Revisiting Fully Distributed Proxy Signature Schemes

Javier Herranz and Germán Sáez

Dept. Matemàtica Aplicada IV, Universitat Politècnica de Catalunya
C. Jordi Girona, 1-3, Mòdul C3, Campus Nord, 08034 Barcelona, Spain
e-mail: {jherranz,german}@mat.upc.es

Abstract

In a proxy signature scheme, a potential signer delegates his capabilities to a proxy signer, who can sign documents on behalf of him. The recipient of the signature verifies both identities: that of the delegator and that of the proxy signer. There are many proposals of proxy signature schemes, but security of them has not been considered in a formal way until the appearance of [2].

If the entities which take part in a proxy signature scheme are formed by sets of participants, then we refer to it as a fully distributed proxy signature scheme [6].

In this work, we extend the security definitions introduced in [2] to the scenario of fully distributed proxy signature schemes, and we propose a specific scheme which is secure in this new model.

1 Introduction

Digital signature schemes provide authenticity, integrity and non-repudiation to digital communications. Sometimes, however, a user must sign messages during a certain period of time in which he is not able to do it. For example, if this user is in holidays or has technical problems with its computer.

Proxy signature schemes were introduced in [11] and give a solution to this problem. An original user delegates his signing capabilities into a different user, the proxy signer. In this delegation, some aspects such as the dates of validity or the kind of messages that the proxy will be able to sign on behalf of the original signer should be stated. Later, the proxy signer can sign messages which conform to the delegation, on behalf of the original user. The recipient of the signature must verify at the same time the delegation of the original signer and the authenticity of the proxy signer.

If the participants of the system are not individual users, but distributed entities, then we must consider *fully distributed* proxy signature schemes (introduced in [6]). The original entity is formed by a set of members, and if an authorized subset of them cooperate, then they can delegate the signing power of the whole entity into the proxy entity. Later, if some authorized subset of members of the proxy entity cooperate, then they can compute a proxy signature of a message on behalf of the original entity. Such schemes can be useful, for example, when the participants in the system are important companies, or a central office of a bank and the branch offices, etc.

Almost all the proxy signature schemes (either individual or distributed) that have been proposed until now lack a formal proof of security. This fact has led to many attacks on some of these schemes. Furthermore, this lack of formalism is not in compliance with the current techniques of public key cryptography, where the security of the protocols is formally proved (this is known as *provable security*). That is, both the capabilities and the goals of an adversary who tries to attack the cryptographic scheme must be clearly stated. Then, the security of the scheme should be proved by showing that a successful attack against it could be used as a part of another attack which would solve a computationally hard problem (discrete logarithm, integer factorization, etc.).

The first step in order to formalize individual proxy signature schemes has been taken in [2]. There, a formal model of security for this kind of schemes is given, along with some schemes which can be proved secure according to this model.

In this work, we extend the above-mentioned work, by giving a formal model of security for fully distributed proxy signature schemes. Then, we explain a distributed version of one of the schemes which are proposed and proved secure in [2]. We prove that this new scheme is secure in the security model for fully distributed proxy signature schemes.

Organization of the paper. In Section 2, we review some aspects of proxy signature schemes, including a specific scheme proposed in [2], as well as some distributed protocols that we will use later. In Section 3, we formally define what a fully distributed signature scheme is, and we give the natural security model for these schemes, derived from the model given in [2]. In Section 4, we propose a new fully distributed proxy signature scheme and prove its security. The work ends with some comments and conclusions in Section 5.

2 Preliminaries

The mathematical framework of the specific protocols that we are going to explain is the following. There are two large prime numbers p and q such that $q|p-1$. We consider an element $g \in \mathbb{Z}_p^*$ whose order is exactly q. We additionally need two hash functions H_1 and H_2 which map arbitrarily long strings of bits into $\mathbb{Z}_q$.

2.1 Proxy Signatures

Since its introduction by Mambo et al. [11], proxy signature schemes have been developed in many papers (for example [8, 10, 16, 9]). Most of the proposed schemes are based on discrete-logarithm type signature schemes, such as Schnorr's [14]. In this signature scheme, each signer has a secret key $x \in \mathbb{Z}_q^*$ and the corresponding public key $y = g^x \bmod p$. To sign a message M, this signer chooses a random value $k \in \mathbb{Z}_q^*$ and then he computes the values $r = g^k \bmod p$ and $s = k + xH_1(M, r) \bmod q$. The signature of the message M is the pair (r, s), and its correctness can be verified by checking the equation $g^s = ry^{H_1(M,r)} \bmod p$. We use the notation $(r,s) = Sch\_Sig(M, sk, H_1)$ to refer to an execution of this signature scheme for message M, with secret key sk and hash function H_1.

Schnorr's signature scheme has been shown [13] to achieve the highest level of security for signature schemes, which is existential unforgeability under chosen message attacks [5]. However, all the proposals of proxy signature schemes have lacked a formal security analysis. These schemes have been considered secure just until some attack against them has appeared (see [9, 18]).

The situation has changed since the appearance of a paper by Boldyreva et al. [2]. There, formal definitions on proxy signature schemes and their security are given, along with some specific schemes which are proved to be secure according to these definitions. We will briefly explain in Section 2.1 one of these schemes. We follow the notation of [2].

Let $Sig = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a standard signature scheme. That is, $\mathcal{G}$ is the parameter-generator, which takes as input a security parameter and outputs some global parameters of the scheme (in our scenario, the prime numbers p and q, etc.). The key-generator $\mathcal{K}$ takes as input the global parameters and outputs a secret-public key pair (sk, pk). The signing algorithm $\mathcal{S}$ takes as input a message and the secret key, and outputs a signature σ. And the verification algorithm $\mathcal{V}$ takes as input a message, a signature and a public key, and returns 1 (if the signature is valid) or 0 (if not).

A proxy signature scheme $Pro\_Sig = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ requires the presence of at least two users (user i delegates its signing capability into user j). The algorithms $\mathcal{G}$, $\mathcal{K}$, $\mathcal{S}$ and $\mathcal{V}$ are the same as explained above. The rest of protocols work as follows:

- $(\mathcal{D}, \mathcal{P})$ is a pair of (possibly interactive) algorithms, where user i delegates his signing capabilities into user j (proxy). The algorithm $\mathcal{D}$ takes as input the public keys pk_i and pk_j and the secret key sk_i of the delegator. The algorithm $\mathcal{P}$ takes as input the public keys pk_i and pk_j and the secret key sk_j of the proxy signer. As a result of this interaction, the proxy signer (user j) obtains a proxy secret key skp_{ij} that he will use to sign messages on behalf of user i.
- The protocol $\mathcal{PS}$ is the proxy signing algorithm, which takes as input a proxy secret key skp and a message M, and outputs a proxy signature $p\sigma$. This proxy signature includes the public key of user j, the proxy signer.
- The protocol $\mathcal{PV}$ verifies the correctness of a proxy signature. It takes as input a message, a proxy signature and the public key of the original signer, and outputs 1 or 0.
- The proxy identification algorithm $\mathcal{ID}$ takes as input a valid proxy signature and outputs the identity of the proxy signer.

The first formal security model for proxy signatures was proposed in [2]. Roughly speaking, such an scheme must be secure against the most powerful attack: first, an adversary who corrupts all the users in a system except one (say user 1); then, the adversary can request this user 1 to execute the different protocols of the scheme as many times as he wants, interacting with the corrupted users; finally, the adversary tries to forge a new valid proxy signature computed by user 1 on behalf of a corrupted user, or by a corrupted user j on behalf of user 1 (provided user 1 was not requested to delegate into user j, during the attack). A proxy signature scheme is secure if the probability of success of such an attack is negligible.

Triple Schnorr Proxy Signature Scheme

Now we explain the triple Schnorr proxy signature scheme which is proposed in [2]. We will refer to this scheme as $T\_Sch\_Pro\_Sig = (\mathcal{G}_{TS}, \mathcal{K}_{TS}, \mathcal{S}_{TS}, \mathcal{V}_{TS}, (\mathcal{D}_{TS}, \mathcal{P}_{TS}), \mathcal{PS}_{TS}, \mathcal{PV}_{TS}, \mathcal{ID}_{TS})$.

• The algorithms $(\mathcal{G}_{TS}, \mathcal{K}_{TS}, \mathcal{S}_{TS}, \mathcal{V}_{TS})$ are those of the standard Schnorr's signature scheme: $\mathcal{G}_{TS}$ generates the primes p and q, the element g and the hash functions H_1 and H_2. The algorithm $\mathcal{K}_{TS}$ generates secret key x and public key $y = g^x \bmod p$. The standard signing algorithm $\mathcal{S}_{TS}$ outputs a signature (r, s) on a message M. And $\mathcal{V}_{TS}$ verifies the correctness of the signatures. The main difference is that in order to sign a message M in a standard way, a user U_i must prepend a 1 to the message, and so apply $(r, s) = Sch\_Sig(1||M \ , \ x_i \ , \ H_1)$.

• The algorithms $(\mathcal{D}_{TS}, \mathcal{P}_{TS})$ are as follows. If a user U_i (with keys x_i and y_i) wants to delegate into a user U_j (with keys x_j and y_j), he creates a message ω which contains the information related to the delegation (identities of the original and proxy signers, dates of validity, which messages are allowed to be signed, etc.). Then user U_i computes the Schnorr signature $(r_i, s_i) = Sch\_Sig(0||y_i||y_j||\omega \ , \ x_i \ , \ H_1)$. User U_j verifies this signature and then computes his proxy secret key as $skp_{ij} = (y_i||y_j||\omega \ , \ r_i \ , \ d_{ij})$, where $d_{ij} = s_i + x_j H_1(0||y_i||y_j||\omega \ , \ r_i) \bmod q$. Note that the public key related to this secret key d_{ij} is $g^{d_{ij}} = r_i (y_i y_j)^{H_1(0||y_i||y_j||\omega \ , \ r_i)} \bmod p$.

• To compute a proxy signature on a message M, on behalf of user U_i, user U_j employs his proxy secret key d_{ij} and hash function H_2 to compute the Schnorr signature $(r, s) = Sch\_Sig(0||M||y_i||y_j||\omega||r_i \ , \ d_{ij} \ , \ H_2)$. The final proxy signature is $p\sigma = (\omega, r_i, y_j, (r, s))$.

• To verify the correctness of a proxy signature $p\sigma = (\omega, r_i, y_j, (r, s))$ on a message M, where the original signer has public key y_i, the recipient must check the following equation (Schnorr verification with public key $g^{d_{ij}}$ and hash function H_2):

$$g^s = r \left[r_i (y_i y_j)^{H_1(0||y_i||y_j||\omega \ , \ r_i)} \right]^{H_2(0||M||y_i||y_j||\omega||r_i \ , \ r)} \bmod p \ .$$

• The proxy identification algorithm takes as input a proxy signature $p\sigma = (\omega, r_i, y_j, (r, s))$ and returns the identity which corresponds to the public key y_j.

Theorem 1. *If the discrete logarithm problem is hard, then the proxy signature scheme $T\_Sch\_Pro\_Sig$ is secure in the random oracle model.*

See [2] for the security model and the proof of this theorem.

2.2 Joint Generation of Discrete Logarithm Keys

In distributed public key cryptography, the secret tasks (decrypting or signing) are not performed by single users, but by entities formed by many users. Let $E =$

$\{P^{(1)}, P^{(2)}, \ldots, P^{(n)}\}$ be a distributed entity formed by n participants. There is an *access structure* $\Gamma \subset 2^E$, which is formed by those subsets of participants which are authorized to perform the secret task. The access structure must be monotone increasing; that is, if $A_1 \in \Gamma$ is authorized, and $A_1 \subset A_2 \subset E$, then A_2 must be authorized, too.

The most usual strategy in distributed cryptography is to use *secret sharing schemes* (introduced in [1, 15]) to share the secret keys among the members of the entity. Some of these schemes do not need the presence of any trusted party (or dealer), and all the protocol can be performed by the members themselves. Linear secret sharing schemes, where the secret can be recovered as a linear combination of the shares from an authorized subset, are the most appropriate for being used as a component of distributed cryptographic protocols.

These distributed protocols must be secure in front of an attack of an adversary who corrupts a non-authorized subset of members of the entity. By corruption we mean that the adversary can see all the secret information of these users, and can control their behavior. The protocols are said to be *robust* if the dishonest members are always detected, and this fact does not avoid that the protocols finish in the correct way. In order to achieve robustness, *verifiable secret sharing schemes* [3, 12] are used.

A particular case of this kind of protocols is the joint generation of discrete logarithm keys. Each participant $P^{(\ell)} \in E$ obtains a secret value $x^{(\ell)} \in \mathbb{Z}_q$. These values $\{x^{(\ell)}\}_{P^{(\ell)} \in E}$ form a sharing of the secret key $x \in \mathbb{Z}_q$, according to some linear secret sharing scheme realizing the access structure Γ. The corresponding public key $y = g^x \bmod p$ is made public, along with other values (commitments) which ensure the robustness of the protocol. We refer to an execution of this protocol as

$$(y, \{x^{(\ell)}\}_{P^{(\ell)} \in E}) = Jo\_DL\_KG(E, \Gamma) \ .$$

The details of this protocol can be found in [4] for the threshold case (that is, the access structure is $\Gamma = \{A \subset E \ : \ |A| \geq t\}$, for some threshold t) and in [6] for the case of general access structures.

Fact 1. *The protocol Jo_DL_KG is simulatable.*

This means that, given an adversary who corrupts a non-authorized subset B of members, there exists an algorithm SIM_1 which takes as input a public key $y \in \langle g \rangle$ and outputs values which are indistinguishable from those that the adversary would see in a real execution of the protocol Jo_DL_KG which would give y as the resulting public key. Mainly, the algorithm SIM_1 must simulate all the information which is made public in the protocol, and the secret information of the dishonest members in B.

2.3 Distributed Schnorr Signature Protocol

In a distributed signature scheme, a set E of users share the secret key of a standard signature scheme. If an authorized subset of members collaborate, they can produce a valid signature on a message. The recipient can verify the correctness of this signature, but cannot know if it has been generated in a standard or a distributed way.

These schemes are said to be *unforgeable* if an adversary who corrupts a non-authorized subset of members is not able to obtain a valid message-signature pair, even if the protocol is previously executed for other messages that the adversary adaptively chooses. The signing protocol is robust if the dishonest participants are detected and furthermore the output of the protocol is always a valid signature.

In the case of Schnorr's signature scheme, the threshold version was proposed in [17], and the version for general access structures was proposed in [6]. We consider the more general case with any access structure Γ. The scheme starts with the joint key generation, that is, an execution of $(y, \{x^{(\ell)}\}_{P^{(\ell)} \in E}) = Jo\_DL\_KG(E, \Gamma)$, and then a protocol to jointly sign a message. We refer to an execution of this last protocol as:

$$(r, s) = Dist\_Sch\_Sig(E, \Gamma, M, \{x^{(\ell)}\}_{P^{(\ell)} \in E}, H_1) \ ,$$

meaning that participants of entity E use their secret shares $\{x^{(\ell)}\}_{P^{(\ell)} \in E}$ of the secret key x (which have been distributed using a linear secret sharing scheme which realizes the access structure Γ), to jointly compute a standard Schnorr signature (r, s) of message M with hash function H_1. This implies that $g^s = ry^{H_1(M,r)} \bmod p$.

Fact 2. *The protocol Dist_Sch_Sig is simulatable.*

This fact means that, given an adversary who corrupts a non-authorized subset $B \notin \Gamma$ of participants, there is an algorithm SIM_2 which runs as follows: it takes as input (M, r, s), where (r, s) is a valid Schnorr signature for message M, along with all the information obtained by the adversary in the execution of the corresponding $(y, \{x^{(\ell)}\}_{P^{(\ell)} \in E}) = Jo\_DL\_KG(E, \Gamma)$. The output values are indistinguishable from those (public and secret information of the corrupted members) that the adversary would see in a real execution of the protocol $Dist\_Sch\_Sig(E, \Gamma, M, \{x^{(\ell)}\}_{P^{(\ell)} \in E}, H)$.

3 Fully Distributed Proxy Signature Schemes

In addition to individual proxy signature schemes, some distributed (usually threshold) proxy signature schemes have been proposed in the last years [19, 8, 7]. In such schemes, a original signer delegates his capabilities into a proxy distributed entity. Members of an authorized subset of this entity can then jointly sign a message on behalf of the original signer. If the original signer is a distributed entity, too, then the proxy signature scheme is fully distributed [6].

As it has happened in the case of individual proxy signature schemes, no formal treatment of the security of distributed (and fully distributed) proxy signature schemes has been given until now. Some of the attacks which have been found against individual schemes are also applicable in the distributed versions of these schemes. For example, the attack explained in [9] against the individual proxy signature scheme in [10] can be also extended to an attack against the fully distributed proxy signature scheme in [6].

In this section, we formally define a fully distributed proxy signature scheme and the security requirements that such a scheme must satisfy. In some way, we extend the work done in [2] to the distributed scenario. Now there will be distributed entities $E_i = \{P_i^{(1)}, \ldots, P_i^{(n_i)}\}$ and $E_j = \{P_j^{(1)}, \ldots, P_j^{(n_j)}\}$ with their corresponding (monotone increasing) access structures $\Gamma_i \subset 2^{E_i}$ and $\Gamma_j \subset 2^{E_j}$. An authorized subset in Γ_i can delegate the signing capabilities of entity E_i into entity E_j. Then, an authorized subset in Γ_j can compute a proxy signature of entity E_j on behalf of entity E_i. Let us formalize the definition of all these protocols.

Let $Dist\_Sig = (\mathcal{G}, \mathcal{JKG}, \mathcal{DS}, \mathcal{V})$ be a distributed signature scheme. That is:

• The parameter-generator $\mathcal{G}$ takes as input a security parameter k and outputs some global (and public) parameters of the scheme (prime numbers, generators of the mathematical groups, etc.).

• The joint key generation protocol $\mathcal{JKG}$ is interactively performed by the members of each distributed entity E_i. It takes as input the global parameters and outputs a public key pk_i. Furthermore, each participant $P_i^{(\ell)} \in E_i$ obtains a secret share $sk_i^{(\ell)}$ of the secret key sk_i which matches with pk_i.

• The distributed signing algorithm $\mathcal{DS}$ takes as input a message and the secret shares of an authorized subset of members of the entity, and outputs a standard signature σ.

• The verification algorithm $\mathcal{V}$ takes as input a message, a signature and a public key, and returns 1 if the signature is valid, or 0 otherwise.

But a fully distributed proxy signature scheme $Dist\_Pro\_Sig = (\mathcal{G}, \mathcal{JKG}, \mathcal{DS}, \mathcal{V}, (\mathcal{DD}, \mathcal{DP}), \mathcal{DPS}, \mathcal{PV}, \mathcal{ID})$ requires also the following extra algorithms:

• $(\mathcal{DD}, \mathcal{DP})$ is a pair of (possibly interactive) algorithms. Entity E_i delegates its signing capabilities into entity E_j (proxy entity). The algorithm $\mathcal{DD}$ takes as input the public keys pk_i and pk_j and the shares of the secret key sk_i corresponding to some authorized subset of entity E_i. The algorithm $\mathcal{DP}$ takes as input the public keys pk_i and pk_j and the shares $\{sk_j^{(\ell)}\}_{P_j^{(\ell)} \in E_j}$ of the secret key of the proxy entity. As a result, each member $P_j^{(\ell)} \in E_j$ of the proxy entity obtains a share $skp_{ij}^{(\ell)}$ of the new proxy secret key skp_{ij}.

• The protocol $\mathcal{DPS}$ is the distributed proxy signing algorithm, which takes as input a message M and the shares of the proxy secret key skp_{ij} from some authorized subset of E_j, and outputs a proxy signature $p\sigma$. This proxy signature includes the public key pk_j of the proxy entity E_j.

• The protocol $\mathcal{PV}$ verifies the correctness of a proxy signature. It takes as input a message, a proxy signature and the public key of the delegator entity, and outputs 1 or 0.

• The proxy identification algorithm $\mathcal{ID}$ takes as input a valid proxy signature and outputs the identity of the proxy entity which has computed the signature.

3.1 Security Requirements

Intuitively, we want an adversary not to be able to forge a proxy or standard signature, even if he corrupts a non-authorized subset of each distributed entity which takes part in the system. In order to formally model this situation, we must consider a distributed entity E_1 and an adversary $\mathcal{DA}$ who corrupts a non-authorized subset $B_1 \subset E_1$, $B_1 \notin \Gamma_1$. The goal of the adversary is to forge a new proxy or standard signature realized by entity E_1 or on behalf of E_1.

Let $Dist\_Pro\_Sig = (\mathcal{G}, \mathcal{JKG}, \mathcal{DS}, \mathcal{V}, (\mathcal{DD}, \mathcal{DP}), \mathcal{DPS}, \mathcal{PV}, \mathcal{ID})$ be a fully distributed proxy signature scheme. We are going to consider an attack (or experiment) $\mathbf{D\_Exp}^{\mathcal{DA}}_{Dist\_Pro\_Sig}(k)$ performed by the adversary $\mathcal{DA}$ against the scheme $Dist\_Pro\_Sig$ under security parameter k.

The experiment starts with the generation of the global parameters. Adversary $\mathcal{DA}$ chooses the subset $B_1 \subset E_1$, such that $B_1 \notin \Gamma_1$, that he corrupts. Then the joint key generation protocol $\mathcal{JKG}$ is executed by the members of E_1 (here $\mathcal{DA}$ obtains the public key pk_1, all the information made public during the execution of the protocol, and the secret key shares $\{sk_1^{(b)}\}_{P_1^{(b)} \in B_1}$ of the corrupted participants).

The adversary initializes a counter $m = 1$, an empty set $Prox = \emptyset$ and an empty array $Array^{(1)}_{skp}$.

What can $\mathcal{DA}$ do?

During the experiment, the adversary $\mathcal{DA}$ is allowed to execute or ask for execution of the following actions:

1. $\mathcal{DA}$ **registers** E_i**.** $\mathcal{DA}$ can create and register a new distributed entity E_i, for $i = m + 1$. The adversary controls the behavior of all the members of this entity. These members run the protocol $\mathcal{JKG}$ which produces a public key pk_i and shares of the corresponding secret key sk_i. A new empty array $Array^{(i)}_{skp}$ is created. The counter is incremented, $m := m + 1$.

2. E_1 **delegates in** E_i**.** $\mathcal{DA}$ can interact with the whole entity E_1 running $\mathcal{DD}(pk_1, pk_i, \{sk_1^{(\ell)}\}_{P_1^{(\ell)} \in E_1})$, and himself playing the role of entity E_i, for some $i \in \{2, 3, \ldots, m\}$, running the protocol $\mathcal{DP}(pk_1, pk_i, \{sk_i^{(\ell)}\}_{P_i^{(\ell)} \in E_i})$. The set $Prox$ increases to $Prox \cup \{pk_i\}$ (this set contains the public keys of the entities in which entity E_1 delegates during the experiment).

3. E_i **delegates in** E_1**.** $\mathcal{DA}$ can interact with entity E_1 running $\mathcal{DP}(pk_i, pk_1, \{sk_1^{(\ell)}\}_{P_1^{(\ell)} \in E_1})$, and himself playing the role of entity E_i, for some $i \in \{2, 3, \ldots, m\}$, running the protocol $\mathcal{DD}(pk_i, pk_1, \{sk_i^{(\ell)}\}_{P_i^{(\ell)} \in E_i})$. As a result, each participant $P_1^{(\ell)}$ of entity E_1 will obtain a share $skp_{i1}^{(\ell)}$ of the new proxy secret key. Note that the adversary knows the shares of the corrupted players in B_1. The whole set of shares $SKP_{i1} = \{skp_{i1}^{(\ell)}\}_{P_i^{(\ell)} \in E_i}$ is stored in the first available position of $Array^{(i)}_{skp}$. This array will therefore contain all the secret proxy keys corresponding to delegations of entity E_i into entity E_1. Obviously, the adversary has not full access to these arrays (he only knows the shares of the corrupted players in B_1).

4. E_1 **delegates in** E_1**.** $\mathcal{DA}$ can request that entity E_1 run the delegation protocol with itself. The adversary will see all the public information and the private information held by the corrupted players. As in Action 3, the shares of the resulting secret proxy key, SKP_{11}, are stored in the first available position of $Array^{(1)}_{skp}$.

5. **Standard distributed signature by E_1.** $\mathcal{DA}$ can ask the members of E_1 for executing the protocol $\mathcal{DS}$ for signing the message M that he chooses. He obtains all public information and private information of the dishonest players (in B_1).

6. **Distributed proxy signature by E_1 on behalf of E_i.** $\mathcal{DA}$ can request that members of E_1 use the shares of some of the proxy secret keys obtained from a delegation of entity E_i (Action 3), and which are stored in some position of $Array_{skp}^{(i)}$, to execute the protocol $\mathcal{DPS}$ with a message M that he chooses. Again, he obtains the signature, all the broadcast information and the private information of the corrupted players.

When is $\mathcal{DA}$ successful?

Once the adversary has done these actions as many times as he wants, he eventually outputs a forgery of a standard signature (M, σ) or of a proxy signature $(M, p\sigma, pk)$.

• If (M, σ) satisfies $\mathcal{V}(M, \sigma, pk_1) = 1$, and M was not queried by $\mathcal{DA}$ to be signed as a standard distributed signature by entity E_1 (action 5), then the output of the experiment is 1 (successful forgery of a standard signature by entity E_1).

• If $(M, p\sigma, pk)$ satisfies $pk = pk_i$ for some $i \in \{1, 2, \dots, m\}$, and $\mathcal{PV}(M, p\sigma, pk_i) = 1$, and $\mathcal{ID}(p\sigma) = pk_1$, and message M was not queried to be signed by E_1 on behalf of E_i (action 6), then the output of the experiment is 1 (successful forgery of a proxy signature by entity E_1 on behalf of some entity E_i).

• If $(M, p\sigma, pk)$ satisfies $pk = pk_1$, and $\mathcal{PV}(M, p\sigma, pk_1) = 1$, and $\mathcal{ID}(p\sigma) \notin Prox \cup \{pk_1\}$, then the output of the experiment is 1 (successful forgery of a proxy signature by some entity $E_i \neq E_1$, which was not designated by entity E_1 during the experiment, on behalf of entity E_1).

Otherwise, the output of the experiment $\mathbf{D\_Exp}_{Dist\_Pro\_Sig}^{\mathcal{DA}}(k)$ is 0. We define the probability of success of the adversary $\mathcal{DA}$ as the probability that the output of the experiment is 1. That is:

$$\mathbf{Succ}_{Dist\_Pro\_Sig}^{\mathcal{DA}}(k) = \Pr\left[\mathbf{D\_Exp}_{Dist\_Pro\_Sig}^{\mathcal{DA}}(k) \ = \ 1\right] \ .$$

Definition 1. *We say that a fully distributed proxy signature scheme $Dist\_Pro\_Sig$ is secure if, for all polynomial time adversary $\mathcal{DA}$, we have that $\mathbf{Succ}_{Dist\_Pro\_Sig}^{\mathcal{DA}}(k)$ is negligible in the security parameter k.*

We recall that a function $f(k)$ is negligible in k if for all polynomial $p()$, there exists $k_p \in \mathbb{N}$ such that $f(k) \leq \frac{1}{p(k)}$, for all $k \geq k_p$.

This security model for fully distributed proxy signature schemes is the natural extension of the security model defined in [2] for individual proxy signature schemes (there, the adversary attacks a user; here, the adversary attacks an entity where he has corrupted some of its members).

4 A New Scheme

We now explain the natural way of fully distributing the triple Schnorr proxy signature scheme given in [2]. We follow the notation introduced in Section 3. We denote the fully distributed triple Schnorr proxy signature scheme by $T\_Sch\_Dist\_Pro\_Sig = (\mathcal{G}_{TS}, \mathcal{JKG}_{TS}, \mathcal{DS}_{TS}, \mathcal{V}_{TS}, (\mathcal{DD}_{TS}, \mathcal{DP}_{TS}), \mathcal{DPS}_{TS}, \mathcal{PV}_{TS}, \mathcal{ID}_{TS})$. The different protocols work as follows:

• The parameter generator $\mathcal{G}_{TS}$ takes as input a security parameter k and outputs the prime numbers p and q such that $q|p-1$, an element g with order q in $\mathbb{Z}_p^*$, and two hash functions $H_1,\ H_2 : \{0,1\}^* \to \mathbb{Z}_q$.

• The key generator $\mathcal{K}_{TS}$ for an entity E_i with access structure Γ_i consists of running the protocol $(y_i, \{x_i^{(\ell)}\}_{P_i^{(\ell)} \in E_i}) = Jo\_DL\_KG(E_i, \Gamma_i)$ for joint generating a public key and shares of the matching secret key (see Section 2.2).

• The distributed signature protocol $\mathcal{DS}_{TS}$ applied to a message M consists of prepending a 1 to the message and executing the protocol of joint computation of a Schnorr signature $(r,s) = Dist\_Sch\_Sig(E_i, \Gamma_i,\ 1||M\ , \{x_i^{(\ell)}\}_{P_i^{(\ell)} \in E_i}, H_1)$ (see Section 2.3).

• The verification protocol $\mathcal{V}_{TS}$ verifies that (r,s) is a valid Schnorr signature for message $1||M$.

• The protocols $(\mathcal{DD}_{TS}, \mathcal{DP}_{TS})$ are as follows. If an entity E_i (with keys x_i and y_i, where x_i is shared) wants to delegate into an entity E_j (with keys x_j and y_j, where x_j is shared), members of entity E_i create a message ω which contains the information related to the delegation. Then members of E_i jointly compute the Schnorr signature

$$(r_i, s_i) = Dist\_Sch\_Sig(E_i, \Gamma_i,\ 0||y_i||y_j||\omega\ , \{x_i^{(\ell)}\}_{P_i^{(\ell)} \in E_i}, H_1)\ .$$

Each member $P_j^{(\ell)}$ of entity E_j verifies this signature and then computes his share of the proxy secret key as $skp_{ij}^{(\ell)} = (y_i||y_j||\omega\ ,\ r_i\ ,\ d_{ij}^{(\ell)})$, where

$$d_{ij}^{(\ell)} = s_i + x_j^{(\ell)} H_1(0||y_i||y_j||\omega\ ,\ r_i) \bmod q\ .$$

Note that it results in a sharing of the proxy secret key $d_{ij} = s_i + x_j H_1(0||y_i||y_j||\omega\ ,\ r_i) \bmod q$, and that the public key related to this secret key d_{ij} is $g^{d_{ij}} = r_i (y_i y_j)^{H_1(0||y_i||y_j||\omega\ ,\ r_i)} \bmod p$.

• The protocol $\mathcal{DPS}_{TS}$ work as follows: to jointly compute a proxy signature on a message M, on behalf of entity E_i, members of the entity E_j employ their shares of the proxy secret key d_{ij} and the hash function H_2 to compute the Schnorr signature

$$(r,s) = Dist\_Sch\_Sig(E_j, \Gamma_j,\ 0||M||y_i||y_j||\omega||r_i\ , \{d_{ij}^{(\ell)}\}_{P_j^{(\ell)} \in E_j}, H_2)\ .$$

The final proxy signature is $p\sigma = (\omega, r_i, y_j, (r,s))$.

• To verify (protocol $\mathcal{PV}_{TS}$) the correctness of a proxy signature $p\sigma = (\omega, r_i, y_j, (r,s))$ on a message M, where the original signer entity has public key y_i, the recipient must check the following equation (Schnorr verification with public key $g^{d_{ij}}$ and hash function H_2):

$$g^s = r \left[r_i (y_i y_j)^{H_1(0||y_i||y_j||\omega\ ,\ r_i)} \right]^{H_2(0||M||y_i||y_j||\omega||r_i\ ,\ r)} \bmod p\ .$$

• The proxy identification algorithm $\mathcal{ID}_{TS}$ takes as input a proxy signature $p\sigma = (\omega, r_i, y_j, (r,s))$ and returns the entity whose public key is y_j.

4.1 Security Analysis

The following theorem asserts that this fully distributed proxy signature scheme is secure in the model introduced in Section 3.1. We prove this fact by reduction to the security of the individual triple Schnorr scheme in the security model for individual proxy signatures (see Section 2.1). The details of the proof are given in the Appendix.

Theorem 2. *If the discrete logarithm problem is hard, then the fully distributed proxy signature scheme $T\_Sch\_Dist\_Pro\_Sig$ is secure in the random oracle model.*

5 Conclusion

In this work we have taken one more step in the formalization of proxy signature schemes, by giving a security model for fully distributed proxy signature schemes. This new model is the natural extension of the security model introduced in [2] for individual proxy signature schemes.

Furthermore, we present a fully distributed proxy signature scheme which is proved to be secure in the new model. The scheme is the distributed version of a individual scheme proposed in [2]. They are based on Schnorr's signature scheme, and their security relies on the hardness of the discrete logarithm problem.

There are a lot of proxy signature schemes (individual or distributed) in the literature whose security has not been formally proved yet. We think that the two abovementioned models, the one in [2] for individual proxy signature schemes and the one in this work for distributed schemes, should be considered from now on, in order to prove the security of both existing and future schemes.

References

[1] G.R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS'79*, pp. 313–317 (1979).

[2] A. Boldyreva, A. Palacio and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. Preprint available at `http://eprint.iacr.org/2003/096/`

[3] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. *Proceedings of FOCS'87*, IEEE Press, pp. 427–437 (1987).

[4] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Proceedings of Eurocrypt'99*, LNCS **1592**, pp. 295–310 (1999).

[5] 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).

[6] 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**, pp. 286–302 (2003).

[7] M. Hwang, I. Lin and E.J. Lu. A secure nonrepudiable threshold proxy signature scheme with known signers. *International Journal of Informatica*, vol. **11**, no. 2, pp. 1–8, (2000).

[8] S. Kim, S. Park and D. Won. Proxy signatures, revisited. *Proceedings of ICISC'97*, pp. 223–232 (1997).

[9] J.-Y. Lee, J. H. Cheon and S. Kim. An analysis of proxy signatures: is a secure channel necessary? *Proceedings of CT-RSA Conference 2003*, LNCS **2612**, pp. 68–79 (2003).

[10] B. Lee, H. Kim and K. Kim. Strong proxy signature and its applications. *Proceedings of SCIS'01*, Vol. 2/2, pp. 603–608 (2001).

[11] M. Mambo, K. Usuda and E. Okamoto. Proxy signatures: delegation of the power to sign messages. *IEICE Transactions Fundamentals*, Vol. E79-A, No. **9**, pp. 1338–1353 (1996).

[12] T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Proceedings of Crypto'91*, LNCS **576**, pp. 129–140 (1991).

[13] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, Vol. **13**, Num. **3**, pp. 361–396 (2000).

[14] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, Vol. **4**, pp. 161–174 (1991).

[15] A. Shamir. How to share a secret. *Communications of the ACM*, No. **22**, pp. 612–613 (1979).

[16] Z. Shao. Proxy signature schemes based on factoring. *Information Processing Letters*, No. **85**, pp. 137–143 (2003).

[17] 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).

[18] H.-M. Sun and B.-T. Hsieh. On the security of some proxy signature schemes. Preprint available at `http://eprint.iacr.org/2003/068/` (2003).

[19] K. Zhang. Threshold proxy signature scheme. *Proceedings of the 1997 Information Security Workshop, Japan*, pp. 191–197 (1997).

Appendix: Proof of Theorem 2

Let us assume there exists an adversary $\mathcal{DA}$ against this fully distributed proxy signature scheme such that its success probability $\mathbf{Succ}^{\mathcal{DA}}_{T\_Sch\_Dist\_Pro\_Sig}(k)$ is non-negligible. We can then construct an adversary $\mathcal{A}$ and an experiment $\mathbf{Exp}^{\mathcal{A}}_{T\_Sch\_Pro\_Sig}(k)$ against the individual scheme $T\_Sch\_Pro\_Sig$, following the definition and notation of [2], as follows:

The public parameters (p, q, g, H_1, H_2) are generated, along with a public and secret key pair (x_1, y_1) for user U_1, where $y_1 = g^{x_1} \bmod p$. A counter m is initialized to 1, an empty set $Prox$ and an empty array $Array^{(1)}_{skp}$ are created. The value y_1 is given to the adversary $\mathcal{A}$.

Now $\mathcal{A}$ executes SIM_1 (see Fact 1) with input y_1 and the information related to the adversary $\mathcal{DA}$ (entity E_1, access structure Γ_1, set B_1 of corrupted players...). Therefore $\mathcal{A}$ obtains values which are indistinguishable from those that $\mathcal{DA}$ would have seen in a real execution of $K_{TS} = Jo\_DL\_KG$ which would have produced y_1 as the resulting public key.

Then, $\mathcal{A}$ requests $\mathcal{DA}$ to run the experiment $\mathbf{D\_Exp}^{\mathcal{DA}}_{T\_Sch\_Dist\_Pro\_Sig}(k)$. For that, $\mathcal{A}$ must provide $\mathcal{DA}$ with the information obtained from SIM_1 in the previous step, and also simulate the real environment of $\mathcal{DA}$ during the experiment $\mathbf{D\_Exp}^{\mathcal{DA}}_{T\_Sch\_Dist\_Pro\_Sig}(k)$, replying all its queries and actions:

1. If $\mathcal{DA}$ wants to register a new entity E_i, where $i = m + 1$, then $\mathcal{A}$ registers a new user U_i (he is allowed to do so, see the security model in [2]), obtaining a pair (x_i, y_i). Then $\mathcal{A}$ executes SIM_1 with input y_i and gives the outputs to $\mathcal{DA}$. The counter is incremented, $m := m + 1$, and an empty array $Array^{(i)}_{skp}$ is created.

2. When $\mathcal{DA}$ requires entity E_1 to delegate into entity E_i (with delegation message ω), then $\mathcal{A}$ requires user U_1 to delegate into user U_i. Therefore, $\mathcal{A}$ obtains a valid Schnorr signature, under public key y_1 and hash function H_1, of the message $0||y_1||y_i||\omega$. Then $\mathcal{A}$ executes SIM_2 (see Fact 2) with input this pair message-signature and the information obtained in the first execution (with input y_1) of SIM_1. The output of SIM_2 perfectly simulates the view of $\mathcal{DA}$ during these queries. The set $Prox$ increases to $Prox \cup \{pk_i\}$.

3. When $\mathcal{DA}$ requires some entity E_i to delegate into entity E_1, then $\mathcal{A}$ requires user U_i to delegate into user U_1. If the delegation message is ω, then $\mathcal{A}$ obtains a valid Schnorr signature (r_i, s_i), under public key y_i and hash function H_1, of the message $0||y_i||y_1||\omega$. Now $\mathcal{A}$ executes SIM_2 for this pair message-signature. Furthermore, for all corrupted player $P_1^{(b)} \in B_1 \subset E_1$, $\mathcal{A}$ computes the corresponding share
$$d_{i1}^{(b)} = s_i + x_1^{(b)} H_1(0||y_i||y_1||\omega \ , \ r_i) \bmod q$$
of the new proxy secret key, where $x_1^{(b)}$ are the shares of the secret key of entity E_1, obtained in the first execution of SIM_1. In this way, $\mathcal{A}$ simulates in a perfect way the view of $\mathcal{DA}$ for these queries. The first available position of $Array^{(i)}_{skp}$ is filled with these shares $d_{i1}^{(b)}$ and other random shares for the non-corrupted

players (since $\mathcal{DA}$ has not full access to these arrays, it is not important what is put in the places corresponding to the non-corrupted players).

4. When $\mathcal{DA}$ requires entity E_1 to delegate into itself, $\mathcal{A}$ requires user U_1 to designate himself. If the delegation message is ω, then $\mathcal{A}$ obtains a valid Schnorr signature (r_1, s_1), under public key y_1 and hash function H_1, of message $0||y_1||y_1||\omega$. Then $\mathcal{A}$ executes SIM_2 for this pair message-signature. Again, for all corrupted players $P_1^{(b)} \in B_1 \subset E_1$, $\mathcal{A}$ computes the corresponding share

$$d_{11}^{(b)} = s_1 + x_1^{(b)} H_1(0||y_1||y_1||\omega\ ,\ r_1) \bmod q$$

of the new proxy secret key. These values and the output of SIM_2 perfectly simulate the view of $\mathcal{DA}$ during these queries. The next available position of $Array_{skp}^{(1)}$ is filled with the computed shares for the corrupted players in B_1 and with random numbers for the non-corrupted players.

5. When $\mathcal{DA}$ requires E_1 to compute a distributed Schnorr signature on a message M, $\mathcal{A}$ queries user U_1 to compute a Schnorr signature on message M (the same message and the same public key). The resulting signature and the message are given as inputs to SIM_2. The outputs simulate the view of $\mathcal{DA}$ during the execution of the distributed Schnorr signature protocol.

6. If $\mathcal{DA}$ requires entity E_1 to compute a proxy signature of message M on behalf of entity E_i (which has previously delegated into E_i by publishing a signature (r_i, s_i) on a delegation message ω), then $\mathcal{A}$ requires user U_1 to compute a proxy signature of message M on behalf of user U_i (who, of course, has previously delegated into U_1 by publishing exactly the signature (r_i, s_i) on the delegation message ω). The result is a valid Schnorr signature (r, s), of message $0||M||y_i||y_1||\omega||r_i$, under hash function H_2 and public key

$$r_i (y_i y_1)^{H_1(0||y_i||y_1||\omega\ ,\ r_i)} \ .$$

Then $\mathcal{A}$ can execute SIM_2 with input this message-signature pair, along with other information which $\mathcal{A}$ had obtained when E_i performed the considered delegation on E_1 (for example, the shares $d_{i1}^{(\ell)}$ of the corresponding secret proxy key). The output of SIM_2 simulates the view of $\mathcal{DA}$ in this phase of the experiment.

By assumption, and since $\mathcal{A}$ perfectly simulates the environment of $\mathcal{DA}$, one of the following facts happens with non-negligible probability:

• $\mathcal{DA}$ outputs $(M, (r, s))$ satisfying $\mathcal{V}_{TS}(M, (r, s), y_1) = 1$, such that M was not queried by $\mathcal{DA}$ to be signed as a standard distributed signature by entity E_1 (action 5). Therefore, $\mathcal{A}$ did not query user U_1 to sign message M in the standard way, either, and so the output of the experiment $\mathbf{Exp}^{\mathcal{A}}_{T\_Sch\_Pro\_Sig}(k)$, performed by $\mathcal{A}$, would be 1 (successful forgery of a standard signature).

• $\mathcal{DA}$ outputs a forgery of a proxy signature by entity E_1, on behalf of entity E_i, of a message that was not queried by $\mathcal{DA}$ to be signed by E_1 on behalf of E_i during the experiment. Therefore, $\mathcal{A}$ obtains a forgery of a proxy signature by U_1, on behalf of U_i, of a message that $\mathcal{A}$ did not query user U_1 to sign on behalf of U_i. That is, the output of $\mathbf{Exp}^{\mathcal{A}}_{T\_Sch\_Pro\_Sig}(k)$ would be again 1.

• $\mathcal{DA}$ outputs a forgery of a proxy signature by some entity $E_i \neq E_1$ (which was not designated by entity E_1 at any time during the experiment) on behalf of entity E_1. Analogously, user U_i was never designated by user U_1 during the experiment performed by $\mathcal{A}$, but $\mathcal{A}$ obtains a valid proxy signature by user $U_i \neq U_1$ on behalf of user U_1. Thus, the output of $\mathbf{Exp}^{\mathcal{A}}_{T\_Sch\_Pro\_Sig}(k)$ would be 1.

Summing up, we have that

$$\mathbf{Succ}^{\mathcal{A}}_{T\_Sch\_Pro\_Sig}(k) \;\geq\; \mathbf{Succ}^{\mathcal{DA}}_{T\_Sch\_Dist\_Pro\_Sig}(k) \;.$$

But we are assuming that $\mathbf{Succ}^{\mathcal{DA}}_{T\_Sch\_Dist\_Pro\_Sig}(k)$ is non-negligible. So we could conclude that $\mathbf{Succ}^{\mathcal{A}}_{T\_Sch\_Pro\_Sig}(k)$ is also non-negligible, which contradicts Theorem 1. Therefore, we prove that there can not exist an adversary $\mathcal{DA}$ with non-negligible probability of successfully attacking the scheme $T\_Sch\_Dist\_Pro\_Sig$, and so this scheme is provably secure (in the random oracle model, as it is the individual triple Schnorr scheme). This completes the proof.