathbb{G}_1$ (additive), $\mathbb{G}_2 = \langle P_2 \rangle$ (additive) and $\mathbb{G}_3$ (multiplicative), all with prime order q. Such pairings are called *asymmetric pairings* in [**?**]. In this case, the security of the protocols can be related to the following problem: given $P_2, aP_2 \in \mathbb{G}_2$ and $Q \in \mathbb{G}_1$, compute aQ, where a is chosen at random in $\mathbb{Z}_q$ and Q is chosen at random in $\mathbb{G}_1$. If we want to relate the difficulty of this problem to the difficulty of standard problems in $\mathbb{G}_1$ (such as the Discrete Logarithm problem or the Computational Diffie-Hellman problem), then we need a computable isomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$.

5.2 A New ID-Based Ring Signature Scheme

The only ID-based ring signature scheme proposed until now (as far as we know) is the one by Zhang and Kim [95]. A formal analysis of the unforgeability of this scheme can be found in [57], where it is reduced to the Computational Diffie-Hellman problem. In this scheme, the generation and verification of a ring signature must be performed in an iterative way: the signer and the verifier must compute a pairing for each member U_i of the ring, and the value corresponding to U_i is necessary to compute the value of U_{i+1}.

We propose in this section a different ID-based ring signature scheme, which is also based on bilinear pairings. In the new scheme the computations in the generation and verification of a signature can be performed in parallel, more efficiently than in the scheme of Zhang and Kim [95]. Furthermore, the number of evaluations of the bilinear pairing that must be performed in an execution (signature and verification) of our scheme is $n + 3$, if n is the number of members of a ring, whereas each execution of the scheme in [95] requires $4n - 1$ pairing evaluations.

On the other hand, our ID-based ring signature scheme is a generic one, as defined in Section 4.2. We could thus apply the Ring Forking Lemma (Theorem 4.1) to analyze the unforgeability of the scheme.

Here we explain the different protocols that form the new scheme.

Setup. Let $\mathbb{G}_1$ be an additive group of prime order q, generated by some element P. Let $\mathbb{G}_2$ be a multiplicative group with the same order q. We need $q \geq 2^k + \hat{n}$, where k is the security parameter of the scheme and $\hat{n}$ is the maximum possible number of users in a ring. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear pairing as defined in Section 5.1. Let $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ be two hash functions (in the proof of security, we will assume that they behave as random oracles). All this information is publicly available.

The master entity chooses at random his secret key $x \in \mathbb{Z}_q^*$ and publishes the value $Y = xP$.

Secret key extraction. A user U, with identity ID_U, has public key $PK_U = H_1(ID_U)$. When he requests the master for his matching secret key, he obtains the value $SK_U = xPK_U$.

Ring signature generation. Consider a ring $\mathcal{U} = \{U_1, \ldots, U_n\}$ of users; for simplicity we denote $PK_i = PK_{U_i} = H_1(ID_{U_i})$. If some of these users U_s, where $s \in \{1, \ldots, n\}$, wants to anonymously sign a message m on behalf of the ring $\mathcal{U}$, he acts as follows:

1. For all $i \in \{1, \ldots, n\}$, $i \neq s$, choose A_i uniformly at random in $\mathbb{G}_1^*$, pairwise different (for example, by choosing $a_i \in \mathbb{Z}_q^*$ at random and considering $A_i = a_i P$). Compute $R_i = e(A_i, P) \in \mathbb{G}_2$ and $h_i = H_2(\mathcal{U}, m, R_i)$, for all $i \neq s$.

2. Choose at random $A \in \mathbb{G}_1$.

3. Compute $R_s = e(A, P) \cdot e(-Y, \sum_{i \neq s} h_i PK_i)$. If $R_s = 1_{\mathbb{G}_2}$ or $R_s = R_i$ for some $i \neq s$, then go to step 2.

4. Compute $h_s = H_2(\mathcal{U}, m, R_s)$.

5. Compute $\sigma = h_s SK_s + A + \sum_{i \neq s} A_i$.

6. Define the signature of the message m made by the ring $\mathcal{U} = \{U_1, \ldots, U_n\}$ to be $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$.

In fact, the values h_i can be publicly computed from the ring $\mathcal{U}$, the message m and the values R_i. We include them in the signature for clarity in the treatment of the security of the scheme.

Verification of a ring signature. The validity of the signature is verified by the recipient of the message by checking that $h_i = H_2(\mathcal{U}, m, R_i)$ and that

$$e(\sigma, P) = R_1 \cdot \ldots \cdot R_n \cdot e(Y, \sum_{i=1}^{n} h_i PK_i) \ .$$

5.2.1 Analysis of the Scheme

In this section we prove that the proposed ID-based ring signature scheme satisfies the properties of correctness, anonymity and unforgeability.

Correctness

The property of correctness is satisfied. In effect, if the ring signature has been correctly generated, then:

$$R_1 \cdot \ldots \cdot R_n \cdot e(Y, \sum_{i=1}^{n} h_i PK_i) = e(A + \sum_{i \neq s} A_i, P) \cdot e(-Y, \sum_{i \neq s} h_i PK_i) \cdot e(Y, \sum_{i=1}^{n} h_i PK_i) =$$

$$= e(A + \sum_{i \neq s} A_i, P) \cdot e(Y, h_s PK_s) = e(A + \sum_{i \neq s} A_i, P) \cdot e(P, h_s x PK_s) =$$

$$= e(A + \sum_{i \neq s} A_i + h_s SK_s, P) = e(\sigma, P) \ .$$

Anonymity

The unconditional anonymity of the scheme is also easy to prove. Intuitively, the scheme is completely symmetric. Given a valid ring signature on behalf of a ring $\mathcal{U}$, the probability that a specific user in $\mathcal{U}$ is the actual author of

this signature is the same for all the members of $\mathcal{U}$. Specifically, if the ring has n members, this probability is

$$\frac{1}{q-1} \cdot \frac{1}{q-2} \cdot \ldots \cdot \frac{1}{q-n+1} \cdot \frac{1}{q-n}$$

which does not depend on the considered member of $\mathcal{U}$.

So we can conclude that any attacker outside a ring of n possible users has probability $1/n$ to guess which member of the ring has actually computed a given signature on behalf of this ring.

Unforgeability

Remember (see Section 4.3.1) that a $(T, \varepsilon, Q_1, Q_2, Q_e, Q_s, \hat{n})$-successful attacker against a ring signature scheme is an algorithm which takes as input a list of $\hat{n}$ identities, corresponding to the total set of considered users. This attacker runs in time T, makes Q_1 queries to the random oracle H_1, Q_2 queries to the random oracle H_2, corrupts Q_e users, obtaining their secret keys and asks for Q_s valid ring signatures. With probability greater than ε, this algorithm obtains a valid signature for a new pair (ring,message), and such that no secret key of any of the members of the forged ring has been queried during the attack.

Note also that our ID-based scheme is a suitable generic ring signature scheme, satisfying that $R_i \neq R_j$ for all pair $i \neq j$, that $h_i = H_2(\mathcal{U}, m, R_i)$ and that any randomness value $R_i \in \mathbb{G}_1^*$ appears in a ring signature with probability less than $\frac{1}{q-\hat{n}} \leq \frac{2}{2^k}$, as required. Therefore, we could use the Ring Forking Lemma (Theorem 4.1).

The following theorem states that our ID-based ring signature scheme is secure in the random oracle model, assuming that the Computational Diffie-Hellman problem is hard to solve.

Theorem 5.1. *Let $\mathcal{A}$ be a $(T, \varepsilon, Q_1, Q_2, Q_e, Q_s, \hat{n})$-successful attacker against the ring signature scheme presented in Section 5.2, with security parameter $k \geq 6$ and such that $Q_s \leq \frac{2^{k/2}}{4}$ and $Q \leq \frac{2^{k/2}}{3}$.*

Then the Computational Diffie-Hellman problem in $\mathbb{G}_1$ can be solved within time $T' \leq 2T + 2Q_1 + 2Q_2 + 2\hat{n}T_{bp}Q_s$ and with probability $\varepsilon' \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q_2,\hat{n}}}$.

Proof. Let (P, aP, bP) be the input of an instance of the Computational Diffie-Hellman problem in $\mathbb{G}_1$. Here P is a generator of $\mathbb{G}_1$, with prime order q, and the elements a, b are taken uniformly at random in $\mathbb{Z}_q^*$.

We are going to construct a probabilistic polynomial time Turing machine $\mathcal{F}$ to which we will apply the result of Theorem 4.1 with hash function

H_2; therefore, $\mathcal{F}$ can make Q_2 queries to the random oracle which models the behaviour of H_2. The goal of this machine $\mathcal{F}$ is to solve the given instance of the Computational Diffie-Hellman problem. It will use the attacker $\mathcal{A}$ as a sub-routine; therefore, $\mathcal{F}$ must perfectly simulate the environment of the attacker $\mathcal{A}$. Figure 5.1 below tries to capture this idea.

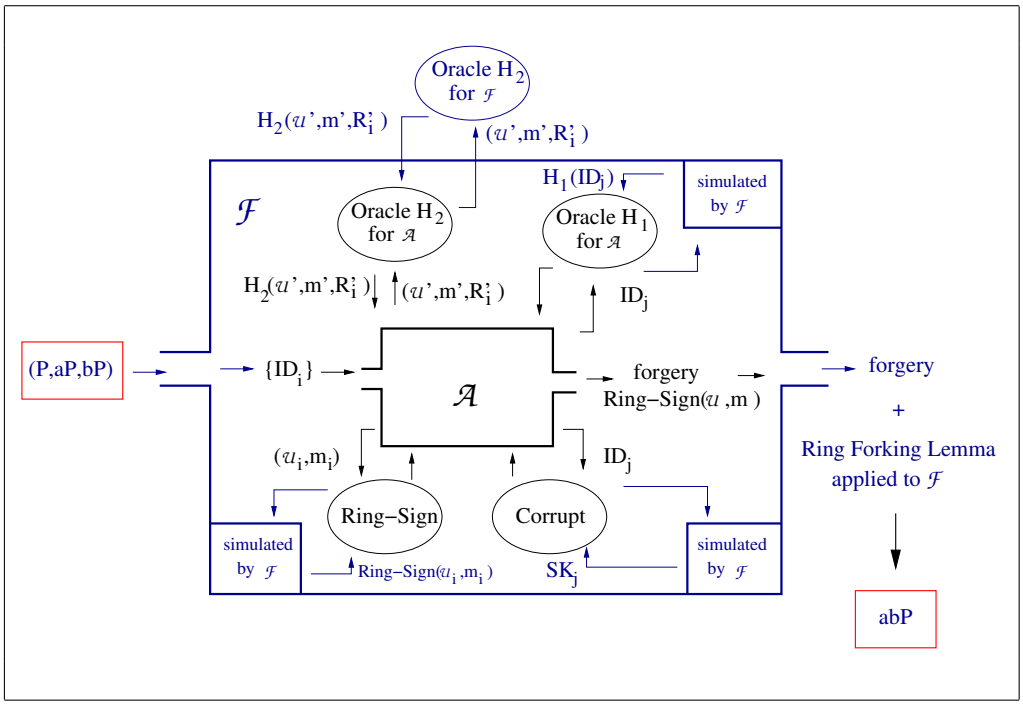

Figure 5.1: $\mathcal{F}$ executes $\mathcal{A}$ and solves the CDH problem

The machine $\mathcal{F}$ receives the public data (P, aP, bP). Its goal is to compute the value abP. The machine $\mathcal{F}$ runs the attacker $\mathcal{A}$ against the ID-based ring signature scheme, answering to all the queries that $\mathcal{A}$ makes. The public key of the master entity is defined to be $Y = aP$, and sent to the attacker $\mathcal{A}$, along with the list containing the $\hat{n}$ identities of the considered users.

Without loss of generality, we can assume that $\mathcal{A}$ asks the random oracle H_1 for the value $H_1(ID)$ before asking for the secret key of ID.

Let us define $\mu = (5/6)^{1/Q_e}$, which satisfies that $1 - \mu \geq \frac{1}{6Q_e}$. If we consider attacks where $Q_e = 0$, then we define $\mu = 0$.

The machine $\mathcal{F}$ constructs a table TAB_{H_1} to simulate the random oracle H_1. Every time an identity ID_i is asked by $\mathcal{A}$ to the oracle H_1, the machine $\mathcal{F}$ acts as follows: first $\mathcal{F}$ checks if this input is already in the table; if this is the case, then $\mathcal{F}$ sends to $\mathcal{A}$ the corresponding relation $H_1(ID_i) = PK_i$.

Otherwise, with probability μ, the machine $\mathcal{F}$ chooses the bit $c_i = 0$ and a different $x_i \in \mathbb{Z}_q^*$ at random and defines $PK_i = x_i P$ and $SK_i = x_i Y$. The entry $(ID_i, PK_i, x_i, SK_i, c_i)$ is stored in the table TAB_{H_1}. On the other hand, with probability $1 - \mu$, the machine $\mathcal{F}$ chooses the bit $c_i = 1$ and a different $\alpha_i \in \mathbb{Z}_q^*$ at random and defines $PK_i = (\alpha_i)bP$ and $SK_i = \perp$. The values $(ID_i, PK_i, \alpha_i, c_i)$ are stored in a new entry of TAB_{H_1}. The relation $H_1(ID_i) = PK_i$ is sent to $\mathcal{A}$. The condition $PK_i \neq PK_j$ must be satisfied for all the different entries $i \neq j$ of the table; if this is not the case, the process is repeated for one of these users.

Since we are assuming that H_1 behaves as a random function, and the values PK_i are all randomly chosen, the information that $\mathcal{A}$ receives in this step is consistent.

Later, every time $\mathcal{A}$ asks for the secret key corresponding to an identity ID_i, the machine $\mathcal{F}$ looks for ID_i in the table TAB_{H_1}. If $c_i = 0$, then $\mathcal{F}$ sends $SK_i = x_i Y$ to $\mathcal{A}$. If $c_i = 1$, the machine $\mathcal{F}$ cannot answer and halts. Note that the probability that $\mathcal{F}$ halts in this process is less than $1 - \mu^{Q_e} \leq \frac{1}{6}$.

On the other hand, when $\mathcal{A}$ makes a query to the random oracle H_2, the machine $\mathcal{F}$ can ask its own oracle for this hash function, and then return to $\mathcal{A}$ the obtained answer.

Finally, the attacker $\mathcal{A}$ can ask Q_s times for a valid ring signature for messages m' and rings $\mathcal{U}'$ of $n \leq \hat{n}$ members (for simplicity, we denote $\mathcal{U}' = \{U'_1, \ldots, U'_n\}$). We assume that $\mathcal{A}$ has not asked for any of the secret keys of the ring $\mathcal{U}'$ because, otherwise, $\mathcal{A}$ could obtain a valid ring signature by itself. To answer such a query, the machine $\mathcal{F}$ proceeds as follows:

1. Choose at random an index $s \in \{1, \ldots, n\}$.
2. For all $i \in \{1, \ldots, n\}$, $i \neq s$, choose A_i at random in $\mathbb{G}_1^*$, pairwise different. Compute $R'_i = e(A_i, P)$, for all $i \neq s$.
3. For $i \neq s$, compute $h'_i = H_2(\mathcal{U}', m', R'_i)$ (by asking to the random oracle for H_2).
4. Choose at random $h'_s \in \mathbb{Z}_q$.
5. Choose at random $\sigma' \in \mathbb{G}_1$.
6. Compute $R'_s = e(\sigma' - \sum_{i \neq s} A_i, P) \cdot e(-Y, \sum_{i=1}^{n} h'_i PK_i)$. If $R'_s = 1$ or $R'_s = R'_i$ for some $i \neq s$, then go to step 5.
7. At this point, the machine $\mathcal{F}$ must "falsify" the random oracle H_2, by imposing the relation $H_2(\mathcal{U}', m', R'_s) = h'_s$. Later, if $\mathcal{A}$ asks to the random oracle H_2 for this input, then $\mathcal{F}$ will answer with h'_s.

8. Return the tuple $(\mathcal{U}', m', R'_1, \ldots, R'_n, h'_1, \ldots, h'_n, \sigma')$.

In such a simulation, the machine $\mathcal{F}$ must perform $n+1$ pairing evaluations, which is the most costly operation (recall that each pairing evaluation costs time T_{bp}).

Since h'_s is a random value and we are in the random oracle model for H_2, the information provided to $\mathcal{A}$ is indistinguishable from what $\mathcal{A}$ would see in a real execution of the signing protocol. However, there is some risk of "collisions" because of the values falsified by $\mathcal{F}$ in step 7 of the simulation above.

Since the considered scheme is a generic ring signature scheme, no R'_i can appear with probability greater than $1/2^{k-1}$ in a ring signature. Two kinds of collisions can occur:

- A tuple $(\mathcal{U}', m', R'_s)$ that $\mathcal{F}$ outputs, inside a simulated ring signature, has been asked before to the random oracle by $\mathcal{A}$. In this case, it is quite unlikely that the relation $H_2(\mathcal{U}', m', R'_s) = h'_i$ corresponding to the values output in the simulation of the signature coincides with the relation previously stored in TAB_{H_2}. The probability of such a collision is, however, less than $Q_2 \cdot Q_s \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$.
- The same tuple $(\mathcal{U}', m', R'_s)$ is output by $\mathcal{F}$ in two different simulated ring signatures. Using the result in Fact 4.1 (Section 4.3.1), we have that the probability of this collision is less than $\frac{Q_s^2}{2} \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$.

Altogether, the probability of collisions is less than $1/3$. Now we can compute the success probability of the machine $\mathcal{F}$; that is, the probability that $\mathcal{F}$ obtains a valid ring signature:

$$\varepsilon_{\mathcal{F}} = \Pr[\mathcal{F} \text{ succeeds}] =$$

$$\Pr[\mathcal{F} \text{ does not halt AND no-collisions in the simulations AND } \mathcal{A} \text{ succeeds}] \geq$$

$$\geq \Pr[\mathcal{A} \text{ succeeds} \mid \mathcal{F} \text{ does not halt AND no-collisions in the simulations }] \cdot$$

$$\cdot (1 - \Pr[\mathcal{F} \text{ halts OR collisions in the simulations}]) \geq \varepsilon \left(1 - \frac{1}{6} + \frac{1}{3}\right) = \frac{\varepsilon}{2}.$$

On the other hand, the execution time of the machine $\mathcal{F}$ is $T_{\mathcal{F}} \leq T + Q_1 + Q_2 + \hat{n} T_{bp} Q_s$. Summing up, we have a Turing machine $\mathcal{F}$ that forges a generic ring signature scheme in time $T_{\mathcal{F}}$ and with probability $\varepsilon_{\mathcal{F}} \geq \frac{\varepsilon}{2}$. Now we apply Theorem 4.1 to the machine $\mathcal{F}$, with respect to the hash function H_2.

This means that, by executing twice the machine $\mathcal{F}$, we will obtain in time $T' \leq 2T_{\mathcal{F}}$ and with probability $\tilde{\varepsilon}' \geq \frac{\varepsilon_{\mathcal{F}}^2}{65V_{Q_2,\hat{n}}}$ two valid ring signatures $(\mathcal{U}, m, R_1, \ldots, R_n, h_1, \ldots, h_n, \sigma)$ and $(\mathcal{U}, m, R_1, \ldots, R_n, h'_1, \ldots, h'_n, \sigma')$ such that $h_j \neq h'_j$, for some $j \in \{1, \ldots, n\}$ and $h_i = h'_i$ for all $i = 1, \ldots, n$ such that $i \neq j$. Then we have that

$$e(\sigma, P) = R_1 \cdot \ldots \cdot R_n \cdot e(Y, h_1 PK_1) \cdot \ldots \cdot e(Y, h_n PK_n)$$

$$e(\sigma', P) = R_1 \cdot \ldots \cdot R_n \cdot e(Y, h'_1 PK_1) \cdot \ldots \cdot e(Y, h'_n PK_n)$$

Dividing these two equations, we obtain $e(\sigma - \sigma', P) = e(Y, (h_j - h'_j)PK_j) = e(aP, (h_j - h'_j)PK_j)$. Now we look again to the table TAB_{H_1}; since the forgeries of $\mathcal{A}$ are valid, then user U_j is not corrupted and so, with probability $1 - \mu$, we have that $c_j = 1$ and so $PK_j = (\alpha_j)bP$.

Then the relation becomes $e(\sigma - \sigma', P) = e(aP, (h_j - h'_j)\alpha_j bP) = e(ab\alpha_j(h_j - h'_j)P, P)$. Since the pairing is non-degenerate, this implies that $\sigma - \sigma' = ab\alpha_j(h_j - h'_j)P$. Therefore, we can compute

$$abP = \frac{1}{\alpha_j(h_j - h'_j)}(\sigma - \sigma') \ .$$

The inverse is computed modulo q, and it always exists because $\alpha_j \in \mathbb{Z}_q^*$ and $h_j \neq h'_j$.

Summing up, the machine $\mathcal{F}$ solves the given instance of the Computational Diffie-Hellman problem with probability

$$\varepsilon' = (1 - \mu)\tilde{\varepsilon}' \geq (1 - \mu)\frac{\varepsilon_{\mathcal{F}}^2}{65V_{Q_2,\hat{n}}} \geq (1 - \mu)\frac{(\varepsilon/2)^2}{65V_{Q_2,\hat{n}}} \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q_2,\hat{n}}}.$$

And the total time needed to solve the problem has been $T' \leq 2T_{\mathcal{F}} \leq 2T + 2Q_1 + 2Q_2 + 2\hat{n}T_{bp}Q_s$.

□

As we mentioned in the schemes for Disc-Log scenarios presented in Chapter 4, if we consider attackers who cannot corrupt any user, then we have $Q_e = 0$ and the security result becomes $\varepsilon' \geq \frac{\varepsilon^2}{147V_{Q_2,\hat{n}}}$.

We must note here that the quality of the reduction is worse than the reduction in the proof of the unforgeability of the scheme of Zhang and Kim [95], provided in [57]. This is due to the presence of the value $V_{Q_2,\hat{n}}$ in the relation between ε' and ε. Therefore, our new ID-based ring signature scheme improves the efficiency of the scheme of Zhang and Kim, but the security reduction that we have found is less tight.

5.3 An ID-Based Distributed Ring Signature Scheme for General Families

We now extend the ID-based ring signature scheme proposed and analyzed above, to the distributed scenario explained in Section 4.4: a set of users who want to jointly sign a message with a certain anonymity. They choose other subsets of users to form a family of possible signing sets, which must include the actually signing set. They use their secret keys and the public keys of the rest of users to compute a ring signature. The recipient of the signature is convinced that all the players of some set in the family of possible signing subsets have cooperated to sign the message, but he does not have any information about which is actually the signing subset.

The following proposal, contrary to the one in Section 4.4.1, works in an identity-based scenario, where the public keys of the users can be directly computed from their identities. Again, we will assume that any specific set of users can always have access to an authenticated and secret broadcast channel: users out of the subset do not obtain any information about the broadcast messages. Our proposal supports general families of possible signing subsets, contrary to other ID-based distributed ring signature schemes (see [25], for example), which work only for threshold families.

The protocols of the scheme work as follows:

Setup. Let $\mathbb{G}_1$ be an additive group of prime order q, generated by some element P. Let $\mathbb{G}_2$ be a multiplicative group with the same order q. We need $q \geq 2^k + \hat{d}$, where k is the security parameter of the scheme and $\hat{d}$ is the maximum possible number of sets in a family of possible signing sets. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear pairing as defined in Section 5.1. Let $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ be two hash functions. All this information is public.

The master entity chooses at random his secret key $x \in \mathbb{Z}_q^*$ and publishes the value $Y = xP$.

Secret key extraction. Any user U_i of the system, with identity ID_i, has public key $PK_i = H_1(ID_i)$. When he requests the master for his matching secret key, he obtains the value $SK_i = xPK_i$.

Distributed ring signature generation. Assume that a set $\mathcal{U}_s$ of users (we denote them as $\mathcal{U}_s = \{U_1, U_2, \ldots, U_{n_s}\}$) want to compute an anonymous signature. They choose *ad-hoc* the family $\mathcal{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_d\}$ of possible signing sets, such that $\mathcal{U}_s \in \mathcal{U}$.

For each of the sets $\mathcal{U}_i \in \mathcal{U}$, we consider the public value

$$Y_i = \sum_{U_j \in \mathcal{U}_i} PK_j \ .$$

The algorithm for computing the ring signature is the following:

1. Each user $U_j \in \mathcal{U}_s$ chooses at random $a_{sj} \in \mathbb{Z}_q^*$ and computes $R_{sj} = e(a_{sj}P, P)$. He broadcasts the value R_{sj}.

2. One of the users in $\mathcal{U}_s$, for example U_1, chooses, for all $i = 1, \ldots, d$, $i \neq s$, random values $a_i \in \mathbb{Z}_q^*$, pairwise different, and computes $R_i = e(a_i P, P)$. He broadcasts these values R_i, and therefore all the members of $\mathcal{U}_s$ can compute $h_i = H_2(\mathcal{U}, m, R_i)$, for all $i = 1, \ldots, d$, $i \neq s$.

3. Members of $\mathcal{U}_s$ compute the value

$$R_s = e(-Y, \sum_{i \neq s} h_i Y_i) \prod_{U_j \in \mathcal{U}_s} R_{sj} \ .$$

If $R_s = 1_{\mathbb{G}_2}$ or $R_s = R_i$ for some $i = 1, \ldots, d$, $i \neq s$, they return to step 1. Members of $\mathcal{U}_s$ can then compute $h_s = H_2(\mathcal{U}, m, R_s)$.

4. User U_1 computes and broadcasts the value $\sigma_1 = a_{s1}P + h_s SK_1 + \sum_{1 \leq i \leq d, i \neq s} a_i P \ \in \mathbb{G}_1$.

5. For $j = 2, \ldots, n_s$, player U_j computes and broadcasts the value $\sigma_j = a_{sj}P + h_s SK_j + \sigma_{j-1} \ \in \mathbb{G}_1$.

6. Define $\sigma = \sigma_{n_s}$. The resulting signature is $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$.

Verification of a distributed ring signature. The validity of the signature is verified by the recipient of the message by checking that $h_i = H_2(\mathcal{U}, m, R_i)$, for $i = 1, \ldots, d$ and that the following equation fulfills:

$$e(\sigma, P) = e(Y, \sum_{i=1}^{d} h_i Y_i) \prod_{1 \leq i \leq d} R_i \ ,$$

where $Y_i = \sum_{U_j \in \mathcal{U}_i} PK_j$, for all the sets $\mathcal{U}_i$ in the family $\mathcal{U}$.

5.3.1 Some Remarks

- As it happens in the distributed ring signature scheme proposed in Section 4.4.1 for Disc-Log scenarios, the ID-based distributed ring signature scheme proposed above allows to detect whether some of the signers in the subset $\mathcal{U}_s$ tries to boycott the process of signing. In effect, the correctness of the values σ_j can be verified by the rest of the signers, by using public information. Namely, for $j = 1$ the equation

$$e(\sigma_1, P) = R_{s1} \cdot e(h_s PK_1, Y) \cdot \prod_{1 \leq i \leq d, i \neq s} R_i$$

must be satisfied. For the rest of users $U_j \in \mathcal{U}_s$, with $j \neq 1$, the equation that must be checked is

$$e(\sigma_j, P) = R_{sj} \cdot e(h_s PK_j, Y) \cdot e(\sigma_{j-1}, P).$$

- Again, if the subsets $\mathcal{U}_i$ in the family $\mathcal{U}$ are all individual users $\mathcal{U}_i = \{U_i\}$, then the resulting signature is a standard ring signature according to the ID-based ring signature scheme proposed in Section 5.2.

5.3.2 Analysis of the Scheme

The proposed distributed ring signature schemes for ID-based scenarios satisfies the required properties of correctness, anonymity and unforgeability, as we show below.

Correctness

A distributed ring signature $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ computed by following the method explained above satisfies the verification equation. In effect:

$$e(\sigma, P) = e(\sigma_{n_s}, P) = e\left((\sum_{U_j \in \mathcal{U}_s} a_{sj} P + h_s SK_j) \ + \ (\sum_{1 \leq i \leq d, i \neq s} a_i P) \ , \ P \right) =$$

$$\left(\prod_{U_j \in \mathcal{U}_s} e(a_{sj} P, P) \cdot e(h_s x PK_j, P) \right) \prod_{1 \leq i \leq d, i \neq s} e(a_i P, P) =$$

$$= \left(\prod_{U_j \in \mathcal{U}_s} R_{sj} \cdot e(h_s PK_j, xP) \right) \prod_{1 \leq i \leq d, i \neq s} R_i =$$

$$= R_s \cdot e(\sum_{1 \leq i \leq d, i \neq s} h_i Y_i \ , \ Y) \cdot e(h_s \sum_{U_j \in \mathcal{U}_s} PK_j \ , \ Y) \prod_{1 \leq i \leq d, i \neq s} R_i = e(\sum_{i=1}^{d} h_i Y_i, Y) \prod_{i=1}^{d} R_i$$

Anonymity

With respect to the anonymity of the scheme, we can again argue as follows: the resulting signatures can be seen as ring signatures produced by the (individual) ring scheme proposed in Section 5.2, where the members of the ring are now the sets of users $\mathcal{U}_i$ in the family $\mathcal{U}$, with public keys $Y_i = \sum_{U_j \in \mathcal{U}_i} PK_j$.

The unconditional anonymity of the new ID-based distributed ring signature scheme directly infers from the anonymity property of the individual ID-based ring scheme, proved in Section 5.2.1. Informally, any verifier has no information about which set is the actual author of a given signature, because all the sets have the same probability of having computed it.

Unforgeability

We first review the definition of an adversary against distributed ring signature schemes, introduced in Section 4.4.2: a $(T, \varepsilon, Q_1, Q_2, Q_e, Q_s, \hat{n})$-successful attacker against a distributed ring signature scheme is an algorithm which receives as input the identities of the $\hat{n}$ considered users. This attacker runs in time T, makes Q_1 queries to the random oracle H_1, Q_2 queries to the random oracle H_2, adaptively corrupts Q_e users, obtaining their secret keys, and asks for Q_s valid distributed ring signatures. With probability at least ε, this algorithm obtains a valid new signature for a pair $(\mathcal{U}, m)$, such that all the sets of the family $\mathcal{U}$ contain at least one user who has not been corrupted by the adversary.

In the following theorem, we relate the difficulty of forging our ID-based distributed ring signature scheme with the difficulty of solving the Computational Diffie-Hellman problem.

Theorem 5.2. *Let $\mathcal{A}$ be a $(T, \varepsilon, Q_1, Q_2, Q_e, Q_s, \hat{n})$-successful adversary against the ID-based distributed ring signature scheme proposed in Section 5.3, with security parameter $k \geq 6$, such that $Q_s \leq \frac{2^{k/2}}{4}$ and $Q \leq \frac{2^{k/2}}{3}$.*

We denote by $\hat{d}$ the maximum number of subsets that can form the families for which $\mathcal{A}$ asks for a valid signature.

Then the Computational Diffie-Hellman problem in $\mathbb{G}_1$ can be solved in time $T' \leq 2T+2Q_1+2Q_2+4(\hat{d}+\hat{n})T_{bp}Q_s$ and with probability $\varepsilon' \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q_2,\hat{d}}}$.

Proof. Let (P, aP, bP) be the input of an instance of the Computational Diffie-Hellman problem in $\mathbb{G}_1$. Here P is a generator of $\mathbb{G}_1$, with prime order q, and the elements a, b are taken uniformly at random in $\mathbb{Z}_q^*$.

We construct a probabilistic polynomial time Turing machine $\mathcal{F}$ which will solve the given instance of the Computational Diffie-Hellman problem;

that is, it will compute the value abP. It will use the attacker $\mathcal{A}$ as a subroutine, so it must perfectly simulate the environment of the attacker $\mathcal{A}$. Later, we will apply Theorem 4.3 to the machine $\mathcal{F}$ with respect to the hash function H_2, so this machine is allowed to make Q_2 queries to the random oracle which models H_2.

The public data (P, aP, bP) is given to the machine $\mathcal{F}$. Then $\mathcal{F}$ runs the attacker $\mathcal{A}$ against our ID-based distributed ring signature scheme, answering to all the queries that $\mathcal{A}$ makes. First of all, $\mathcal{F}$ defines the public key of the master entity to be $Y = aP$, and gives it to the attacker $\mathcal{A}$, along with the list of the $\hat{n}$ identities of the considered users.

Without loss of generality, we can assume that $\mathcal{A}$ asks the random oracle H_1 for the value $H_1(ID)$ before asking for the secret key of ID.

We define $\mu = (5/6)^{1/Q_e}$. If we consider attacks where $Q_e = 0$, then we set $\mu = 0$. As we have done in the previous similar proofs, we can bound $1 - \mu \geq \frac{1}{6Q_e}$.

The machine $\mathcal{F}$ constructs a table TAB_{H_1} to simulate the random oracle H_1. Every time an identity ID_j is asked by $\mathcal{A}$ to the oracle H_1, the machine $\mathcal{F}$ first checks if this input is already in the table; if this is the case, then $\mathcal{F}$ returns to $\mathcal{A}$ the corresponding relation $H_1(ID_j) = PK_j$. Otherwise, $\mathcal{F}$ acts as follows: with probability μ, it chooses the random bit $c_j = 0$; in this case, $\mathcal{F}$ chooses a different $x_j \in \mathbb{Z}_q^*$ at random and defines $PK_j = x_j P$ and $SK_j = x_j Y$; the entry $(ID_j, PK_j, x_j, SK_j, c_j)$ is stored in the table TAB_{H_1}. On the other hand, with probability $1 - \mu$, the machine $\mathcal{F}$ chooses $c_j = 1$; in this case, it chooses a different $\alpha_j \in \mathbb{Z}_q^*$ at random and defines $PK_j = (\alpha_j)bP$ and $SK_j = \perp$. The values $(ID_j, PK_j, \alpha_j, c_j)$ are stored in a new entry of TAB_{H_1}. The relation $H_1(ID_j) = PK_j$ is sent to $\mathcal{A}$. The condition $PK_j \neq PK_\ell$ must be satisfied for all the different entries $j \neq \ell$ of the table; if this is not the case, the process is repeated for one of these users.

Since we are assuming that H_1 behaves as a random function, and the values PK_j are all randomly chosen, the information that results from this step is consistent for $\mathcal{A}$.

For any possible set of users $\mathcal{U}_i$, we define the value $Y_i = \sum_{U_j \in \mathcal{U}_i} PK_j$. Because of the way in which we have computed the values PK_j, the equation

$$Y_i = \gamma_i P + \delta_i (bP)$$

is fulfilled for some values $\gamma_i, \delta_i \in \mathbb{Z}_q$ that the machine $\mathcal{F}$ knows.

When $\mathcal{A}$ asks for the secret key corresponding to an identity ID_i, the machine $\mathcal{F}$ looks for ID_i in the table TAB_{H_1}. If $c_i = 0$, then $\mathcal{F}$ sends $SK_i = x_i Y$ to $\mathcal{A}$. If $c_i = 1$, the machine $\mathcal{F}$ cannot answer and halts. Note that the probability that $\mathcal{F}$ halts in this process is less than $1 - \mu^{Q_e} \leq \frac{1}{6}$.

To answer the queries that $\mathcal{A}$ makes to the random oracle for H_2, let us recall that the machine $\mathcal{F}$ has access to its own oracle for this hash function. Therefore, $\mathcal{F}$ obtains the answers from its oracle and sends them to $\mathcal{A}$.

The adversary $\mathcal{A}$ is allowed to query for Q_s valid ring signatures for messages and families of its choice. The machine $\mathcal{F}$ must simulate the information that $\mathcal{A}$ would obtain from these executions of the signing algorithm. Let B be the set of the users for whom $\mathcal{A}$ has asked for their secret keys (we call them corrupted users). When $\mathcal{A}$ asks for a valid signature for a message m' and a family $\mathcal{U}' = \{\mathcal{U}'_1, \ldots, \mathcal{U}'_d\}$, the machine $\mathcal{F}$ chooses at random one of the sets of $\mathcal{U}'$ to be the "real" author of the ring signature; for simplicity, we denote this set as $\mathcal{U}'_s = \{U'_1, U'_2, \ldots, U'_{n_s}\}$. The information that $\mathcal{A}$ would obtain from such a real computation consists of the secret information generated by the corrupted players in $B \cap \mathcal{U}'_s$, as well as all the information broadcast in the private broadcast channel of $\mathcal{U}'_s$. This is because we can consider the worst case where some of the users in $\mathcal{U}'_s$ is corrupted, and so $\mathcal{A}$ has access to this channel. The machine $\mathcal{F}$ must execute the following algorithm in order to simulate this information:

1. For each user $U'_\ell \in \mathcal{U}'_s \cap B$, choose at random $a_{s\ell} \in \mathbb{Z}_q^*$, compute and broadcast $R'_{s\ell} = e(a_{s\ell}P, P)$.

2. Choose, for all $i = 1, \ldots, d$, $i \neq s$, random values $a_i \in \mathbb{Z}_q^*$, pairwise different, and compute $R'_i = e(a_i P, P)$ and $h_i = H_2(\mathcal{U}', m', R_i)$ (by asking to the oracle that models H_2).

3. Choose at random $h'_s \in \mathbb{Z}_q$.

4. For user U'_1:

   - if $U'_1 \in B$ (since $\mathcal{F}$ has not halted, this means that the machine $\mathcal{F}$ knows the secret key SK_1 of this corrupted user, as well as the value a_{s1}), compute $\sigma_1 = a_{s1}P + h'_s SK_1 + \sum_{1 \leq i \leq d, i \neq s} a_i P$;
   - if $U'_1 \notin B$, choose at random $\sigma_1 \in \mathbb{G}_1$ and compute

$$R'_{s1} = e(\sigma_1, P) \cdot e(h'_s P K_1, -Y) \cdot \prod_{1 \leq i \leq d, i \neq s} (R'_i)^{-1}.$$

5. For user U'_j, for $j = 2, \ldots, n_s$:

   - if $U'_j \in B$ (since $\mathcal{F}$ has not halted, this means that the machine $\mathcal{F}$ knows the secret key SK_j of this corrupted user, as well as the value a_{sj}), compute $\sigma_j = a_{sj}P + h'_s SK_j + \sigma_{j-1}$;

- if $U'_j \notin B$, choose at random $\sigma_j \in \mathbb{G}_1$ and compute

$$R'_{sj} = e(\sigma_j - \sigma_{j-1}, P) \cdot e(h'_s PK_j, -Y).$$

6. Compute the value

$$R'_s = e(-Y, \sum_{1 \leq i \leq d, i \neq s} h'_i Y_i) \prod_{U_j \in \mathcal{U}_s} R'_{sj}.$$

If $R'_s = 1$ or $R'_s = R'_i$ for some $i = 1, \ldots, d$, $i \neq s$, then return to step 1.

7. Now $\mathcal{F}$ "falsifies" the random oracle H_2, by setting the relation $H_2(\mathcal{U}', m', R'_s) = h'_s$. If $\mathcal{A}$ asks again for this input to the random oracle H_2, then $\mathcal{F}$ will obviously answer with h'_s.

8. Define $\sigma' = \sigma_{n_s}$ and return the tuple $(\mathcal{U}', m', R'_1, \ldots, R'_d, h'_1, \ldots, h'_d, \sigma')$.

In each execution of this simulation process, $\mathcal{F}$ must perform less than $2(d + n_s)$ evaluations of the bilinear pairing.

The resulting signature is valid and indistinguishable from one obtained with the signing algorithm, because we are assuming that H_2 behaves as a random function and h_s is taken at random from $\mathbb{Z}_q$. However, this assignment $H_2(\mathcal{U}', m', R'_s) = h'_s$, in step 7 of the simulation, can cause some collision if the query $(\mathcal{U}', m', R'_s)$ has been previously made to the random oracle H_2, or if the same tuple is produced twice in two different runs of the signature simulation algorithm.

Since no R'_i appears with probability greater than $1/2^{k-1}$ in a simulated ring signature, we can bound the probability that such collisions occur:

- The probability that a tuple $(\mathcal{U}', m', R'_s)$ that $\mathcal{F}$ outputs, as part of a simulated ring signature, has been asked before to the random oracle by $\mathcal{A}$ is less than $Q_2 \cdot Q_s \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$.

- The probability that the same tuple is output two times by $\mathcal{F}$ in two different signature simulations is less than $\frac{Q_s^2}{2} \cdot \frac{1}{2^{k-1}} \leq \frac{1}{6}$ (using Fact 4.1).

Altogether, the probability of collisions is less than $1/3$. The probability that the machine $\mathcal{F}$ succeeds in obtaining a valid ring signature is the following:

$$\varepsilon_{\mathcal{F}} = \Pr[\mathcal{F} \text{ succeeds}] =$$

$\Pr[\mathcal{F}$ does not halt AND no-collisions in the simulations AND $\mathcal{A}$ succeeds$] \geq$

$\geq \Pr[\mathcal{A}$ succeeds $\mid \mathcal{F}$ does not halt AND no-collisions in the simulations $]\cdot$

$\cdot (1 - \Pr[\mathcal{F}$ halts OR collisions in the simulations$]) \geq \varepsilon \left(1 - \frac{1}{6} - \frac{1}{3}\right) = \frac{\varepsilon}{2}.$

The execution time of the machine $\mathcal{F}$ is $T_{\mathcal{F}} \leq T + Q_1 + Q_2 + 2(\hat{d} + \hat{n})T_{bp}Q_s$. Summing up, we have a Turing machine $\mathcal{F}$ that forges a generic distributed ring signature scheme in time $T_{\mathcal{F}}$ and with probability $\varepsilon_{\mathcal{F}} \geq \frac{\varepsilon}{2}$. Therefore, we can apply the Distributed Ring Forking Lemma (Theorem 4.3) to the machine $\mathcal{F}$.

As a result, by executing twice the machine $\mathcal{F}$, we will obtain in time $T' \leq 2T_{\mathcal{F}}$ and with probability $\tilde{\varepsilon}' \geq \frac{\varepsilon_{\mathcal{F}}^2}{65V_{Q_2,\hat{d}}}$ two valid distributed ring signatures $(\mathcal{U}, m, R_1, \ldots, R_d, h_1, \ldots, h_d, \sigma)$ and $(\mathcal{U}, m, R_1, \ldots, R_d, h'_1, \ldots, h'_d, \sigma')$ such that $h_j \neq h'_j$, for some $j \in \{1, \ldots, d\}$ and $h_i = h'_i$ for all $i = 1, \ldots, d$ such that $i \neq j$.

By definition of valid forgery against a distributed ring signature scheme, there exists at least one non-corrupted user in each subset $\mathcal{U}_i \in \mathcal{U}$; in particular there exists a non-corrupted user $U_z \in \mathcal{U}_j - B$ in the subset $\mathcal{U}_j$. Remember that $Y_j = \gamma_j P + \delta_j(bP)$, where γ_j and δ_j are values known by the machine $\mathcal{F}$.

For this non-corrupted user $U_z \in \mathcal{U}_j$, we have $c_z = 1$ with probability $1 - \mu$, which means that $PK_z = \alpha_z(bP)$. So the value α_z is one of the terms added in the factor δ_j that appears in Y_j. If this is the case, then with overwhelming probability we will have that $\delta_j \neq 0 \bmod q$.

If now we come back to the two forged signatures, and we write the corresponding verification equations, we have:

$$e(\sigma, P) = R_1 \cdot \ldots \cdot R_d \cdot e(Y, h_1 Y_1) \cdot \ldots \cdot e(Y, h_d Y_d)$$

$$e(\sigma', P) = R_1 \cdot \ldots \cdot R_d \cdot e(Y, h'_1 Y_1) \cdot \ldots \cdot e(Y, h'_d Y_d)$$

Dividing these two equations, we obtain $e(\sigma - \sigma', P) = e(Y, (h_j - h'_j)Y_j) = e\big(aP, (h_j - h'_j)(\gamma_j P + \delta_j(bP))\big) = e\big(aP, (h_j - h'_j)\gamma_j P\big) \cdot e\big(aP, (h_j - h'_j)\delta_j(bP)\big)$.

We can conclude from this relation the equality

$$e\big(ab\delta_j(h_j - h'_j)P, P\big) = e\big(\sigma - \sigma' - (a\gamma_j(h_j - h'_j))P, P\big).$$

Since the pairing is non-degenerate, this implies that $ab\delta_j(h_j - h'_j)P = \sigma - \sigma' - [a\gamma_j(h_j - h'_j)]P$. Therefore, one can compute the solution of the given

instance of the Computational Diffie-Hellman problem:

$$abP = \frac{1}{\delta_j(h_j - h'_j)}(\sigma - \sigma') - \frac{a\gamma_j}{\delta_j}P \ .$$

The inverses are computed modulo q, and they always exist because $h_j \neq h'_j$ and $\delta_j \neq 0 \bmod q$ with overwhelming probability.

Summing up, the machine $\mathcal{F}$ has solved the Computational Diffie-Hellman problem with probability

$$\varepsilon' = (1-\mu)\tilde{\varepsilon}' \geq (1-\mu)\frac{\varepsilon_{\mathcal{F}}^2}{65V_{Q_2,\hat{d}}} \geq (1-\mu)\frac{(\varepsilon/2)^2}{65V_{Q_2,\hat{d}}} \geq \frac{\varepsilon^2}{1560\ Q_e V_{Q_2,\hat{d}}}.$$

And the total time needed to solve the problem has been $T' \leq 2T_{\mathcal{F}} \leq 2T + 2Q_1 + 2Q_2 + 4(\hat{d} + \hat{n})T_{bp}Q_s$.

□

5.4 A Different Construction from Dual Access Structures

As it happened in the proposal for the Disc-Log scenario (Section 4.4.1), the efficiency of the distributed ID-based ring signature scheme proposed in the previous section depends on the total number of users and the number of sets in the family of possible signing sets.

For example, the construction is not very efficient in a threshold scenario: t users want to anonymously sign a message on behalf of a set (or ring) of n users which includes themselves, in such a way that the recipient is convinced that t of the n users of the set have cooperated to compute a signature, but he does not have any information about who the t signers are. In this case, the number of possible signing subsets is $\begin{pmatrix} n \\ t \end{pmatrix}$, which can be a quite large number. Other examples where the above construction can be inefficient are weighted threshold families (introduced in [88]), multipartite families [81], etc.

We next propose a different scheme for computing distributed ring signatures in a more efficient way, in an ID-based scenario. As we mention in Section 5.4.2, the new scheme can be modified to work in more general scenarios, where users can have different kinds of keys (ID-based, or RSA keys, or Disc-Log keys) and of different sizes. For simplicity, we formally analyze the scheme only for the particular case where all the keys are based on identities, with the same common parameters.

The proposal follows the ideas introduced in [94], where threshold ring signatures are designed for PKI scenarios (with users having either Disc-Log or RSA keys, for example). Our scheme admits ID-based keys and, more importantly, it runs with general families of possible signing sets, not only with threshold ones. Independently, a different distributed ring signature scheme for ID-based scenarios has been proposed in [25], but again it works only in the threshold case.

The protocols of our proposed scheme are described below. We will consider, as in the previous proposals of distributed ring schemes, that any specific set of users can always have access to a private and authenticated broadcast channel.

Key generation. Let $\mathbb{G}_1$ be an additive group of prime order q, generated by some element P. Let $\mathbb{G}_2$ be a multiplicative group with the same order q. We need $q \geq 2^k$, where k is the security parameter of the scheme. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear pairing as defined in Section 5.1. Let $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ be two hash functions.

The master entity chooses at random his secret key $x \in \mathbb{Z}_q^*$ and publishes the value $Y = xP$.

Secret key extraction. Any user U_i of the system, with identity ID_i (which can be an IP or e-mail address, for example), has public key $PK_i = H_1(ID_i)$. When he requests the master for his matching secret key, he obtains the value $SK_i = xPK_i$.

Distributed ring signature generation. Assume that a subset of users A want to compute an anonymous signature on behalf of a family $\mathcal{U} \subset 2^{\mathcal{P}}$ of possible signing subsets, taken over a set $\mathcal{P} = \{U_1, \ldots, U_n\}$ of n users. Users in A choose the family $\mathcal{U}$ in an ad-hoc way, with the only condition that $A \in \mathcal{U}$.

We will assume that the family of subsets $\mathcal{U}$ is in some way *normalized*: there do not exist two subsets $A, B \in \mathcal{U}$ such that $A \subset B$. In this case, it is easy to see that $(cl(\mathcal{U}))_0 = \mathcal{U}$; that is, $\mathcal{U}$ is the basis of its closure (recall the definition of closure in Section 1.4).

For simplicity, we will assume that $cl(\mathcal{U})$ is a vector space access structure. In this case, we consider the dual access structure (see Section 1.4.1) $\Gamma = (cl(\mathcal{U}))^*$, which is also a vector space access structure: there exist a positive integer r and a mapping $\psi : \mathcal{P} \cup \{D\} \to \mathbb{Z}_q^r$ such that $B \in \Gamma$ if and only if $\psi(D) \in \langle \{\psi(U_i)\}_{U_i \in B} \rangle$. Our construction can be easily extended to the case of more general access structures, where the mapping ψ assigns possibly more than one vector to some users. For example, a generic solution would be to use the construction of Simmons et al. [92]: if $\mathcal{U} = \{A_1, \ldots, A_d\}$, then

ψ assigns vectors in $GF(q)^d$ in the following way: $\psi(D) = (1, 0, \ldots, 0)$ and $\psi(U) = \{(1, i, i^2, \ldots, i^{d-1}) : U \in A_i\}$ for any user $U \in \mathcal{P}$. This assignment ψ realizes the access structure $\Gamma = (cl(\mathcal{U}))^*$.

Since $\mathcal{U} = (cl(\mathcal{U}))_0$, we have that $A \in (cl(\mathcal{U}))_0$. This means that $\mathcal{P} - A \notin \Gamma$, and is maximal with respect to the inclusion, meaning that $(\mathcal{P} - A) \cup \{U_j\} \in \Gamma$ for any user $U_j \in A$.

The signing users in A execute the following protocol to compute a valid distributed signature on a message $m \in \{0,1\}^*$:

1. They consider a basis of the subspace $\langle \psi(\mathcal{P} - A) \rangle$. This basis corresponds to some subset of users $C \subset \mathcal{P} - A$; that is, vectors in $\psi(C)$ are linearly independent and $\langle \psi(C) \rangle = \langle \psi(\mathcal{P} - A) \rangle$.

2. For every user $U_i \in C$, the signing users choose uniformly at random $c_i \in \mathbb{Z}_q$ and $R_i \in \mathbb{G}_1$; they compute and broadcast the value

$$z_i = e(R_i, P) \cdot e(Y, c_i PK_i).$$

3. For users $U_t \in (\mathcal{P} - A) - C$, we have that $\psi(U_t) = \sum_{U_i \in C} \lambda_{it} \psi(U_i)$, for some $\lambda_{it} \in \mathbb{Z}_q$, because $\psi(C)$ is a basis of $\langle \psi(\mathcal{P} - A) \rangle$. The signing users choose uniformly at random $R_t \in \mathbb{G}_1$ and consider $c_t = \sum_{U_i \in C} \lambda_{it} c_i$; they compute and broadcast the value

$$z_t = e(R_t, P) \cdot e(Y, c_t PK_t).$$

4. Each signing user $U_j \in A$ chooses uniformly at random $T_j \in \mathbb{G}_1$; he computes and broadcasts the value

$$z_j = e(T_j, P).$$

5. The signing users compute then the value $c = H_2(\mathcal{U}, m, z_1, \ldots, z_n)$.

6. They choose uniformly at random one of the vectors $\mathbf{v} \in \mathbb{Z}_q^r$ that verifies:

   (i) $\mathbf{v}\psi(D) = c$, and

   (ii) $\mathbf{v}\psi(U_i) = c_i$, for all $U_i \in C$.

   Note that this vector $\mathbf{v}$ exists because $C \notin \Gamma$ and so the vectors $\{\psi(D), \{\psi(U_i)\}_{U_i \in C}\}$ are linearly independent.

7. Every signing user $U_j \in A$ computes $c_j = \mathbf{v}\psi(U_j)$ individually; then he computes and broadcasts the value

$$R_j = T_j - c_j SK_j.$$

Note that the rest of users in A can verify that this value R_j is consistent with the value z_j broadcast in step 4, by checking if $z_j = e(R_j, P) \cdot e(Y, c_j PK_j)$. In this way, they detect dishonest users who try to boycott the process.

8. The resulting signature is $(\mathcal{U}, m, \mathbf{v}, R_1, \ldots, R_n, \psi)$.

Note that the length of the signature is linear with respect to the number n of users. Recall that in the other distributed ring signature schemes proposed in this thesis, the length of the signature was linear with respect to the number of possibly signing subsets in the family.

Verification of a distributed ring signature. The recipient of the message first computes $c_i = \mathbf{v}\psi(U_i)$, for every user $U_i \in \mathcal{P}$ and then computes the values

$$z_i = e(R_i, P) \cdot e(Y, c_i PK_i).$$

The signature is valid if $\mathbf{v}\psi(D) = H_2(\mathcal{U}, m, z_1, \ldots, z_n)$.

5.4.1 Analysis of the Scheme

In this section we prove that our new scheme satisfies the three required properties for distributed ring signature schemes: correctness, anonymity and unforgeability. The two last properties are proved to be achieved in the random oracle model.

Correctness

We show that a signature that has been generated following the above method is always valid. The vector $\mathbf{v}$ in the signature satisfies

(i) $\mathbf{v}\psi(D) = c$, and

(ii) $\mathbf{v}\psi(U_i) = c_i$, for all $U_i \in C$.

Therefore, for users U_i in the set C, we have that $c_i = \mathbf{v}\psi(U_i)$ and $z_i = e(R_i, P) \cdot e(Y, c_i PK_i)$.

For users $U_t \in (\mathcal{P} - A) - C$, we have that $\psi(U_t) = \sum_{U_i \in C} \lambda_{it}\psi(U_i)$, by definition of the set C. This implies that

$$c_t = \sum_{U_i \in C} \lambda_{it} c_i = \sum_{U_i \in C} \lambda_{it} \mathbf{v}\psi(U_i) = \mathbf{v}\psi(U_t).$$

And $z_t = e(R_t, P) \cdot e(Y, c_t PK_t)$ for these users, as well.

Finally let us consider users $U_j \in A$. By construction, the equality $c_j = \mathbf{v}\psi(U_j)$ is also satisfied. Note that these values are independent of the choice of the vector $\mathbf{v}$, as long as it satisfies the two required conditions. In effect, as far as $\langle \psi(C) \rangle = \langle \psi(\mathcal{P} - A) \rangle$ and $\mathcal{P} - A$ is maximal verifying $\mathcal{P} - A \notin \Gamma$, then $C \cup \{U_j\} \in \Gamma$, for any user $U_j \in A$. So there exist coefficients λ_j and $\{\lambda_{ji}\}_{U_i \in C}$ satisfying

$$\psi(D) = \sum_{U_i \in C} \lambda_{ji}\psi(U_i) \ + \ \lambda_j \psi(U_j)$$

where $\lambda_j \neq 0$. From this equality we can derive

$$c_j = \mathbf{v}\psi(U_j) = \lambda_j^{-1} \left(\mathbf{v}\psi(D) - \sum_{U_i \in C} \lambda_{ji} \mathbf{v}\psi(U_i) \right) = \lambda_j^{-1} \left(c - \sum_{U_i \in C} \lambda_{ji} c_i \right),$$

which does not depend on the specific vector $\mathbf{v}$.

Furthermore, for users $U_j \in A$ we have that

$$z_j = e(T_j, P) = e(R_j + c_j SK_j, P) = e(R_j, P) \cdot e(c_j x PK_j, P) = e(R_j, P) \cdot e(c_j PK_j, Y),$$

as desired.

Therefore, for all users U_i in $\mathcal{P}$ we have that $c_i = \mathbf{v}\psi(U_i)$ and $z_i = e(R_i, P) \cdot e(Y, c_i PK_i)$, and so the correctness of the signature is verified because $\mathbf{v}\psi(D) = c = H_2(\mathcal{U}, m, z_1, \ldots, z_n)$.

Anonymity

Given a valid distributed ring signature $Sig = (\mathcal{U}, m, \mathbf{v}, R_1, \ldots, R_n, \psi)$ on behalf of a family of subsets of users $\mathcal{U}$, the probability that a particular subset $B \in \mathcal{U}$ is the author of this signature can be exactly computed. In effect, if the full set of users is $\mathcal{P}$, we know that $\psi : \mathcal{P} \cup \{D\} \to \mathbb{Z}_q^r$ is a mapping which defines the access structure $\Gamma = (cl(\mathcal{U}))^*$. Since $B \in \mathcal{U} = (cl(\mathcal{U}))_0$, we have that $\mathcal{P} - B \notin \Gamma$. Let $C \subset \mathcal{P} - B$ be a subset of users such that $\langle \psi(C) \rangle = \langle \psi(\mathcal{P} - B) \rangle$ and such that the vectors in $\{\psi(U_i)\}_{U_i \in C}$ are linearly independent. Since $C \notin \Gamma$, we have that the set of vectors

$\{\psi(D), \{\psi(U_i)\}_{U_i \in C}\}$ are linearly independent in $\mathbb{Z}_q^r$. Therefore, the number of users in C is $\omega = |C| \leq r - 1$.

Consider the values $c = \mathbf{v}\psi(D)$ and $c_i = \mathbf{v}\psi(U_i)$, for all users $U_i \in C$. The probability that users in B choose these values $\{c_i\}_{U_i \in C}$ in step 2 of the signing protocol is exactly $1/q^\omega$. Later, the value c is the output of the hash function H_2. If we assume that this hash function behaves as a random oracle, then the probability that users in B obtain this value c in step 5 of the protocol is exactly $1/q$, independently of the inputs taken by the hash function.

After that, users in B would choose at random one vector among the solutions of the system of equations $M\mathbf{x} = \mathbf{b}$, where

$$M = \begin{pmatrix} \cdots & \psi(D) & \cdots \\ \cdots & \psi(U_{i_1}) & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \psi(U_{i_\omega}) & \cdots \end{pmatrix} \qquad b = \begin{pmatrix} c \\ c_{i_1} \\ \vdots \\ c_{i_\omega} \end{pmatrix} ,$$

if we denote $C = \{U_{i_1}, \ldots, U_{i_\omega}\}$.

The number of different vectors in $\mathbb{Z}_q^r$ which are solution of this system is q^γ, where $\gamma = \dim(\ker M) = r - \dim(\mathrm{Im} M) = r - (\omega + 1)$. Therefore, the probability that users in B choose in step 6 of the protocol the vector $\mathbf{v}$ that appears in Sig is exactly $1/q^\gamma$.

The probability that members of B choose, in steps 2 and 3 of the signing protocol, the values $\{R_i\}_{U_i \notin B}$ that appear in Sig and, in step 4, the values $\{T_j\}_{U_j \in B}$ that lead to the values $\{R_j\}_{U_j \in B}$ in Sig is exactly equal to $1/q^n$.

Summing up, the probability that users in B obtain the signature Sig when they execute the signing protocol is exactly

$$\frac{1}{q^\omega} \cdot \frac{1}{q} \cdot \frac{1}{q^\gamma} \cdot \frac{1}{q^n} = \frac{1}{q^{\omega+1+\gamma+n}} = \frac{1}{q^{r+n}} ,$$

which does not depend on B and so is the same for all the subsets in the family $\mathcal{U}$. This proves that the scheme is unconditionally anonymous, in the random oracle model for the hash function H_2.

Unforgeability

As in the previous proposals, we will analyze the exact unforgeability of our scheme, that measures all the resources and performances of an adversary against it. Recall that a $(T, \varepsilon, Q_1, Q_2, Q_e, Q_s, \hat{n})$-successful adversary against a distributed ring signature scheme is given as input a list with the identities

of the $\hat{n}$ considered users. This adversary is allowed to adaptively corrupt up to Q_e users, obtaining their secret keys. The adversary can also make Q_1 queries to the random oracle H_1 and Q_2 queries to the random oracle H_2. Finally, the adversary can require the execution of the signing algorithm for Q_s pairs of messages and families of subsets that it adaptively chooses, obtaining a valid distributed ring signature for each query. As a result, it obtains in time T and with probability at least ε a valid and new distributed ring signature for some message m and some family of subsets $\mathcal{U}$, such that all the subsets in the family $\mathcal{U}$ contain at least one non-corrupted user.

Remember that, without loss of generality, we can assume that $Q_1 \geq 1$ and $Q_2 \geq 1$, because otherwise the probability that a forger guesses the values of the two (random) hash functions and produces a valid forgery is negligible.

In the following theorem, we relate the unforgeability of our scheme to the difficulty of solving the Computational Diffie-Hellman problem.

Theorem 5.3. *Let $\mathcal{A}$ be a $(T, \varepsilon, Q_1, Q_2, Q_e, Q_s, \hat{n})$-successful adversary against the ID-based distributed ring signature scheme proposed in Section 5.4, with security parameter $k \geq 9$, and such that $Q_s \leq \frac{2^{k/2}}{4}$ and $Q_2 \leq \frac{2^{k/2}}{3}$.*

Then the Computational Diffie-Hellman problem in $\mathbb{G}_1$ can be solved with probability $\varepsilon' \geq \frac{\varepsilon^2}{385 Q_e Q_2}$ and in time $T' \leq 2T + 2Q_1 + 2Q_2 + 2(T_\psi + 2\hat{n}T_{bp})Q_s$, where T_ψ is the expected time to perform some specific computations related to the access structure defined by the assignment of vectors ψ.

Proof. We are going to construct a probabilistic polynomial time Turing machine $\mathcal{F}$ which will use the attacker $\mathcal{A}$ as a sub-routine in order to solve the given instance of the Computational Diffie-Hellman problem. Therefore, $\mathcal{F}$ must perfectly simulate the environment of the attacker $\mathcal{A}$.

The machine $\mathcal{F}$ receives the public data (P, aP, bP), and its goal is to compute the value abP. The public key of the master entity is defined to be $Y = aP$. Then $\mathcal{F}$ runs the attacker $\mathcal{A}$ against the threshold ID-based ring signature scheme, answering to all the queries that $\mathcal{A}$ makes. The public key $Y = aP$ is also sent to the attacker $\mathcal{A}$. Finally, let $\mathcal{P}$ denote the set of $\hat{n}$ considered users. The identities of these users are also provided to $\mathcal{A}$.

Without loss of generality, we can assume that $\mathcal{A}$ asks the random oracle H_1 for the value $H_1(ID)$ before asking for the secret key of ID.

Let us define $\mu = (5/6)^{1/Q_e}$ (we assume $Q_e \geq 1$; otherwise, we would take $\mu = 0$).

The machine $\mathcal{F}$ constructs a table TAB_{H_1} to simulate the random oracle H_1. Every time an identity ID_i is asked by $\mathcal{A}$ to the oracle H_1, the machine

$\mathcal{F}$ acts as follows: first $\mathcal{F}$ checks if this input is already in the table; if this is the case, then $\mathcal{F}$ sends to $\mathcal{A}$ the corresponding relation $H_1(ID_i) = PK_i$. Otherwise, with probability μ, the machine $\mathcal{F}$ chooses the bit $d_i = 0$ and a different $x_i \in \mathbb{Z}_q^*$ at random, and defines $PK_i = x_i P$ and $SK_i = x_i Y$; the new entry $(ID_i, PK_i, x_i, SK_i, d_i)$ is stored in the table TAB_{H_1}. On the other hand, with probability $1 - \mu$, the machine $\mathcal{F}$ chooses the bit $d_i = 1$ and a different $\alpha_i \in \mathbb{Z}_q^*$ at random, and defines $PK_i = (\alpha_i)bP$ (in this case $\mathcal{F}$ does not know the secret key for this identity). The values $(ID_i, PK_i, \alpha_i, d_i)$ are stored in a new entry of TAB_{H_1}. The relation $H_1(ID_i) = PK_i$ is sent to $\mathcal{A}$. The condition $PK_i \neq PK_j$ must be satisfied for all the different entries $i \neq j$ of the table; if this is not the case, the process is repeated for one of these users.

Since we are assuming that H_1 behaves as a random function, and the values PK_i are all randomly chosen, this simulation of the hash function H_1 produces consistent values for $\mathcal{A}$.

Later, every time $\mathcal{A}$ asks for the secret key corresponding to an identity ID_i, the machine $\mathcal{F}$ looks for ID_i in the table TAB_{H_1}. If $d_i = 0$, then $\mathcal{F}$ sends $SK_i = x_i Y$ to $\mathcal{A}$. If $d_i = 1$, the machine $\mathcal{F}$ cannot answer and halts. The probability that $\mathcal{F}$ halts in this process is less than $1 - \mu^{Q_e} = 1/6$.

$\mathcal{F}$ also constructs a table TAB_{H_2} to simulate the random oracle H_2. Every time $\mathcal{A}$ makes a query to this oracle, $\mathcal{F}$ looks for this value in the table. If it is already there, then $\mathcal{F}$ sends the corresponding relation to $\mathcal{A}$; if not, $\mathcal{F}$ chooses at random an output of the random oracle for the queried input, different from the outputs which are already in the table, sends the relation to $\mathcal{A}$ and stores it in the table TAB_{H_2}.

Finally, the attacker $\mathcal{A}$ can ask Q_s times for valid distributed ring signatures for messages m' and families of subsets $\mathcal{U}' \subset 2^{\mathcal{P}}$. To answer such queries, the machine $\mathcal{F}$ proceeds as follows:

1. Define $\Gamma = (cl(\mathcal{U}))^*$; then find a mapping $\psi' : \mathcal{P} \cup \{D\} \rightarrow \mathbb{Z}_q^{r'}$ such that $B \in \Gamma$ if and only if $\psi'(D) \in \langle \psi'(B) \rangle$. Then choose a subset $A \in \mathcal{U}$; consider a basis of the subspace $\langle \psi'(\mathcal{P} - A) \rangle$. This basis corresponds to some subset of users $C \subset \mathcal{P} - A$.

2. For every user $U_i \in C$, choose uniformly at random $c_i' \in \mathbb{Z}_q$. Choose uniformly at random a value $c' \in \mathbb{Z}_q$.

3. Choose at random a vector $\mathbf{v}'$ among the set of vectors $\mathbf{v}$ satisfying $\mathbf{v}\psi'(D) = c'$ and $\mathbf{v}\psi'(U_i) = c_i'$ for all users $U_i \in C$.

4. For users $U_j \in \mathcal{P} - C$, compute the values $c_j' = \mathbf{v}'\psi'(U_j)$.

5. Choose at random $\hat{n}$ values $R'_1, \ldots, R'_{\hat{n}} \in \mathbb{G}_1$, one for each user in $\mathcal{P}$.
6. Compute, for $i = 1, \ldots, \hat{n}$, the values $z'_i = e(R'_i, P) \cdot e(Y, c'_i PK_i)$.
7. Impose and store in the table TAB_{H_2} the relation $H_2(\mathcal{U}', m', z'_1, \ldots, z'_{\hat{n}}) = c'$.
8. Define the signature to be $(\mathcal{U}', m', \mathbf{v}', R'_1, \ldots, R'_{\hat{n}}, \psi')$.

In each of these simulations, the machine $\mathcal{F}$ must find a suitable assignment ψ, choose at random some values, then choose a vector $\mathbf{v}$, and so on. We denote as T_ψ a bound for the expected time necessary for performing all these tasks. Furthermore, $\mathcal{F}$ must compute $2\hat{n}$ values of the bilinear pairing.

The process results in a valid distributed ring signature, because we are assuming that H_2 behaves as a random function, and c' is taken uniformly at random in $\mathbb{Z}_q$. However, the assignment $H_2(\mathcal{U}', m', z'_1, \ldots, z'_{\hat{n}}) = c'$ can produce some collisions in the management of the table TAB_{H_2} that simulates the random oracle H_2.

A first possible collision occurs if a tuple $(\mathcal{U}', m', z'_1, \ldots, z'_{\hat{n}})$ produced in the simulation of a signature has been already queried to the random oracle H_2. The probability of this event is less than $\frac{Q_s Q_2}{q} \leq 1/12$.

A second possible collision occurs when the same tuple $(\mathcal{U}', m', z'_1, \ldots, z'_{\hat{n}})$ is produced in two different signature simulations. Using again the result stated in Fact 4.1, we have that the probability of this event is less than $\frac{Q_s^2}{2q} \leq 1/12$.

We denote by ω the whole set of random tapes that take part in an attack by $\mathcal{A}$, with the environment simulated by $\mathcal{F}$, but excluding the randomness related to the oracle H_2. The success probability of $\mathcal{A}$ in forging a valid ring signature scheme is then taken over the space (ω, H_2).

In an execution of the attacker $\mathcal{A}$, we use the notation $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_{Q_2}$ for the different queries that $\mathcal{A}$ makes to the random oracle H_2. If $\mathcal{A}$ produces a valid forged signature $(\mathcal{U}, m, \mathbf{v}, R_1, \ldots, R_{\hat{n}}, \psi)$, by the ideal randomness of the oracle H_2, the probability that $\mathcal{A}$ has not asked to this oracle for the corresponding tuple $(\mathcal{U}, m, z_1, \ldots, z_{\hat{n}})$, and so $\mathcal{A}$ must have guessed the corresponding output, is less than $\frac{1}{q}$. We define $\beta = \infty$ in this case; otherwise, β denotes the index of the query where the tuple above was asked. That is, $\mathcal{Q}_\beta = (\mathcal{U}, m, z_1, \ldots, z_{\hat{n}})$.

We denote by $\mathcal{S}$ the set of successful executions of $\mathcal{A}$, with $\mathcal{F}$ simulating its environment, and such that $\beta \neq \infty$. We also define the following subsets of $\mathcal{S}$: for every $i = 1, 2, \ldots, Q_2$, the set $\mathcal{S}_i$ contains the successful executions such that $\beta = i$.

This gives us a partition $\{\mathcal{S}_i\}_{i=1,\ldots,Q_2}$ of $\mathcal{S}$ in exactly Q_2 classes.

The probability that an execution (ω, H_2) of $\mathcal{A}$ with the environment simulated by $\mathcal{F}$ results in a valid forgery with $\beta \neq \infty$ is

$$\tilde{\varepsilon} = \Pr[(\omega, H_2) \in \mathcal{S}] \geq \varepsilon \left(1 - \frac{1}{q}\right) \left(1 - (1 - \mu^{Q_e}) - \frac{Q_s Q_2}{q} - \frac{Q_s^2}{2q}\right) \geq$$

$$\geq \varepsilon \cdot \frac{3}{4} \cdot \left(1 - \frac{1}{3}\right) = \frac{\varepsilon}{2}.$$

Now we define the set of indexes which are more likely to appear as

$$I = \{i \text{ such that } \Pr[(\omega, H_2) \in \mathcal{S}_i \mid (\omega, H_2) \in \mathcal{S}] \geq \frac{1}{2Q_2}\}.$$

And the corresponding subset of successful executions as $\mathcal{S}_I = \{(\omega, H_2) \in \mathcal{S}_i$ such that $i \in I\}$.

For a specific index $i \in I$, the following inequality holds:

$$\Pr[(\omega, H_2) \in \mathcal{S}_i] = \Pr[(\omega, H_2) \in \mathcal{S}] \cdot \Pr[(\omega, H_2) \in \mathcal{S}_i \mid (\omega, H_2) \in \mathcal{S}] \geq$$

$$\geq \tilde{\varepsilon} \cdot \frac{1}{2Q_2}.$$

Lemma 5.1. *It holds that* $\Pr[(\omega, H_2) \in \mathcal{S}_I \mid (\omega, H_2) \in \mathcal{S}] \geq 1/2$.

Proof. Since the sets $\mathcal{S}_i$ are disjoint, we can write

$$\Pr[(\omega, H_2) \in \mathcal{S}_I \mid (\omega, H_2) \in \mathcal{S}] = \sum_{i \in I} \Pr[(\omega, H_2) \in \mathcal{S}_i \mid (\omega, H_2) \in \mathcal{S}] =$$

$$1 - \sum_{i \notin I} \Pr[(\omega, H_2) \in \mathcal{S}_i \mid (\omega, H_2) \in \mathcal{S}].$$

Since the complement of I contains at most Q_2 indexes, we have that this probability is greater than $1 - Q_2 \cdot \frac{1}{2Q_2} = 1/2$. □

We come back to the execution of $\mathcal{A}$ with the environment simulated by $\mathcal{F}$. With probability at least $\tilde{\varepsilon}$, such an execution (ω, H_2) results in a valid forgery with $\beta \neq \infty$. In this case, applying Lemma 5.1, we know that this successful execution belongs to $\mathcal{S}_I$ with probability at least $1/2$.

Now we split H_2 as (H_2', c), where H_2' corresponds to the answers of all the queries to H_2 except the query $\mathcal{Q}_\beta$, whose answer is denoted as c.

We apply the Splitting Lemma (Lemma 4.1), taking $X = (\omega, H_2')$, $Y = c$, $A = \mathcal{S}_\beta$, $\delta = \frac{\tilde{\varepsilon}}{2Q_2}$ and $\alpha = \frac{\tilde{\varepsilon}}{4Q_2}$. The lemma says that there exists a subset of executions Ω_β such that

$$\Pr[(\omega, H_2) \in \Omega_\beta \mid (\omega, H_2) \in \mathcal{S}_\beta] \geq \frac{\alpha}{\delta} = \frac{1}{2}$$

and such that, for any $(\omega, H_2) \in \Omega_\beta$:

$$\Pr_{\tilde{c}}[(\omega, H_2', \tilde{c}) \in \mathcal{S}_\beta] \geq \delta - \alpha = \frac{\tilde{\varepsilon}}{4Q_2}.$$

With probability at least $\frac{\tilde{\varepsilon}}{2}$, the first execution (ω, H_2', c) of $\mathcal{A}$ simulated by $\mathcal{F}$ is successful and the index β belongs to the set I. Furthermore, in this case we have that $(\omega, H_2', c) \in \Omega_\beta$ with probability at least $1/2$. If we now repeat this simulated execution of $\mathcal{A}$ with fixed (ω, H_2') and randomly chosen $\tilde{c} \in \mathbb{Z}_q$, we know that $(\omega, H_2', \tilde{c}) \in \mathcal{S}_\beta$ and furthermore $\tilde{c} \neq c$ with probability at least $\frac{\tilde{\varepsilon}}{4Q_2}\left(1 - \frac{1}{q}\right)$.

Now consider the two successful executions of the attack, (ω, H_2', c) and $(\omega, H_2, \tilde{c})$, that the algorithm $\mathcal{F}$ has obtained by executing the attack $\mathcal{A}$. We denote by $(\mathcal{U}, m, \mathbf{v}, R_1, \ldots, R_{\hat{n}}, \psi)$ and $(\tilde{\mathcal{U}}, \tilde{m}, \tilde{\mathbf{v}}, \tilde{R}_1, \ldots, \tilde{R}_{\hat{n}}, \tilde{\psi})$, respectively, the two forged distributed ring signatures. Since the random tapes and H_1 are identical, and the answers of the random oracle H_2 are the same until the query $\mathcal{Q}_\beta = (\mathcal{U}, m, z_1, \ldots, z_{\hat{n}})$, we have in particular that $\tilde{\mathcal{U}} = \mathcal{U}$, $\tilde{\psi} = \psi$, $\tilde{m} = m$ and $\tilde{z}_i = z_i$, for $i = 1, \ldots, \hat{n}$.

Let us define the subset $B = \{U_i \in \mathcal{P} : \mathbf{v}\psi(U_i) = \tilde{\mathbf{v}}\psi(U_i)\}$. Since $\mathbf{v}\psi(D) = c \neq \tilde{c} = \tilde{\mathbf{v}}\psi(D)$ then B cannot be in Γ. Otherwise, if $B \in \Gamma$ then there would exist coefficients $\lambda_i \in \mathbb{Z}_q$ for users $U_i \in B$ satisfying $\psi(D) = \sum_{U_i \in B} \lambda_i \psi(U_i)$. This would imply

$$c = \mathbf{v}\psi(D) = \sum_{U_i \in B} \lambda_i \mathbf{v}\psi(U_i) = \sum_{U_i \in B} \lambda_i \tilde{\mathbf{v}}\psi(U_i) = \tilde{\mathbf{v}} \sum_{U_i \in B} \lambda_i \psi(U_i) = \tilde{\mathbf{v}}\psi(D) = \tilde{c},$$

a contradiction. Therefore we must have $B \notin \Gamma$, and so $\mathcal{P} - B \in \Gamma^* = cl(\mathcal{U})$; in other words, $A = \mathcal{P} - B = \{U_j \in \mathcal{P} : \mathbf{v}\psi(U_j) \neq \tilde{\mathbf{v}}\psi(U_j)\} \in cl(\mathcal{U})$.

By definition of successful forgery, there must exist some user $U_j \in A$, satisfying $c_j = \mathbf{v}\psi(U_j) \neq \tilde{\mathbf{v}}\psi(U_j) = \tilde{c}_j$, whose secret key has not been asked by the attacker $\mathcal{A}$. In this case, with probability $1 - \mu$ we have $d_j = 1$ and so $PK_j = \alpha_j bP$.

The equality $z_j = \tilde{z}_j$ becomes $e(R_j, P) \cdot e(Y, c_j PK_j) = e(\tilde{R}_j, P) \cdot e(Y, \tilde{c}_j PK_j)$. This is equivalent to

$$e(R_j - \tilde{R}_j, P) = e(Y, (\tilde{c}_j - c_j)PK_j) = e(aP, (\tilde{c}_j - c_j)\alpha_j bP) = e(a(\tilde{c}_j - c_j)\alpha_j bP, P).$$

This implies that $R_j - \tilde{R}_j = a(\tilde{c}_j - c_j)\alpha_j bP$. Therefore, the machine $\mathcal{F}$ obtains the solution of the given instance of the Computational Diffie-Hellman problem as

$$abP = \frac{1}{(\tilde{c}_j - c_j)\alpha_j}(R_j - \tilde{R}_j).$$

The inverse can be taken modulo q, since $\alpha_j \in \mathbb{Z}_q^*$ and $c_j \neq \tilde{c}_j$.

The total success probability ε' of the attack performed by $\mathcal{F}$ is

$$\varepsilon' \geq (1-\mu)\frac{\tilde{\varepsilon}}{2} \cdot \frac{1}{2} \cdot \frac{\tilde{\varepsilon}}{4Q_2} \cdot \frac{q-1}{q} \geq (1-\mu)\frac{\tilde{\varepsilon}^2}{16Q_2} \cdot \frac{q-1}{q} \geq$$

$$\geq (1-\mu)\frac{\varepsilon^2}{64Q_2} \cdot \frac{q-1}{q} \geq \frac{\varepsilon^2}{384Q_eQ_2} \cdot \frac{q-1}{q} \geq \frac{\varepsilon^2}{385Q_eQ_2}.$$

We have used the fact that $1-\mu = 1-(5/6)^{1/Q_e} \geq 1/6Q_e$ (applying Taylor's series methodology to the function $f(x) = 1-(1-x)^{1/Q_e}$ and then fixing $x = 1/6$). We have also assumed that $q \geq 385$, which happens if the security parameter k is $k \geq 9$.

The total execution time T' of the machine $\mathcal{F}$ consists of running twice the machine $\mathcal{A}$, simulating its environment. Summing up, we have that $T' \leq 2(T + Q_1 + Q_2 + (T_\psi + 2\hat{n}T_{bp})Q_s)$.

□

In the case $Q_e = 0$, the obtained result would be $\varepsilon' \geq \frac{\varepsilon^2}{33Q_2}$.

This last proposal, apart from being more efficient for some families of possible signing subsets, enjoys a better security reduction than the proposal in Section 5.3, since the factor $V_{Q_2,\hat{d}}$ does not appear in the relation between the probabilities ε' and ε. Roughly speaking, this is due to the fact that in the last proposal the hash function H_2 is called only once for each signature generation, whereas in the proposal in Section 5.3 we computed one hash value for each subset in the signing family.

5.4.2 Different Types of Keys

The distributed ring signature scheme proposed in Section 5.4 for ID-based scenarios can be extended to the case where users have different types of keys, of different lengths, etc. This fits in with a more real situation where each user generates his keys in an independent way. We consider three possibilities: RSA keys, Disc-Log keys and ID-based keys. The construction follows some ideas of the works [1, 94].

If a user U_i has RSA keys, then there exist a public key (n_i, e_i) such that user U_i knows the matching public key: the primes p_i and q_i such that $n_i = p_iq_i$, and the value d_i such that $d_ie_i = 1 \bmod \phi(n_i)$. There exists a public hash function $\hat{H}_i : \{0,1\}^* \to \mathbb{Z}_{n_i}^*$.

If a user U_i has a Disc-Log pair of keys, then there exists a pair of prime numbers p_i and q_i, and an element $g_i \in \mathbb{Z}_{p_i}$ such that $q_i | p_i - 1$ and g_i has order q_i in $\mathbb{Z}_{p_i}$. The secret key of user U_i is a value $x_i \in \mathbb{Z}_{q_i}^*$, whereas the matching public key is $y_i = g_i^{x_i} \bmod p_i$.

Finally, if a user has ID-based keys, this means that there exist an additive group $\mathbb{G}_{1,i}$, generated by some element P_i, and a multiplicative group $\mathbb{G}_{2,i}$, both with the same prime order q_i. There exist a bilinear pairing $e_i : \mathbb{G}_{1,i} \times \mathbb{G}_{1,i} \to \mathbb{G}_{2,i}$ and a public hash function $H_i : \{0,1\}^* \to \mathbb{G}_{1,i} - \{0\}$. User U_i is under the control of a master entity whose secret key is $x_i \in \mathbb{Z}_{q_i}$ and whose public key is $Y_i = x_i P \in \mathbb{G}_{1,i}$. The public key of a user U_i, with identity ID_i is $PK_i = H_i(ID_i)$, whereas his secret key is $SK_i = x_i PK_i$.

Distributed ring signature generation. Assume that a subset of users A want to compute an anonymous signature on behalf of a family $\mathcal{U}$ of possible signing subsets, taken over a set $\mathcal{P} = \{U_1, \ldots, U_n\}$ of n users.

Let k be twice the length of the largest q_i or n_i, among the n users in $\mathcal{P}$. Let $H : \{0,1\}^* \to \{0,1\}^k$ be a public hash function.

For simplicity, we will assume that both $cl(\mathcal{U})$ and $\Gamma = (cl(\mathcal{U}))^*$ are vector space access structures, and that there exist an integer r and a mapping $\psi : \mathcal{P} \cup \{D\} \to GF\left(2^k\right)^r$ such that $B \in \Gamma \Leftrightarrow \psi(D) \in \langle \{\psi(U_i)\}_{U_i \in B} \rangle$.

Since $\mathcal{U} = (cl(\mathcal{U}))_0$, we have that $A \in (cl(\mathcal{U}))_0$. This means that $\mathcal{P} - A \notin \Gamma$, and is maximal in the sense that $(\mathcal{P} - A) \cup \{U_j\} \in \Gamma$ for any user $U_j \in A$.

The signing users in A execute the following protocol to compute a valid distributed signature on a message $m \in \{0,1\}^*$:

1. They consider a basis of the subspace $\langle \psi(\mathcal{P} - A) \rangle$. This basis corresponds to some subset of users $C \subset \mathcal{P} - A$; that is, vectors in $\psi(C)$ are linearly independent and $\langle \psi(C) \rangle = \langle \psi(\mathcal{P} - A) \rangle$.

2. For every user $U_i \in C$, the signing users choose uniformly at random $c_i \in \{0,1\}^k$, and then proceed as follows:

   (a) If U_i has RSA keys, they choose uniformly at random $s_i \in \mathbb{Z}_{n_i}$; then they compute and broadcast the value $z_i = \hat{H}_i(c_i) + s_i^{e_i} \bmod n_i$.

   (b) If U_i has Disc-Log keys, they choose uniformly at random $s_i \in \mathbb{Z}_{q_i}$; then they compute and broadcast the value $z_i = g_i^{s_i} y_i^{c_i} \bmod p_i$.

   (c) If U_i has ID-based keys, they choose uniformly at random $s_i \in \mathbb{G}_{1,i}$; they compute and broadcast the value $z_i = e_i(s_i, P_i) \cdot e(Y_i, c_i PK_i)$.

3. For users $U_t \in (\mathcal{P} - A) - C$, we have that $\psi(U_t) = \sum_{U_i \in C} \lambda_{it} \psi(U_i)$, for some $\lambda_{it} \in GF(2^k)$, because $\psi(C)$ is a basis of $\langle \psi(\mathcal{P} - A) \rangle$. The signing users consider $c_t = \sum_{U_i \in C} \lambda_{it} c_i$, then they proceed as follows:

   (a) If U_t has RSA keys, they choose uniformly at random $s_t \in \mathbb{Z}_{n_t}$; then they compute and broadcast the value $z_t = \hat{H}_t(c_t) + s_t^{e_t} \bmod n_t$.

(b) If U_t has Disc-Log keys, they choose uniformly at random $s_t \in \mathbb{Z}_{q_t}$; then they compute and broadcast the value $z_t = g_t^{s_t} y_t^{c_t} \bmod p_t$.

(c) If U_t has ID-based keys, they choose uniformly at random $s_t \in \mathbb{G}_{1,t}$; they compute and broadcast the value $z_t = e_t(s_t, P_t) \cdot e(Y_t, c_t PK_t)$.

4. Each signing user $U_j \in A$ acts as follows:

(a) If U_j has RSA keys, he chooses uniformly at random $z_j \in \mathbb{Z}_{n_j}$ and computes this value.

(b) If U_j has Disc-Log keys, he chooses uniformly at random $a_j \in \mathbb{Z}_{q_j}$; then he computes and broadcasts the value $z_j = g_j^{a_j} \bmod p_j$.

(c) If U_j has ID-based keys, he chooses uniformly at random $T_j \in \mathbb{G}_{1,j}$; he computes and broadcasts the value $z_j = e(T_j, P_j)$.

5. The signing users compute then the value $c = H(\mathcal{U}, m, z_1, \ldots, z_n)$.

6. They choose uniformly at random one of the vectors $\mathbf{v} \in GF\left(2^k\right)^r$ that verifies:

(i) $\mathbf{v}\psi(D) = c$, and

(ii) $\mathbf{v}\psi(U_i) = c_i$, for all $U_i \in C$.

Note that this vector $\mathbf{v}$ exists because $C \notin \Gamma$ and so the vectors $\{\psi(D), \{\psi(U_i)\}_{U_i \in C}\}$ are linearly independent.

7. Every signing user $U_j \in A$ individually computes $c_j = \mathbf{v}\psi(U_j)$; then he proceeds as follows:

(a) If U_j has RSA keys, he computes and broadcasts the value $s_j = \left(z_j - \hat{H}_j(c_j)\right)^{d_j} \bmod n_j$.

(b) If U_j has Disc-Log keys, he computes and broadcasts the value $s_j = a_j - c_j x_j \bmod q_j$.

(c) If U_j has ID-based keys, he computes and broadcasts the value $s_j = T_j - c_j SK_j$.

Note that the rest of users in A can verify if the broadcast value s_j is consistent with the value z_j broadcast in step 4, by using the public key of user U_j. In this way, they detect dishonest users who try to boycott the process.

8. The resulting signature is $(\mathcal{U}, m, \mathbf{v}, s_1, \ldots, s_n, \psi)$.

Verification of a distributed ring signature. The recipient of the message first computes $c_i = \mathbf{v}\psi(U_i)$, for every user $U_i \in \mathcal{P}$ and then computes the following values:

(a) If U_i has RSA keys, compute $z_i = \hat{H}_i(c_i) + s_i^{e_i} \bmod n_i$.

(b) If U_i has Disc-Log keys, compute $z_i = g_i^{s_i} y_i^{c_i} \bmod p_i$.

(c) If U_i has ID-based keys, compute $z_i = e_i(s_i, P_i) \cdot e(Y_i, c_i PK_i)$.

The signature is valid if $\mathbf{v}\psi(D) = H(\mathcal{U}, m, z_1, \ldots, z_n)$.

The correctness of the scheme is easy to verify. With respect to anonymity and unforgeability, it can be proved using a combination of the techniques that appear in the proof of Theorem 5.3 and in the security proofs of the papers [1, 94]. To show that the scheme is unforgeable, one proves that if there would exist a successful adversary against it, then one could solve either the RSA problem, or the Discrete Logarithm problem, or the Computational Diffie-Hellman problem.

Conclusions

In this thesis we have proposed different schemes for some extensions of the concept of digital signature. These extensions have a characteristic in common: the origin of the signature is a collective of users, and not a single one. We analyze the security of all the schemes that we propose, considering the most powerful attackers against such schemes. In this way, we conclude that attacking our schemes is difficult, provided some problems from number theory are hard to solve.

In this section we summarize the main results of the thesis, and we state some related problems which remain unsolved and can therefore lead to future research.

In Chapter 1 we give the basic notions on digital signature schemes: the involved protocols, the requirements that must be satisfied and how to exactly analyze the security of a signature scheme. We exemplify these concepts with two well-known signature schemes: RSA and Schnorr. In particular, we show a proof of the security of RSA, which can be useful in order to familiarize the reader with the kind of security proofs that appear later in this thesis.

We also recall in that chapter the concept of secret sharing scheme, which is an essential tool for the design of cryptographic schemes involving many users who must jointly perform a task (e.g. signing a message), as it happens in some of the schemes that we propose throughout the thesis.

Distributed Signature Schemes

In this kind of schemes, the power of signing a message is shared among a fixed set of users, according to a fixed access structure, which is the family of subsets authorized to compute a valid signature on behalf of the whole set. For example, a threshold access structure contains all the subsets with at least t users, if t is the threshold.

Regarding the security of these schemes, an adversary structure must be defined, containing those subsets of users that an attacker can corrupt without compromising the security of the system. Most of the existing works

dealing with distributed signatures consider only the threshold case, although this one is quite particular and does not fit in with many real situations, where users have different computational resources or different probabilities to be attacked.

We consider a more general framework, and we try to find the necessary conditions to extend to this framework some proposals of distributed signature schemes which were designed for the threshold case. The most interesting case is that of RSA distributed signature schemes; we study the threshold proposal of Shoup [91], which is currently considered as the most complete one for this scenario. In Chapter 2, we give the algebraic and combinatorial conditions that must be satisfied in order to securely extend the proposal of Shoup to more general frameworks, with more general access and adversary structures. We believe that our analysis provides a better understanding of Shoup's work. An interesting way of research would be to look for new wide families of access and adversary structures satisfying the required conditions so that our scheme could be securely implemented.

In our extended scheme, as it happens in Shoup's scheme, the key generation phase must be performed by an external and trusted dealer, who generates the RSA keys and distributes shares of the secret key among the users. This fact can be avoided in the threshold case, as pointed out in [31, 47], by using some existing protocols which allow a set of users to jointly generate a RSA public key along with shares of the matching secret key [12, 49, 21]. But extending these protocols to general frameworks seems to be a hard task; in particular, one must solve the problem of generating shares of the product of secrets. This problem is securely solved, against active attacks, only for the threshold case. Therefore, the problem of designing a fully distributed signature scheme based on RSA, without a trusted dealer, remains open for the case of general access and adversary structures.

In Chapter 3, we relate the concept of distributed signature scheme with other two types of cryptographic schemes: metering schemes and distributed key distribution schemes.

We show that any non-interactive distributed signature scheme can be used to construct a metering scheme, which measures the number of interactions between a set of clients and a set of servers. On the other hand, we show how to construct from any deterministic distributed signature scheme a distributed key distribution scheme, where a set of servers provide information to users in conferences to obtain common conference keys. The constructed metering and key distribution schemes achieve the maximum level of security, provided the considered distributed signature schemes are secure.

In particular, we can employ the RSA distributed signature scheme that we design in Chapter 2 for general structures, extending the threshold

proposal of Shoup [91], because this scheme is both non-interactive and deterministic. In this way, we can consider general structures defined in the set of clients (in metering schemes) and in the set of servers (in distributed key distribution schemes), which makes the constructed schemes suitable for more realistic situations.

Ring Signature Schemes

The second block of contributions of this thesis is related to ring signature schemes. In such a scheme, a user signs a message in an anonymous way, by following this procedure: he chooses a set of users (or ring) which includes himself, and then he uses his own secret key and the public keys of all the members of the ring to compute a ring signature. The recipient of the signature is convinced that the signature has been computed by some of the members of the ring, but he has no information about who is the real signer.

In Chapter 4, we define a new family of ring signature schemes, that we call generic. For this family of schemes, we prove a security result, the Ring Forking Lemma, which is an extension of some results proved by Pointcheval and Stern in [83] for the case of standard signature schemes. Then we design two specific generic ring signature schemes, one in Chapter 4 for Discrete Logarithm scenarios, and another one in Chapter 5 for identity-based scenarios, where the public keys of users can be directly derived from their identities. We prove that these two schemes provide perfectly anonymous ring signatures. On the other hand, we can use the Ring Forking Lemma to prove that both schemes are unforgeable under the most powerful attacks; in the case of the first scheme, this is true assuming that the Discrete Logarithm problem is hard to solve, whereas in the second scheme, this is true assuming the difficulty of solving the Computational Diffie-Hellman problem. There are two negative points in these proofs. First, they are valid in the random oracle model, where some hash function is assumed to behave as a totally random function (which is a very strong, although very usual, assumption). And secondly, the reductions that we prove between forging a ring signature for our schemes and solving some difficult problem are very far from being tight. This means that the security result makes sense only when the length of the keys employed in the scheme is very large.

The first problem seems really hard to solve, because all the ring signature schemes proposed in the literature are proved secure in the random oracle model. With respect to the second one, it is inherent in generic ring signature schemes, due to the high number of hash evaluations (one for each member of the ring) that must be performed in each ring signature.

Finally, we combine the concepts of distributed signatures and ring

signatures. This combination leads to the concept of distributed ring signature schemes: some users want to cooperate to jointly sign a message, in an anonymous way. They choose a family of possible signing subsets, including the subset that they form, and then compute the signature by using their secret keys and the public keys of the rest of users. As a result, the recipient of the signature must be convinced that all the members of some of the subsets in the family have participated in the signature generation, but he must have no information about which subset of the family is actually the signing subset.

This kind of schemes had been already considered, but again only for threshold scenarios: if the signing users are t, then the family of possible signing subsets contains all the subsets with t users among the whole set of users. We propose schemes which run for any family of possibly signing subsets. In Chapter 4 we design a scheme for Discrete Logarithm scenarios, where all the users must have the same public parameters. In Chapter 5 we propose a scheme for identity-based scenarios, where the users must share the same public parameters, as well. These two schemes can be seen as distributed generic ring signature schemes, so we can apply an extension of the Ring Forking Lemma to prove the unforgeability of both schemes, assuming that the Discrete Logarithm problem and the Computational Diffie-Hellman problem, respectively, are hard to solve.

In the last section of Chapter 5 we propose a different distributed ring signature scheme, which is in some way more elegant that the two previous constructions, and is more efficient in some cases, where the considered family contains a lot of possibly signing subsets. Although we explain and analyze, for simplicity, the identity-based case where all the users have the same parameters, this scheme can be extended to more general cases where users have different types of keys, with different lengths, etc. For example, there can be some users with RSA keys, other users with Disc-Log keys, and other users with identity-based keys. The security result that we prove for this scheme is more tight that in the two previous proposals of distributed ring signature schemes.

Again, all the proofs of security of our distributed ring signature schemes are valid in the random oracle model, so a very interesting, although hard, future line of research could consist in finding (distributed) ring signature schemes whose security can be proved in the standard model, without the assumption that some hash functions behave as random oracles.

Bibliography

[1] M. Abe, M. Ohkubo and K. Suzuki. $1-\text{out}-\text{of}-n$ signatures from a variety of keys. *Proceedings of Asiacrypt'02*, LNCS **2501**, Springer-Verlag, pp. 415–432 (2002).

[2] M. Abe, M. Ohkubo and K. Suzuki. Efficient threshold signer-ambiguous signatures from variety of keys. *IEICE Transactions on Fundamentals*, Vol. **E87-A** (2), pp. 471–479 (2004).

[3] J. Algesheimer, J. Camenisch and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. *Proceedings of Crypto'02*, LNCS **2442**, Springer-Verlag, pp. 417–432 (2002).

[4] M. Bellare, A. Boldyreva and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 171–188 (2004).

[5] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 62–73 (1993).

[6] M. Bellare and P. Rogaway. The exact security of digital signatures - How to sign with RSA and Rabin. *Proceedings of Eurocrypt'96*, LNCS **1070**, Springer-Verlag, pp. 399–416 (1996).

[7] G.R. Blakley. Safeguarding cryptographic keys. *Proceedings of the National Computer Conference, American Federation of Information*, Processing Societies Proceedings **48**, pp. 313–317 (1979).

[8] C. Blundo, S. Cimato and B. Masucci. A note on optimal metering schemes. *Information Processing Letters*, Vol. **84** (6), pp. 319–326 (2002).

[9] C. Blundo, P. D'Arco, V. Daza and C. Padró. Bounds and constructions for unconditionally secure distributed key distribution schemes for general access structures. *Theoretical Computer Science*, **320**, pp. 269–291 (2004).

[10] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. *Proceedings of the PKC'03*, LNCS **2567**, Springer-Verlag, pp. 31–46 (2002).

[11] D. Boneh and X. Boyen. Short signatures without random oracles. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 56–73 (2004).

[12] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. *Proceedings of Crypto'97*, LNCS **1294**, Springer-Verlag, pp. 425–439 (1997).

[13] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, Vol. **32 (3)**, pp. 586–615 (2003). An extended abstract had appeared in the *Proceedings of Crypto'01*.

[14] D. Boneh, C. Gentry, B. Lynn and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *Proceedings of Eurocrypt'03*, LNCS **2656**, Springer-Verlag, pp. 416–432 (2003).

[15] C. Boyd. Digital multisignatures. *Cryptography and Coding*, Oxford University Press, pp. 241–246 (1989).

[16] E. Bresson, J. Stern and M. Szydlo. Threshold ring signatures for ad-hoc groups. *Proceedings of Crypto'02*, LNCS **2442**, Springer-Verlag, pp. 465–480 (2002).

[17] E.F. Brickell. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **9**, pp. 105–113 (1989).

[18] J. Camenisch. Efficient and generalized group signatures. *Proceedings of Eurocrypt'97*, LNCS **1233**, Springer-Verlag, pp. 465–479 (1997).

[19] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Adaptive security for threshold cryptosystems. *Proceedings of Crypto'99*, LNCS **1666**, Springer-Verlag, pp. 98–115 (1999).

[20] R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. *Proceedings of STOC'98*, pp. 209–218 (1998).

[21] D. Catalano, R. Gennaro and S. Halevi. Computing inverses over a shared secret modulus. *Proceedings of Eurocrypt'00*, LNCS **1807**, Springer-Verlag, pp. 190–206 (2000).

[22] D. Chaum and E. van Heyst. Group signatures. *Proceedings of Eurocrypt'91*, LNCS **547**, Springer-Verlag, pp. 257–265 (1991).

[23] D. Chaum and T.P. Pedersen. Wallet databases with observers. *Proceedings of Crypto'92*, LNCS **740**, Springer-Verlag, pp. 89–105 (1992).

[24] L. Chen, C. Kudla and K.G. Patterson. Concurrent signatures. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 287–305 (2004).

[25] S. Chow, L. Hui and S.M. Yiu. Identity based threshold ring signatures. *Proceedings of ICISC'04*, Springer-Verlag (to appear, 2005).

[26] J.-S. Coron. On the exact security of Full-Domain-Hash. *Proceedings of Crypto'00*, LNCS **1880**, Springer-Verlag, pp. 229–235 (2000).

[27] R. Cramer and I. Damgård. New generation of secure and practical RSA-based signatures. *Proceedings of Crypto'96*, LNCS **1109**, Springer-Verlag, pp. 173–185 (1996).

[28] R. Cramer, I. Damgård and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *Proceedings of Crypto'94*, LNCS **839**, Springer-Verlag, pp. 174–187 (1994).

[29] R. Cramer and S. Fehr. Optimal black-box secret sharing over arbitrary Abelian groups. *Proceedings of Crypto'02*, LNCS **2442**, Springer-Verlag, pp. 272–287 (2002).

[30] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 46–51 (1999).

[31] I. Damgård and M. Koprowski. Practical threshold RSA signatures without a trusted dealer. *Proceedings of Eurocrypt'01*, LNCS **2045**, Springer-Verlag, pp. 152–165 (2001).

[32] P. D'Arco and D.R. Stinson. On unconditionally secure robust distributed key distribution centers. *Proceedings of Asiacrypt'02*, LNCS **2501**, Springer-Verlag, pp. 346–363 (2002).

[33] V. Daza. On linear secret sharing schemes and distributed cryptographic protocols. Ph.D. Thesis, UPC, Barcelona (2004).

[34] V. Daza, J. Herranz, C. Padró and G. Sáez. A distributed and computationally secure key distribution scheme. *Proceedings of ISC'02*, LNCS **2433**, Springer-Verlag, pp. 342–356 (2002).

[35] V. Daza, J. Herranz and G. Sáez. Constructing general dynamic group key distribution schemes with decentralized user join. *Proceedings of ACISP'03*, LNCS **2727**, Springer-Verlag, pp. 464–475 (2003).

[36] V. Daza, J. Herranz and G. Sáez. Some protocols useful on the Internet from threshold signature schemes. *Proceedings of the Workshop TrustBus'03*, Ed. IEEE Computer Society, pp. 359–363 (2003).

[37] V. Daza, J. Herranz and G. Sáez. Protocols useful on the Internet from distributed signature schemes. *International Journal of Information Security*, Vol. **3** (2), Springer Verlag, pp. 61–69 (2004).

[38] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. *Proceedings of STOC'94*, pp. 522–533 (1994).

[39] Y. Desmedt and Y. Frankel. Threshold cryptosystems. *Proceedings of Crypto'89*, LNCS **435**, Springer-Verlag, pp. 307–315 (1989).

[40] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, Vol. **22** (6), pp. 644–654 (1976).

[41] Y. Dodis, A. Kiayias, A. Nicolosi and V. Shoup. Annonymous identification in ad hoc groups. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 609–626 (2004).

[42] C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. *Journal of Cryptology*, Vol. **11** (3), Springer-Verlag, pp. 187–208 (2000). A preliminary version appeared in the *Proceedings of Crypto'94*.

[43] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **31**, pp. 469–472 (1985).

[44] A. Fiat and M. Naor. Broadcast encryption. *Proceedings of Crypto'93*, LNCS **773**, Springer-Verlag, pp. 480–491 (1993).

[45] A. Fiat and A. Shamir. How to prove yourself: practical solutions of identification and signature problems. *Proceedings of Crypto'86*, LNCS **263**, Springer-Verlag, pp. 186–194 (1986).

[46] P.A. Fouque, G. Poupard and J. Stern. Sharing decryption in the context of voting or lotteries. *Proceedings of Financial Cryptography'00*, LNCS **1962**, Springer-Verlag, pp. 90–104 (2001).

[47] P.A. Fouque and J. Stern. Fully distributed threshold RSA under standard assumptions. *Proceedings of Asiacrypt'01*, LNCS **2248**, Springer-Verlag, pp. 310–330 (2001).

[48] Y. Frankel, P. Gemmell, P. MacKenzie and M. Yung. Proactive RSA. *Proceedings of Crypto'97*, LNCS **1294**, Springer-Verlag, pp. 440–454 (1997).

[49] Y. Frankel, P. MacKenzie and M. Yung. Robust efficient distributed RSA-key generation. *Proceedings of STOC'98*, pp. 663–672 (1998).

[50] D. Galindo. Personal communication (2005).

[51] R. Gennaro, S. Halevi and T. Rabin. Secure hash-and-sign signature without the random oracle. *Proceedings of Eurocrypt'99*, LNCS **1592**, Springer-Verlag, pp. 295–310 (1999).

[52] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust and efficient sharing of RSA functions. *Proceedings of Crypto'96*, LNCS **1109**, Springer-Verlag, pp. 157–172 (1996).

[53] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust Threshold DSS signatures. *Proceedings of Eurocrypt'96*, LNCS **1070**, Springer-Verlag, pp. 354–371 (1996).

[54] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Proceedings of Eurocrypt'99*, LNCS **1592**, Springer-Verlag, pp. 295–310 (1999).

[55] S. Goldwasser S and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, **28**, pp. 270–299 (1984).

[56] S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptative chosen-message attacks. *SIAM Journal of Computing*, **17 (2)**, pp. 281–308 (1988).

[57] J. Herranz. A formal proof of security of Zhang and Kim's ID-based ring signature scheme. *Proceedings of WOSIS'04*, INSTICC Press, pp. 63–72 (2004).

[58] J. Herranz, C. Padró and G. Sáez. Distributed RSA signature schemes for general access structures. *Proceedings of ISC'03*, LNCS **2851**, Springer-Verlag, pp. 122–136 (2003).

[59] J. Herranz and G. Sáez. Verifiable secret sharing for general access structures, with application to fully distributed proxy signatures. *Proceedings of Financial Cryptography Conference 2003*, LNCS **2742**, Springer-Verlag, pp. 286–302 (2003).

[60] J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. *Proceedings of Indocrypt'03*, LNCS **2904**, Springer-Verlag, pp. 266–279 (2003).

[61] J. Herranz and G. Sáez. Ring signature schemes for general access structures. *Proceedings of ESAS'04*, LNCS **3313**, Springer-Verlag, pp. 54–65 (2005).

[62] J. Herranz and G. Sáez. New ID-based ring signature schemes. *Proceedings of ICICS'04*, LNCS **3269**, Springer-Verlag, pp. 27–39 (2004).

[63] J. Herranz and G. Sáez. Revisiting fully distributed proxy signature schemes. *Proceedings of Indocrypt'04*, LNCS **3348**, Springer-Verlag, pp. 356–370 (2004).

[64] J. Herranz and G. Sáez. ID-based ring signatures for general families of signing subsets. Submitted (2005).

[65] J. Herranz and J.L. Villar. An unbalanced protocol for group key exchange. *Proceedings of TrustBus'04*, LNCS **3184**, Springer-Verlag, pp. 172–180 (2004).

[66] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk and M. Yung. Proactive public key and signature systems. *Proceedings of the 3rd ACM Conference on Computers and Communication Security*, pp. 100–110 (1996).

[67] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. *Proceedings of PODC'97*, pp. 25–34 (1997).

[68] K. Itakura and K. Nakamura. A public key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, **71**, pp. 1–8 (1983).

[69] W.A. Jackson and K.M. Martin. Geometric secret sharing schemes and their duals. *Designs, Codes and Cryptography*, Vol. **4**, pp. 83–95 (1994).

[70] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pp. 155–164 (2003).

[71] A. Lenstra and H. Lenstra. *The Development of the Number Field Sieve*, volume **1554** of *Lecture Notes in Mathematics*. Springer-Verlag (1993).

[72] A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham. Sequential aggregate signatures from trapdoor permutations. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 74–90 (2004).

[73] B. Masucci and D.R. Stinson. Metering schemes for general access structures. *Proceedings of ESORICS'00*, LNCS **1895**, Springer-Verlag, pp. 72–87 (2000).

[74] S. Micali, K. Ohta and L. Reyzin. Accountable-subgroup multisignatures. *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 245–254 (2001).

[75] M. Naor and B. Pinkas. Secure and efficient metering. *Proceedings of Eurocrypt'98*, LNCS **1403**, Springer-Verlag, pp. 576–590 (1998).

[76] M. Naor, B. Pinkas and O. Reingold. Distributed pseudo-random functions and KDCs. *Proceedings of Eurocrypt'99*, LNCS **1592**, Springer-Verlag, pp. 327–346 (1999).

[77] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, **21**, pp. 993–999 (1978).

[78] J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. *Proceedings of Crypto'02*, LNCS **2442**, Springer-Verlag, pp. 111–126 (2002).

[79] W. Ogata and K. Kurosawa. Provably secure metering scheme. *Proceedings of Asiacrypt'00*, LNCS **1976**, Springer-Verlag, pp. 388–398 (2000).

[80] K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. *Proceedings of Crypto'98*, LNCS **1462**, Springer-Verlag, pp. 354–369 (1998).

[81] C. Padró and G. Sáez. Secret sharing schemes with bipartite access structure. *IEEE Transactions on Information Theory*, Vol. **46** (7), pp. 2596–2604 (2000).

[82] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Proceedings of Eurocrypt'99*, LNCS **1592**, Springer-Verlag, pp. 223–238 (1999).

[83] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, Vol. **13** (3), Springer-Verlag, pp. 361–396 (2000).

[84] T. Rabin. A simplified approach to threshold and proactive RSA. *Proceedings of Crypto'98*, LNCS **1462**, Springer-Verlag, pp. 89–104 (1998).

[85] R.L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, **21**, pp. 120–126 (1978).

[86] R. Rivest, A. Shamir and Y. Tauman. How to leak a secret. *Proceedings of Asiacrypt'01*, LNCS **2248**, Springer-Verlag, pp. 552–565 (2001).

[87] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, Vol. **4**, Springer-Verlag, pp. 161–174 (1991).

[88] A. Shamir. How to share a secret. *Communications of the ACM*, **22**, pp. 612–613 (1979).

[89] A. Shamir. Identity-based cryptosystems and signature schemes. *Proceedings of Crypto'84*, LNCS **196**, Springer-Verlag, pp. 47–53 (1984).

[90] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, **27**, pp. 379–423 and 623–656 (1948).

[91] V. Shoup. Practical threshold signatures. *Proceedings of Eurocrypt'00*, LNCS **1807**, Springer-Verlag, pp. 207–220 (2000).

[92] G. J. Simmons, W. Jackson and K. Martin. The geometry of secret sharing schemes. *Bulletin of the ICA* **1** pp. 71–88 (1991).

[93] D.R. Stinson and R. Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. *Proceedings of ACISP'01*, LNCS **2119**, Springer-Verlag, pp. 417–434 (2001).

[94] J.K. Sui Liu, V.K. Wei and D.S. Wong. A separable threshold ring signature scheme. *Proceedings of ICISC'03*, LNCS **2971**, Springer-Verlag, pp. 12–26 (2004).

[95] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. *Proceedings of Asiacrypt'02*, LNCS **2501**, Springer-Verlag, pp. 533–547 (2002).

[96] The Pairing-Based Crypto Lounge. Web page maintained by Paulo Barreto:

`http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html`

Index