A Distributed and Computationally Secure Key Distribution Scheme

Vanesa Daza, Javier Herranz, Carles Padró 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: {vdaza,jherranz,matcpl,german}@mat.upc.es

Abstract

In [16], Naor, Pinkas and Reingold introduced schemes in which some groups of servers distribute keys among a set of users in a distributed way. They gave some specific proposals both in the unconditional and in the computational security framework. Their computationally secure scheme is based on the Decisional Diffie-Hellman Assumption. This model assumes secure communication between users and servers. Furthermore it requires users to do some expensive computations in order to obtain a key.

In this paper we modify the model introduced in [16], requiring authenticated channels instead of assuming the existence of secure channels. Our model makes the user's computations easier, because most computations of the protocol are carried out by servers, keeping to a more realistic situation. We propose a basic scheme, that makes use of ElGamal cryptosystem, and that fits in with this model in the case of a passive adversary. We then add zero-knowledge proofs and verifiable secret sharing to prevent from the action of an active adversary. We consider general structures (not only the threshold ones) for those subsets of servers that can provide a key to a user and for those tolerated subsets of servers that can be corrupted by the adversary. We find necessary combinatorial conditions on these structures in order to provide security to our scheme.

1 Introduction

When a group of users wish to communicate securely over insecure channels, either symmetric or public key cryptosystems are used. Public key schemes present some drawbacks both from the communication and the computational point of view. On the other hand, when symmetric algorithms are considered in order to solve this problem, the question is then how to set up an efficient protocol to give each group of users a common key.

The solution of setting a server responsible of the distribution and management of the secret keys was introduced in [17] by Needham and Schroeder. This idea of a Key Distribution Center was formalized in [3]. This model presents some drawbacks. A single server that is in charge of the distribution of keys to a group of users presents some weak points. It is a possible bottleneck of the system and it must be trusted. Among the proposed solutions in order to remove this drawback, the use of Distributed Key Distribution Centers is one of the most accepted.

The model in which the task of a single server is distributed among a set of servers, the Distributed Key Distribution Center model, was introduced in [16]. Schemes fitting this model are called Distributed Key Distribution Schemes. Some specific realizations were proposed both in the information theoretic model, where no limits in the computational power of an adversary are assumed, and in the computationally secure framework, where the computational capability of an adversary is bounded.

With regard to the information theoretic point of view some studies have been done since then: in [5] an exhaustive study on the amount of information needed to set up and manage such a system was presented. They considered *threshold* access structures, that is, those sets of servers that are authorized to provide keys have at least t servers (t is the threshold). Afterwards, in [6] it was extended to a model considering *general* access structures. Moreover, a relation between distributed key distribution schemes and secret sharing schemes was shown.

However, in this paper we focus on computationally secure distributed key distribution schemes. Previously, in [16] such a scheme was proposed, based on the Decisional Diffie-Hellman Assumption [11], as an application of their scheme for evaluating a pseudo-random function in a distributed way. This scheme assumes secure communication between users and servers. Moreover, it requires users to do some expensive computations in order to get a key.

We propose a new model for describing distributed key distribution schemes, where the secure channel assumption is weakened to an authenticated channel assumption. We provide an explicit construction realizing this model. The use of the homomorphic properties of the ElGamal cryptosystem [12] allows the servers to carry out most computations of the protocol. Note that this fact fits with a more real world oriented situation. The basic scheme is secure against a *passive adversary* which can corrupt some set of servers and obtain all their secret information, but can not force them to change their correct role in the protocol. Those subsets of servers that can be corrupted by the adversary are given by an *adversary structure* $\mathcal{A}$, which must be monotone decreasing: if a set of servers $B_1 \in \mathcal{A}$ can be corrupted, and $B_2 \subset B_1$, then the set of servers $B_2 \in \mathcal{A}$ can be corrupted, too.

But we want our scheme to be secure even under the most powerful attacks. In this case, in which we accept the presence of an *active adversary* which is able to alter the behavior of the corrupted players during the protocol, we must add some mechanisms in order to maintain the security and the correctness of the scheme. These tools are basically the use of verifiable secret sharing and

non-interactive zero-knowledge proofs of knowledge.

In all cases, we consider general adversary and access structures in the set of servers, not only the threshold ones. That is, those subsets of dishonest servers tolerated by the system as well as those subsets of servers that can provide a valid key to a user are not necessarily defined according to their cardinality. This general framework modelizes situations in which servers do not have all the same power or the same susceptibility to be corrupted. We state the combinatorial conditions that these structures must satisfy if we want our schemes to run securely.

Organization of the paper. In Section 2 we review some cryptographic tools that we will need throughout the rest of the paper, such as ElGamal encryption or basics on zero-knowledge proofs, and we also present the model of distributed key distribution schemes described in [16]. In Section 3 we explain secret sharing schemes for general access structures, and how they can be used by a set of participants to jointly generate a random secret shared value. In Section 4 we propose a new model in order to minimize users' computations and we propose a distributed key distribution scheme for this model, computationally secure against both passive and active adversaries. We give an explicit construction for the passive case, based on the homomorphic properties of ElGamal encryption scheme. Then, we introduce all the techniques that we use in order to provide robustness for the active case to our proposal. All our results consider general structures, not only threshold ones. Finally, in Section 5 we conclude the work summarizing our contribution and future research.

2 Preliminaries

In this section we describe some cryptographic tools that we will need later on. We will also explain the model of computationally secure Distributed Key Distribution Schemes introduced in [16].

2.1 ElGamal Encryption

In [12], ElGamal proposed a public-key probabilistic encryption scheme. We explain here an specific version of this scheme, but it can be generalized to work in any finite cyclic group (see [15], Section 8.4.2, for example).

The public parameters of the scheme are two large primes p and q, such that $q|p-1$, and a generator g of the multiplicative subgroup of $\mathbb{Z}_p^*$ with order q. Every user U generates both his public and private keys by choosing a random element $x \in \mathbb{Z}_q^*$ and computing $y = g^x \mod p$. The public key of user U is (p, q, g, y) and his private key is x.

If a user wants to encrypt a message $m \in \mathbb{Z}_p$ for user U, he chooses a random element $\beta \in \mathbb{Z}_q^*$, and computes $r = g^\beta \mod p$ and $s = my^\beta \mod p$. The ciphertext of message m that is sent to user U is $c = (r, s)$.

When U wants to recover the original message m from the ciphertext $c =$

(r, s), he computes

$$m \ = \ sr^{-x} \mod p$$

The semantic security of ElGamal cryptosystem is equivalent to the Decisional Diffie-Hellman Assumption [11]. One of the most useful features of this encryption schemes is its *homomorphic property*: if $c_i = (r_i, s_i)$ is a ciphertext corresponding to the plaintext m_i, for $i = 1, 2$ then $c = (r_1 r_2, s_1 s_2)$ is a ciphertext corresponding to plaintext $m = m_1 m_2$. This property is the one we need for the encryption scheme that we will use in our proposal of a new distributed and computationally secure key distribution scheme.

2.2 Zero-Knowledge Proofs of Knowledge

A zero-knowledge proof of knowledge allows a prover to demonstrate knowledge of a secret while revealing no information about it to the verifier of the proof, other than the mentioned knowledge and what the verifier was able to deduce prior to the protocol run. Zero-knowledge protocols are examples of interactive proof systems, in which a prover and a verifier exchange multiple messages, typically dependent on random numbers which they may keep secret. In these systems, there are security requirements for both the prover and the verifier: for the prover, security means that the protocol should be zero-knowledge, that is, the verifier gains no information on the secret; for the verifier, it means that the protocol should be a proof of knowledge: complete and sound. Intuitively, these two conditions mean that, with overwhelming probability, a honest verifier accepts a proof if and only if the prover is also honest. See [15], Section 10.4.1, for a comprehensive definition of these concepts.

Interactive proof systems can be transformed into non-interactive protocols, following the techniques and ideas of [14] and [19]. The security of such a non-interactive system is argued by showing that the plain interactive protocol is secure and then replacing the verifier with a collision resistant and random hash function; this approach has been formalized as the random oracle model [2].

In the context of this paper, we are specially interested in zero-knowledge proofs of the validity of statements about discrete logarithms. This topic has been deeply studied in works such as [8, 9]. We will use notation introduced by Camenisch and Stadler [9]: for instance, the statement

$$PK \ \{ \ (\alpha, \beta) : \ A = g_1^{\alpha} g_2^{\beta} \ \wedge \ B = g_3^{\alpha} \ \}$$

denotes a zero-knowledge proof of knowledge of values α and β such that $A = g_1^{\alpha} g_2^{\beta}$ and $B = g_3^{\alpha}$. By convention, Greek letters $(\alpha, \beta, \ldots)$ denote quantities whose knowledge is being proved, while all other parameters are known to the verifier (in this case, the values A, B, g_1, g_2, g_3).

2.3 Previous Computational Distributed Key Distribution Schemes

In [16] it was introduced the notion of Distributed Key Distribution Schemes in order to avoid the main drawbacks that the existence of a single Key Distribution

Scheme had. They considered a set of servers $\mathcal{S} = \{S_1, \ldots, S_n\}$ and a group of users $\mathcal{U} = \{U_1, \ldots, U_m\}$ (they also referred to them as clients). Each user U has private communication with at least t servers. Let $\mathcal{C} \subset 2^{\mathcal{U}}$ a family of sets of users, the *conferences* , who want to communicate securely among them.

Initialization: Each server S_i receives a share α_i of some random secret α, shared among the servers by means of Shamir secret sharing scheme. The generation of these values can be performed by either a central authority or jointly by a group of servers.

Regular Operation: if a user U in a conference $C \in \mathcal{C}$ needs the key of this conference, he proceeds as follows:

- He contacts t servers $S_1, \ldots, S_t$ and asks them for the key of the conference C. Each conference C is related to a public value h_C.
- Each server S_i, for $i = 1, \ldots, t$, verifies that the user is allowed to ask for that key and, if so, computes the value $h_C^{\alpha_i}$ and sends it to him through their private channel.
- After receiving the information from the servers, the user is able to compute the conference key κ_C as follows: $\kappa_C = h_C^{\alpha} = \prod_{i=1}^{t} (h_C^{\alpha_i})^{\lambda_i}$, where λ_i are the Lagrange interpolation coefficients.

3 Secret Sharing Schemes and Distributed Generation of a Random Secret Shared Value

In a *secret sharing scheme*, a dealer distributes shares of a secret value among a set of players $\mathcal{P} = \{P_1, \ldots, P_n\}$ in such a way that only authorized subsets of players (those in the called *access structure*) can recover the secret value from their shares, whereas non-authorized subsets do not obtain any information about the secret. The access structure is usually noted Γ. It must be monotone increasing, i.e. any subset containing an authorized subset will also be authorized.

Secret sharing schemes were introduced independently by Shamir [21] and Blakley [4] in 1979. Shamir proposed a *threshold* scheme, i.e. subsets that can recover the secret are those with at least t members (t is the threshold). Other works have proposed schemes realizing more general structures, such as *vector space secret sharing schemes* [7]. An access structure Γ is realizable by such a scheme defined in a finite field $\mathbb{Z}_q$, for some prime q, if there exists a positive integer r and a function $\psi : \mathcal{P} \cup \{D\} \longrightarrow (\mathbb{Z}_q)^r$ such that $W \in \Gamma$ if and only if $\psi(D) \in \langle \psi(P_i) \rangle_{P_i \in W}$. Here D denotes a special entity (real or not), outside the set $\mathcal{P}$. If a dealer wants to distribute a secret value $x \in \mathbb{Z}_q$, he takes a random element $\mathbf{v} \in (\mathbb{Z}_q)^r$, such that $\mathbf{v} \cdot \psi(D) = x$. The share of a participant $P_i \in \mathcal{P}$ is $x_i = \mathbf{v} \cdot \psi(P_i) \in \mathbb{Z}_q$. Let W be an authorized subset, $W \in \Gamma$; then,

$\psi(D) = \sum_{P_i \in W} \lambda_i^W \psi(P_i)$, for some $\lambda_i^W \in \mathbb{Z}_q$. In order to recover the secret, the players of W compute

$$\sum_{P_i \in W} \lambda_i^W x_i \;=\; \sum_{P_i \in W} \lambda_i^W \mathbf{v}\cdot\psi(P_i) \;=\; \mathbf{v}\cdot \sum_{P_i \in W} \lambda_i^W \psi(P_i) \;=\; \mathbf{v}\cdot\psi(D) \;=\; x \mod q\,.$$

Simmons, Jackson and Martin [22] introduced *linear secret sharing schemes*, that can be seen as vector space secret sharing schemes in which each player can be 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 they proposed results in an inefficient secret sharing scheme). From now on in our work, we will consider any possible access structure Γ, so we will know that there exists a linear secret sharing scheme realizing this structure. For simplicity, however, we will suppose that this scheme is a vector space one defined by a function ψ over $\mathbb{Z}_q$. See [24] for a comprehensive introduction to secret sharing schemes.

In many protocols, it is interesting to avoid the presence of a dealer who knows all the secret information of the system. The role of the dealer can be distributed among the players, as long as the secret is chosen at random. This distributed protocol must be protected against the presence of some coalition of players corrupted by an adversary. The monotone decreasing family of these tolerated coalitions of corrupted servers is the adversary structure $\mathcal{A}$. If the adversary is passive, the only required condition is $\Gamma \cap \mathcal{A} = \emptyset$, and the distributed generation of a random secret value can be performed by any authorized subset $A \in \Gamma$, as follows:

- Each player $P_i \in A$ chooses at random a value $k_i \in \mathbb{Z}_q$, and distributes it among all players in $\mathcal{P}$, using the corresponding vector space secret sharing scheme. That is, P_i chooses a random vector $\mathbf{v_i} \in (\mathbb{Z}_q)^r$ such that $\mathbf{v_i} \cdot \psi(D) = k_i$. Then P_i sends to each player P_j in $\mathcal{P}$ his share $k_{ij} = \mathbf{v_i} \cdot \psi(P_j)$. The generated random secret will be $x = \sum_{i \in A} k_i$.
- Each player $P_j \in \mathcal{P}$ computes his share of the secret x as $x_j = \sum_{i \in A} k_{ij}$.

In effect, suppose that an authorized subset of players $W \in \Gamma$ wants to recover the secret x. We know that there exist values $\{\lambda_j^W\}_{j \in W}$ such that $\psi(D) = \sum_{j \in W} \lambda_j^W \psi(P_j)$. Then players in W can obtain the secret x from their shares:

$$\sum_{j \in W} \lambda_j^W x_j = \sum_{j \in W} \lambda_j^W \sum_{i \in A} k_{ij} = \sum_{j \in W} \sum_{i \in A} \lambda_j^W \mathbf{v_i} \cdot \psi(P_j) = \sum_{i \in A} \mathbf{v_i} \sum_{j \in W} \lambda_j^W \psi(P_j) =$$

$$= \sum_{i \in A} \mathbf{v_i} \psi(D) = \sum_{i \in A} k_i = x$$

We denote an execution of this distributed protocol, in the passive adversary scenario, with the following expression:

$$(x_1, \ldots, x_n) \quad \overset{(\mathcal{P},\Gamma,\mathcal{A})}{\longleftrightarrow} \quad x$$

However, if the adversary is active, some players of $\mathcal{P}$ can cheat during the protocols. *Verifiable secret sharing schemes* were introduced in order to tolerate this kind of situations. The two most used verifiable secret sharing schemes are the proposals of Pedersen [18] and Feldman [13], which are both based on Shamir's secret sharing scheme. Whereas the security of secret sharing schemes is unconditional, that is, subsets that are not in the access structure do not obtain any information about the secret, independently of their computational capability, the security of some verifiable secret sharing schemes is based on some computational assumption; for instance, Feldman's scheme is secure assuming that the discrete logarithm problem in some finite field is hard.

Now we explain a distributed generation of a random secret value, shared among players in $\mathcal{P}$ according to the access structure Γ, and secure against the action of an active adversary who can corrupt a subset of players in the adversary structure $\mathcal{A}$. It must be performed by a subset R of players satisfying that for all $B \in \mathcal{A}$, we have $R - B \in \Gamma$. We denote by $\Omega = \Omega(\Gamma, \mathcal{A})$ the monotone increasing family formed by those subsets R. This family is not empty if and only if $\mathcal{A}^c \subset \Gamma$, where $\mathcal{A}^c = \{\mathcal{P} - B \mid B \in \mathcal{A}\}$. In effect, $\mathcal{P} \in \Omega$ if and only if for all $B \in \mathcal{A}$ we have that $\mathcal{P} - B \in \Gamma$, and this is equivalent to $\mathcal{A}^c \subset \Gamma$.

In the threshold case, where $\Gamma = \{W \subset \mathcal{P} \ : \ |W| \geq t\}$ and the adversary structure is usually taken as $\mathcal{A} = \{B \subset \mathcal{P} \ : \ |B| < t\}$, we have that $\Omega = \{R \subset \mathcal{P} \ : \ |R| \geq 2t - 1\}$. This family is not empty if and only if $n \geq 2t - 1$.

The protocol for generating a random secret value in a distributed way can be performed by a subset $R \in \Omega$ as follows:

- Each player $P_i \in R$ chooses at random a value $k_i \in \mathbb{Z}_q$, and distributes it among all players in $\mathcal{P}$, using the following (verifiable) vector space secret sharing scheme (it is a generalization of the threshold scheme of Feldman [13]). Let q and p be large primes such that $q | p - 1$. Let $\tilde{g}$ be a generator of a multiplicative subgroup of $\mathbb{Z}_p^*$ with order q.

  P_i chooses a random vector $\mathbf{v_i} = (v_i^{(1)}, \ldots, v_i^{(r)}) \in (\mathbb{Z}_q)^r$ such that $\mathbf{v_i} \cdot \psi(D) = k_i$. Then P_i sends to each player P_j in $\mathcal{P}$ his share $k_{ij} = \mathbf{v_i} \cdot \psi(P_j)$. He also makes public the commitments $V_{i\ell} = \tilde{g}^{v_i^{(\ell)}}$, for $1 \leq \ell \leq r$.

- Each player $P_j \in \mathcal{P}$ verifies the correctness of his share k_{ij} by checking that

$$\tilde{g}^{k_{ij}} = \prod_{\ell=1}^{r} (V_{i\ell})^{\psi(P_j)^{(\ell)}}$$

  If this check fails, P_j makes public a complaint against P_i.

- If player $P_i \in R$ receives complaints from players that form a subset that is not in $\mathcal{A}$, he is rejected. Otherwise, P_i makes public the shares k_{ij} corresponding to the players that have complained against him. If any one of these published shares do not satisfy the previous verification equation, P_i is also rejected.

- We denote by $Qual \subset R$ the (public) set of players that pass this verification phase. Due to the definition of the structure Ω, we have that $Qual$ belongs to Γ.

- The generated random secret will be $x = \sum_{i \in Qual} k_i$. Note that, since $Qual \in \Gamma$, we have that $Qual \notin \mathcal{A}$, and so any subset in $\mathcal{A}$ cannot obtain the secret x from their initial secret values k_i. Each player $P_j \in \mathcal{P}$ computes his share of the secret x as $x_j = \sum_{i \in Qual} k_{ij}$.

An authorized subset of players could obtain the value of x from their shares exactly in the same way explained for the passive case.

Note that the values $D_j = \tilde{g}^{x_j}$ can be publicly computed by all players as follows:

$$D_j = \tilde{g}^{\sum\limits_{i \in Qual} k_{ij}} = \prod_{i \in Qual} \tilde{g}^{k_{ij}} = \prod_{i \in Qual} \prod_{\ell=1}^{r} (V_{i\ell})^{\psi(P_j)^{(\ell)}}$$

We denote the output of this protocol with the expression:

$$(x_1, \ldots, x_n) \quad \overset{(\mathcal{P}, \Gamma, \mathcal{A})}{\longleftrightarrow} \quad (x, \tilde{g}, \{D_j\}_{1 \leq j \leq n})$$

4 Our Computational Secure Distributed Key Distribution Scheme

In [16] a construction based on the decisional Diffie-Hellman assumption was presented. However, this proposal requires a user to compute $O(t)$ exponentiations in order to obtain a key (where t is the minimum number of servers the user must contact with) whereas a server should compute only a single exponentiation in order to help a user. This may not correspond to real situations, where it is possible to take profit of the computational power of the servers. Thus, we are interested in a scheme minimizing the computational effort of the user. Next we will set up the new model of computationally secure distributed key distribution scheme that we will use from now on. Afterwards, we will present an explicit construction based on ElGamal encryption. We will take into account both passive and active adversaries. When we describe the protocol, first, we will consider a passive adversary, and later on, we will note which changes should be made in the protocol to provide security against an active adversary.

4.1 Setting up the model

Let $\mathcal{U} = \{U_1, \ldots, U_m\}$ be a set of m users and $\mathcal{S} = \{S_1, \ldots, S_n\}$ a set of n servers. Let $\Gamma \subset 2^{\mathcal{S}}$ be a general monotone increasing access structure, formed by those subsets of servers that are allowed to recover a secret from their shares; and let $\mathcal{A} \subset 2^{\mathcal{S}}$ be a general monotone decreasing adversary structure, formed by those subsets of dishonest servers that the system is able to tolerate. These two structures must satisfy $\mathcal{A} \cap \Gamma = \emptyset$. For simplicity, we assume that the access structure Γ can be realized by a vector space secret sharing scheme. That is,

there exist a positive integer r and a function $\psi : \mathcal{S} \cup \{D\} \longrightarrow (\mathbb{Z}_q)^r$ such that $A \in \Gamma$ if and only if $\psi(D) \in \langle \psi(S_i) \rangle_{S_i \in A}$.

Let $\mathcal{C} \subset 2^{\mathcal{U}}$ be a family of sets of users (conferences). Every user in a conference needs to know the conference key in order to communicate securely with other members of the conference. Let $\mathcal{R} \subset 2^{\mathcal{S}}$ be the family of sets of servers that a user must contact with in order to obtain the conference key. This family $\mathcal{R}$ must be monotone increasing, and will be different depending on the kind of adversary (passive or active) that we consider. We say that a set of servers in $\mathcal{R}$ is *robust*. We divide a distributed key distribution scheme into three different phases:

Initialization Phase. We assume that the initialization phase is performed by a robust subset of servers, that jointly performs the generation of shares $\{\alpha_i\}_{i \in \mathcal{S}}$ of a random value α, realizing the access structure Γ, by using the protocols explained in Section 3. Each server has a share α_i of α and any set that is not in Γ can obtain no information of this random secret value α.

Key Request and Computational Phase. A user U_j in a conference $C \in \mathcal{C}$ contacts with a robust subset of servers $A \in \mathcal{R}$ asking for the key of the conference C, which we will call κ_C. Every server $S_i \in A$ checks for membership of U_j in C. If he belongs to, server S_i computes a share of the conference key using α_i and a value related with the conference C. Afterwards, server S_i encrypts his share of the key by means of a suitable homomorphic encryption scheme with the public key of user U_j. The contacted group of servers A, by using homomorphic properties of the used cryptosystem, is able to compute an encryption of the conference key κ_C from the encryptions of the shares of the key.

Key Delivery Phase. Either a single server in A or the whole set A (depending on the behavior of the adversary, passive or active, respectively) sends the computed result to user U_j through an authenticated channel. Using his private key, the user will be able to decrypt this message obtaining in this way the conference key.

4.2 Our Proposal for the Passive Adversary Case

Now we propose a method to construct a Distributed Key Distribution Scheme computationally secure against a passive adversary who corrupts servers on a subset in $\mathcal{A}$, following the model introduced in Section 4.1. We use ElGamal cryptosystem [12], and take profit from its homomorphic properties.

We have an access structure Γ, such that the condition $\Gamma \cap \mathcal{A} = \emptyset$ holds. In this passive case, we have that the family of robust subsets is $\mathcal{R} = \Gamma$. Let p and q be two large primes such that $q|p-1$. Let H be a hash function (collision and pre-image resistant) that inputs a conference in $\mathcal{C}$ and outputs an element in $\mathbb{Z}_p^*$. We assume that each user U_j has a public ElGamal key (p, q, g, y_j) corresponding to a private key $x_j \in \mathbb{Z}_q^*$; that is, $y_j = g^{x_j} \mod p$, where g is an element with order q in $\mathbb{Z}_p^*$. Here we present our scheme:

Initialization Phase

A subset in $\mathcal{R} = \Gamma$ jointly performs the passive version of the protocol in Section 3 for the generation of a random shared secret, which results in

$$(\alpha_1, \ldots, \alpha_n) \quad \overset{(\mathcal{S},\Gamma,\mathcal{A})}{\longleftrightarrow} \quad \alpha$$

where $\alpha, \alpha_i \in \mathbb{Z}_q$ are random.

Key Request and Computational Phase

A user U_j in a conference $C \in \mathcal{C}$ asks for the conference key κ_C to a robust subset of servers $A \in \mathcal{R}$. These servers check the membership of the user in the conference and perform the following distributed encryption protocol. Note that $A \in \mathcal{R} = \Gamma$ is an authorized set of servers and we are assuming that the access structure Γ is realized by a vector space secret sharing scheme defined by the function ψ. Thus, there exist values $\{\lambda_i^A\}_{S_i \in A}$ in $\mathbb{Z}_q$ such that $\psi(D) = \sum_{S_i \in A} \lambda_i^A \psi(S_i)$ and so $\alpha = \sum_{S_i \in A} \lambda_i^A \alpha_i \mod q$ (in the threshold case, these values $\{\lambda_i^A\}_{S_i \in A}$ would be the Lagrange interpolation coefficients). Servers in A proceed as follows:

- Each server $S_i \in A$ applies the hash function H to the conference C, obtaining $h_C = H(C) \in \mathbb{Z}_p^*$. The conference key will be $\kappa_C = h_C^\alpha$. Then each $S_i \in A$ encrypts the value $h_C^{\alpha_i} \mod p$ using the ElGamal public key of user U_j, which is (p, q, g, y_j). That is:
  - Server S_i chooses a random element $\beta_i \in \mathbb{Z}_q^*$.
  - He computes $r_i = g^{\beta_i} \mod p$ and $s_i = h_C^{\alpha_i} y_j^{\beta_i} \mod p$.
  - Server S_i broadcasts the ciphertext $c_i = (r_i, s_i)$.
- Now each server $S_i \in A$ can mpute the encryption (r, s) of the conference key $\kappa_C = (h_C)^\alpha$ as follows:

$$r = \prod_{S_i \in A} r_i^{\lambda_i^A} = (g)^{\sum_{S_i \in A} \lambda_i^A \beta_i} \mod p$$

$$s = \prod_{S_i \in A} s_i^{\lambda_i^A} = (h_C)^{\sum_{S_i \in A} \lambda_i^A \alpha_i} (y_j)^{\sum_{S_i \in A} \lambda_i^A \beta_i} = h_C^\alpha (y_j)^{\sum_{S_i \in A} \lambda_i^A \beta_i} \mod p$$

  Since the elements $\{\beta_i\}_{S_i \in A}$ are random, we have that the element $\sum_{S_i \in A} \lambda_i^A \beta_i$ is also random, and so (r, s) is a valid ElGamal encryption of the message h_C^α. We also note that the resulting ciphertext (r, s) does not depend on the authorized subset $A \in \Gamma$ that has been considered.

Key Delivery Phase

The ciphertext $c = (r, s)$ is sent by some server $S_i \in A$ to user U_j, who decrypts it (he is the only one who can do this) and obtains automatically the conference key $\kappa_C = h_C^\alpha$.

4.3 Achieving Robustness Against an Active Adversary

Next we will consider an adversary who corrupts servers on a subset in $\mathcal{A}$, in an active way; that is, those corrupted servers may not follow the protocol properly. The condition $\Gamma \cap \mathcal{A} = \emptyset$ is still necessary, of course. In this active scenario, the family $\mathcal{R}$ of robust subsets of servers will be $\mathcal{R} = \Omega(\Gamma, \mathcal{A})$ defined as in Section 3. Note that the condition $\mathcal{A}^c \subset \Gamma$ is necessary and sufficient in order to make sure that the family $\mathcal{R}$ is not empty (again, the justification is explained in Section 3).

The following changes must be introduced in each one of the phases:

Initialization Phase

We require a robust subset of servers to perform this phase. They jointly generate a random shared secret, using verifiable secret sharing (see Section 3) to detect corrupted servers:

$$(\alpha_1, \ldots, \alpha_n) \quad \overset{(\mathcal{S},\Gamma,\mathcal{A})}{\longleftrightarrow} \quad (\alpha, \tilde{g}, \{D_i\}_{1 \leq i \leq n})$$

where $\tilde{g}$ is an element with order q in $\mathbb{Z}_p^*$ and $D_i = \tilde{g}^{\alpha_i}$ are the public commitments associated with the shares α_i's of the secret value α.

Note that although the adversary corrupts a tolerated set of servers, these corrupted servers will be detected; the remaining servers of the robust subset will belong to the access structure Γ, because of the definition of the family $\mathcal{R}$, and they will able to finish the protocol correctly.

Key Request and Computational Phase

Now a user must ask for a conference key κ_C to a robust subset A of servers. After this, every server S_i in A broadcasts a ciphertext $c_i = (r_i, s_i)$ of its share $h_C^{\alpha_i}$ of the conference key as in the passive case.

We must deal with the case of corrupted servers who want to boycott the system, by broadcasting a ciphertext $\tilde{c}_i = (\tilde{r}_i, \tilde{s}_i)$ which does not correspond to the plaintext $h_C^{\alpha_i}$.

We will detect these corrupted servers if we impose them to do a determined proof of knowledge. After the joint generation of the secret shared value α, all servers know public commitments $D_i = g^{\alpha_i}$ to the value α_i, for $1 \leq i \leq n$. Each server, after broadcasting $c_i = (r_i, s_i)$, must prove that he knows values α_i and β_i such that $D_i = g^{\alpha_i}$, $r_i = (g)^{\beta_i}$ and $s_i = (h_C)^{\alpha_i}(y_j)^{\beta_i}$. The rest of servers will play the role of a verifier in this non-interactive proof of knowledge. So, following the notation of Section 2.2, each server must perform :

$$PK \ \{ \ (\alpha_i, \beta_i) : \ D_i = g^{\alpha_i} \ \wedge \ r_i = g^{\beta_i} \ \wedge \ s_i = (h_C)^{\alpha_i}(y_j)^{\beta_i} \ \}$$

where $D_i, r_i, s_i, g, g^{\alpha}, h_C, g$ are elements known to the verifiers. We present now a protocol to achieve this non-interactive proof of knowledge; it is similar to the one that appears in [1], and uses standard techniques introduced by Camenisch [8], Stadler [23] and Camenisch and Stadler [9]. In the random oracle model, the security of this protocol can be proved using the same strategies as them.

The proof $PK\{(\alpha,\beta):\ A=g_1^{\alpha}\ \wedge\ B=g_2^{\beta}\ \wedge\ C=g_3^{\alpha}g_4^{\beta}\}$ is as follows: let $\ell \leq k$ be two security parameters and $\hat{H}:\{0,1\}^* \rightarrow \{0,1\}^k$ be a hash function. The prover does the following:

1. Generate 2ℓ numbers $u_1,\ldots,u_\ell$ and $v_1,\ldots,v_\ell$ at random in $\mathbb{Z}_q^*$
2. Compute, for $1 \leq i \leq \ell$, the values $t_i = g_1^{u_i}$, $t'_i = g_2^{v_i}$ and $t''_i = g_3^{u_i}g_4^{v_i}$
3. Compute $c = \hat{H}(A,B,C,g_1,g_2,g_3,g_4,t_1,\ldots,t_\ell,t'_1,\ldots,t'_\ell,t''_1,\ldots,t''_\ell)$
4. Compute, for $1 \leq i \leq \ell$

   if $c[i]=0$ then $w_i = u_i$ and $w'_i = v_i$
   if $c[i]=1$ then $w_i = u_i - \alpha$ and $w'_i = v_i - \beta$

5. The proof of knowledge is the tuple $(c, w_1,\ldots,w_\ell,w'_1,\ldots,w'_\ell)$

The verifier of the proof must do the following:

1. Compute, for $1 \leq i \leq \ell$

   if $c[i]=0$ then $\tilde{t}_i = g_1^{w_i}$, $\tilde{t}'_i = g_2^{w'_i}$ and $\tilde{t}''_i = g_3^{w_i}g_4^{w'_i}$
   if $c[i]=1$ then $\tilde{t}_i = Ag_1^{w_i}$, $\tilde{t}'_i = Bg_2^{w'_i}$ and $\tilde{t}''_i = Cg_3^{w_i}g_4^{w'_i}$

2. Compute $c' = \hat{H}(A,B,C,g_1,g_2,g_3,g_4,\tilde{t}_1,\ldots,\tilde{t}_\ell,\tilde{t}'_1,\ldots,\tilde{t}'_\ell,\tilde{t}''_1,\ldots,\tilde{t}''_\ell)$
3. If $c'=c$, then accept the proof; otherwise, reject the proof.

Each server S_i verifies the proofs published by the rest of servers, until he obtains accepted partial ciphertexts from a subset of servers in Γ. Notice that this subset in Γ always exists, because of the definition of the family $\mathcal{R}$. Then S_i can use the correct values $c_j = (r_j, s_j)$ corresponding to servers S_j in this subset in Γ to compute, exactly in the same way as we have shown in Section 4.2, an encryption (r,s) of the conference key $\kappa_C = h_C^{\alpha}$, using the homomorphic properties of ElGamal cryptosystem.

Key Delivery Phase

Each server in A sends the encryption of the conference key to user U_j. After receiving these messages, user U_j selects from the whole list of values, the one which is sent by all the servers of a subset that is not in $\mathcal{A}$. This implies that there exists at least one honest server in this subset (otherwise, the subset would be in $\mathcal{A}$), and so the corresponding ciphertext must be the correct one. U_j decrypts it by means of his private key, obtaining in this way the required conference key.

4.4 Some Remarks

Note that, although ElGamal cryptosystem is probabilistic, all the honest servers obtain the same ciphertext (r, s) of the requested conference key, because of the deterministic way in which they must calculate this ciphertext from the probabilistic ciphertexts (r_i, s_i).

In the case of a passive adversary, all servers follow the protocol correctly. So, a user could ask a single server for the key instead of an entire robust subset. This server will then contact with a robust subset, and the protocol will follow as we explain in Section 4.2. In the active case, this is not possible because the users do not know which servers are honest, thus they could ask wrongly a corrupted server, who could boycott the protocol.

The fact that we denote as robust the subsets of servers that can provide a valid conference key to a user is not accidental. We define these robust subsets in such a way that their members can execute the protocol correctly even if they contain some subset of players corrupted by the adversary. Roughly speaking, that is the definition of a robust distributed protocol, and for this reason we use the terminology of robust subsets.

And last but not least, note that in some way, the model we propose can be rewritten as a Multi-party protocol. Indeed, the protocol in which servers compute shares of the encryption of a conference key from their shares of the random secret value α fits in a Multi-party framework. This could be used in order to prove security properties of the protocol by means of using techniques of Canetti [10] to prove security in Multi-party protocols.

5 Conclusion

In this paper we introduce a new model for distributing keys in a distributed way in the computationally secure framework, and we design a protocol realizing it. This model minimizes the computations that every user has to carry out in order to obtain a key, and transmits them to the servers, which are supposed to have more powerful computational resources. In order to fit this protocol into a real oriented scenario we introduce techniques to provide security against both passive and active adversaries who can corrupt some groups of servers. We consider general structures, not only threshold ones, for both subsets of servers that can provide a valid key to a user and subsets of servers that can be corrupted by the adversary. We find the combinatorial conditions that these structures must satisfy if we want our scheme to run securely.

In our model, we require secure and authenticated channels among the servers only in the initialization phase. In the rest of phases, servers only need an authenticated broadcast channel among them. In the communication between a user and a server, authenticated channels are needed, but not secure ones, because the information that servers send to users is encrypted. This last point is an improvement with respect to the model in [16], because in that proposal secure channels between servers and users were required. Even the

requirement of secure channels among the servers can be eliminated (in our proposal as well as in [16]), if the secret sharing schemes that servers use in the initialization phase are *publicly verifiable* (see [23, 20] for the details). The use of these schemes, however, reduces the efficiency of the distributed generation of a random secret shared value in Section 3.

In the passive case, a user only needs to decrypt a value (basically, one exponentiation) in order to obtain the requested key. Recall that in the proposal of [16] each user had to compute $O(t)$ exponentiations to get the key. In the active case, he must in addition compare a list of values and detect the correct ciphertext. But these operations are always necessary if we consider an active adversary, because the user must verify in some way which of the informations that he receives come from a corrupted server and are, therefore, incorrect.

Some interesting questions arise from this work: first of all, it must be defined in a formal way all security requirements that must satisfy a distributed key distribution scheme and prove the security of our scheme based on this security model. Maybe the strategy is to see these schemes as Multi-Party protocols, and apply the security results of Canetti [10] in this scenario. It would be also interesting to check if other cryptosystems could fit in with our model, and if so, to study the efficiency of the consequent schemes. Likewise, some other security requirements such as proactivity or resharing would be desirable.

References

[1] G. Ateniese, D. Song and G. Tsudik. Quasi-efficient revocation in group signatures. *Proc. of Sixth International Financial Cryptography Conference* (2002).

[2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *First ACM Conference on Computer and Communications Security* p. 62-73 (1993).

[3] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. *Proc. 27th Annual Symposium on the Theory of Computing, ACM*, 1995.

[4] G.R. Blakley. Safeguarding cryptographic keys. *Proceedings of the National Computer Conference, American Federation of Information.* Processing Societies Proceedings **48** p. 313-317 (1979).

[5] C. Blundo and P. D'Arco. Unconditionally secure distributed key distribution schemes. Preprint available at `http://www.dia.unisa.it/paodar.dir`

[6] C. Blundo, P. D'Arco, V. Daza and C. Padró. Bounds and constructions for unconditionally secure distributed key distribution schemes with general access structures. *Proc. of the Information Security Conference (ISC 2001).* LNCS 2200, Springer, p. 1-17 (2001).

[7] E.F. Brickell. Some ideal secret sharing schemes. *J.Combin. Math. and Combin. Comput.* **9** p. 105-113 (1989).

[8] J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. PhD thesis, ETH Zurich. *Diss. ETH No. 12520* (1998).

[9] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. *Advances in Cryptology: CRYPTO'97*, LNCS 1294, Springer-Verlag, p. 410-424 (1997).

[10] R.Canetti. Security and composition of multi-party cryptographic protocols. Journal of Cryptology **13** (1) p. 143-202, (2000).

[11] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory, IT-22, 6* p. 644-654 (1976).

[12] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **31** p. 469-472 (1985).

[13] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*. IEEE Press, p. 427-437 (1987).

[14] A. Fiat and A. Shamir. How to prove yourself: practical solution to identification and signature problems. *Advances in Cryptology: CRYPTO'86*, LNCS 263, Springer, p. 186-194 (1987).

[15] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Inc., Boca Raton (1997).

[16] M. Naor, B. Pinkas and O. Reingold. Distributed pseudo-random functions and KDCs. *Advances in Cryptology: Eurocrypt'99*, LNCS 1592, Springer-Verlag, p. 327-346 (1999).

[17] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, vol. **21** p. 993-999 (1978).

[18] T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Advances in Cryptology: CRYPTO'91*, LNCS 576, Springer-Verlag, p. 129-140 (1991).

[19] C. Schnorr. Efficient identification and signatures for smart cards. *Advances in Cryptology: CRYPTO'89*, LNCS 435, Springer-Verlag, p. 239-252 (1989).

[20] B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its applications to electronic voting. *Advances in Cryptology: CRYPTO'99*, LNCS 1666, Springer-Verlag, p. 148-164 (1999).

[21] A. Shamir. How to share a secret. *Communications of the ACM* No. **22** p.612-613 (1979).

[22] G. J. Simmons, W. Jackson and K. Martin. The geometry of secret sharing schemes. *Bulletin of the ICA* **1** p.71-88 (1991).

[23] M. Stadler. Publicly verifiable secret sharing. *Advances in Cryptology: Eurocrypt'96*, LNCS 1070, Springer-Verlag, p. 190-199 (1996).

[24] D.R. Stinson. *Cryptography: Theory and Practice*. CRC Press Inc., Boca Raton (1995).