On Distributed Key Distribution Centers and Unconditionally Secure Proactive Verifiable Secret Sharing Schemes Based on General Access Structure

(Corrected Version)

Ventzislav Nikov[1], Svetla Nikova[2] ⋆, Bart Preneel[2], and Joos Vandewalle[2]

[1] Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB, Eindhoven, the Netherlands
vnikov@mail.com
[2] Department Electrical Engineering, ESAT/COSIC,
Katholieke Universiteit Leuven, Kasteelpark Arenberg 10,
B-3001 Heverlee-Leuven, Belgium
svetla.nikova, bart.preneel, joos.vandewalle@esat.kuleuven.ac.be

Changes

S. Fehr [14] pointed out that denoting the complement $\Gamma_A = \Delta_A^c$ [20, 13] as honest (or good) players structure appears to be a misleading term. Actually its dual access structure $\Gamma_A^\perp$ should be called the honest (or good) players structure, since for any set G of good players the complement G^c is the set of corrupted players from Δ_A. This reflects in some changes of the notations in Theorem 2 and Theorem 4 from [20] as well as in changes of the notations for good players structure in some protocols.

Abstract. A Key Distribution Center of a network is a server enabling private communications within groups of users. A Distributed Key Distribution Center is a set of servers that *jointly* realizes a Key Distribution Center. In this paper we build a robust Distributed Key Distribution Center Scheme secure against active and mobile adversary. We consider a general access structure for the set of servers and for the adversary access structure. We also revise the unconditionally secure Verifiable Secret Sharing Schemes from [11, 10, 20, 23] proposing a modified version which is proactively secure.

1 Introduction

A new approach to the key distribution was introduced by Naor et.al. in [19]. A Distributed Key Distribution Center (DKDC) is a set of n servers of a network

⋆ The author was partially supported by NATO research fellowship and Concerted Research Action GOA-MEFISTO-666 of the Flemish Government.

that *jointly realize* the function of a Key Distribution Center. A user who needs to participate in a conference sends a key-request to a subset of his own choosing of the n servers, and the contacted servers answer with some information enabling the user to compute the conference key. In such a model, a single server by itself does not know the secret keys, since they are *shared* between the n servers. In subsequent papers [3, 4, 8] the notion of the DKDC has been studied from an information theoretic point of view. Recently in [11, 10] a robust verifiable DKDC based on unconditionally secure proactive threshold VSS was proposed.

In [19] Naor et.al. gave some specific proposals both in unconditional and in the computational security framework. Their computational secure scheme is based on the Decisional Diffie-Hellman Assumption. Recently in [9] the Naor's computational security model was modified and a scheme based on the ElGamal cryptosystem was proposed.

Verifiable secret sharing schemes (VSSs) are secret sharing schemes (SSSs) dealing with possible cheating by the participants. The concept of proactive security was introduced by Ostrovsky and Yung in [21] and applied to the secret sharing schemes by Herzberg et.al. in [16]. Basically the idea is that, if the information stored by the servers in order to share a given secret stays the same for all lifetime of the system, then an adversary can eventually break into a sufficient number of servers, to learn and destroy the secret. On the other hand, let the time is divided into periods. At the beginning of each period the information stored by the servers in a given time period changes, while the shared secret stays the same. Then the adversary probably does not have enough time to break into necessary number of servers. Moreover, the information he learns during the period p is useless during the period $p+i$, for $i = 1, 2, \ldots$. So, he has to start a new attack from scratch during each time period. The first unconditionally secure proactive VSS was proposed by Stinson and Wei [23], where proactivity is added to the basic VSS described in the same paper. A generalization of the scheme has subsequently been given in [20]. Recently D'Arco and Stinson [11, 10] showed that some existing proactive schemes [20, 23] can be broken. They proposed two new variations of the schemes to add proactive security to VSS, based on two different approaches, one using symmetric polynomials and another one using non-symmetric polynomials.

In this paper we present a Robust Unconditional Proactive Verifiable DKDS, enabling a set of servers to jointly realize a Key Distribution Center. The basic building block will be an unconditionally secure proactive VSS based on a general access structure. We will use the scheme proposed by D'Arco and Stinson [11, 10], whose round complexity has been improved by applying the technique described by Gennaro et. al. in [15]. We also show an attack on the unconditionally secure proactive SSS with symmetric polynomials from [10] and propose a slightly modified scheme that solves the problem (see also [11]).

2 Background

2.1 Notations

Let K be finite field. For an arbitrary matrix M over K, with m rows labelled by $1, \ldots, m$ let M_A denote the matrix obtained by keeping only those rows i with $i \in A$, where A is an arbitrary non-empty subset of $\{1, \ldots, m\}$. If $\{i\} = A$ we write M_i. Consider the set of row-vectors $\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_k}$ and let $A = \{i_1, \ldots, i_k\}$ be the set of indices, then we denote by $\mathbf{v}_A$ the matrix consisting of rows $\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_k}$. Instead of $\langle \varepsilon, \mathbf{v}_i \rangle$ for $i \in A$ we will write $\langle \varepsilon, \mathbf{v}_A \rangle$. Let M_A^T denote the transpose of M_A, and let $Im(M_A^T)$ denote the K-linear span of the rows of M_A. We use $Ker(M_A)$ to denote the kernel of M_A, i.e. all linear combinations of the columns of M_A, leading to 0.

Let us define the standard inner product $\langle x, y \rangle$ and $x \perp y$, when $\langle x, y \rangle = 0$. For a K-linear subspace V of K^t, $V^\perp$ denotes the collection of elements of K^t, that are orthogonal to all of V (the orthogonal complement), which is again a K-linear subspace. For all subspaces V of K^t we have $V = (V^\perp)^\perp$, $(Im(M_N^T))^\perp = Ker(M_N)$ or $Im(M_N^T) = (Ker(M_N))^\perp$, $\langle x, M_N^T y \rangle = \langle M_N x, y \rangle$.

A matrix which i-th row is of the form $(1, \alpha_i, ..., \alpha_i^{t-1})$, where $\alpha_1, ..., \alpha_n \in K$, is called (n, t)-Vandermonde matrix (over K) with $t < n$. It is well known that any square Vandermonde matrix has non-zero determinant. If M is an (n, t)-Vandermonde matrix over K and A is non-empty subset of $\{1, ..., n\}$, then the rank of M_A is maximal (i.e. is equal to t, or equivalently, $Im(M_A^T) = K^t$) if and only if $|A| \geq t$. Moreover let ε denotes the column vector $(1, 0, ..., 0) \in K^t$. If $|A| < t$, then $\varepsilon \notin Im(M_A^T)$, i.e. there is no $\lambda \in K^{|A|}$ such that $M_A^T \lambda = \varepsilon$.

2.2 General Access Structure, Monotone Span Program and LSSS

We call the groups who are allowed to reconstruct the secret *qualified*, and the groups who should not be able to obtain any information about it *forbidden*. The collection of all qualified groups is denoted by Γ, and the collection of all forbidden groups is denoted by Δ. In fact Γ is *monotone increasing* and Δ is *monotone decreasing*. The tuple (Γ, Δ) is called *access structure* if $\Gamma \cap \Delta = \emptyset$. If $\Gamma \cup \Delta = 2^P$, where P is the set of participants, then we say that (Γ, Δ) is *complete* and we denote it by Γ. Otherwise we say that (Γ, Δ) is *incomplete*. By Γ^- we denote the collection of the *minimal sets* of Γ and by Δ^+ – the collection of the *maximal sets* of Δ. It is obvious that (Γ^-, Δ^+) generates (Γ, Δ). We will consider general monotone access structure (Γ, Δ), which describes subsets of participants that are qualified to recover the secret $s \in K$ in the set of possible secret values.

There exists an adversary A which can *corrupt* a set of servers during any time period. Corrupting a server means learning the secret information in the server, modifying its data, sending out wrong message, and so on. Since the server can be rebooted, the adversary is a mobile one. The collection of all possible corrupted servers for fixed time period we call *bad* and is denoted by Δ_A. The collection of all possible uncorrupted servers for the same period of time we call *good* and

we denote it by $\Gamma_A^{\perp}$. So, we can consider a second complete access structure $\Gamma_A^{\perp}$, which is called an *adversary access structure* [17].

The simplest example of adversary access structure is to set a number b to be the maximum number of broken (corrupt) servers by adversary for fixed time frame (i.e. the threshold case) [11, 10, 23].

A new operation for the access structure, which generalize the notion of $Q^2(Q^3)$ adversary structure introduced by Hirt and Maurer [17], is given in [20].

Definition 1. *[20] For the access structure (Γ, Δ) the operation $*$ can be defined as follows: $n * \Delta = \{A = A_1 \cup A_2; A_1 \in (n-1) * \Delta, A_2 \in \Delta\}$, for $n = 2, 3, \ldots$.*

Definition 2. *[20] For the complete access structure Γ the operation $*$ can be defined as follows: First we set $\Delta = 2^P \setminus \Gamma$ and (as in Definition 1) calculate $n * \Delta$. Then we define $n * \Gamma = 2^P \setminus n * \Delta$, for $n = 2, 3, \ldots$*

The same operation for monotone structures is defined by Fehr and Maurer in [13], which they call element-wise union, in order to give necessary and sufficient conditions for robust VSS and Distributed Commitments.

Brickell [5] pointed out how the linear algebraic view leads to a natural extension to a wider class of secret sharing schemes that are not necessarily of the threshold type. This have later been generalized to all possible so-called monotone access structures by Karchmer and Wigdreson [18] based on a linear algebraic computational device called *monotone span program* (MSP).

Definition 3. *[18] The quadruple $\mathcal{M} = (K, M, \varepsilon, \psi)$ is called monotone span program, where K is a finite field, M is a matrix (with m rows and $d \leq m$ columns) over K, $\psi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ is a surjective function and ε is a fixed vector, called target vector, e.g. column vector $(1, 0, ..., 0) \in K^d$. The size of $\mathcal{M}$ is the number of rows m.*

Here ψ labels each row with a number from $[1, \ldots, m]$ corresponding to a fixed player, so we can think of each player as being the "owner" of one or more rows. And for every player we consider a function φ which gives the set rows owned by the player. In some sense φ is inverse of ψ.

Theorem 1. *[2, 12, 18] MSP is said to compute an access structure (Γ, Δ) if the following holds: $\varepsilon \in Im(M_N^T)$ if and only if N is a member of Γ.*

Lemma 1. *[7] The vector $\varepsilon \notin ImM_N^T$ if and only if there exists $\mathbf{k} \in K^d$ such that $M_N\mathbf{k} = 0$ and $\mathbf{k}_1 = 1$.*

A SSS is linear if the dealer and the participants use only linear operations to compute the shares and the secret. Each *linear SSS* (LSSS) can be viewed as derived from a monotone span program computing its access structure. On the other hand, each monotone span program gives rise to an LSSS. Hence, one can identify an LSSS with its underlying monotone span program. Note that the size of $\mathcal{M}$ is also the size of the corresponding LSSS. Now we will consider any access structure, as long as it admits a linear secret sharing scheme.

2.3 The Model of DKDC

From now on we will follow the settings in [10, 11]. Let $U = \{U_1, \ldots, U_m\}$ be a set of m users and let $S = \{S_1, \ldots, S_n\}$ be a set of n servers. Each user has a *private channel* connecting him or her to *all* the servers. Each pair of servers is connected by a *private channel* and all of them share a *broadcast channel*. Servers can be *good* or *bad* (i.e., they are controlled by an adversary and can deviate from the protocol in arbitrary ways). Let $\mathcal{C} \subset 2^U$ be the family of *conferences*, i.e. the family of groups of users which want to communicate privately. And let F be the family of *tolerated coalitions*, i.e. the family of coalitions of users who can try to break the scheme in some way. We consider a general access structure (Γ, Δ) for the set of servers, we also consider the adversary access structure $\Gamma_A^{\perp}$.

A *verifiable distributed key distribution scheme* (VDKDS) is divided in three phases: an *initialization phase*, which involves only the servers; a *key-request phase*, in which users ask servers for keys; and a *key-computation phase*, in which users construct keys from the messages received from the servers who were contacted during the key-request phase.

Initialization phase: We assume that the initialization phase is performed by a *joint computation* of all the servers. As a primitive for these phase we use a VSS (proactive VSS), so each server S_i is able to verify the information received. Moreover, each server constructs a list $\mathcal{G}$ of good servers presented across the network at the end of this phase. (Note that the lists held by the good servers contain the same identifiers.)

Key-request: Let $C_h \in \mathcal{C}$ be a conference. Each user U_j in C_h, contacts a subset $\mathcal{G}$ of good servers belonging to $\Gamma_A^{\perp}$, requesting a key for the conference C_h. We denote such key $\overline{k}_h$. Each good server S_i, contacted by a user U_j, checks for membership of U_j in C_h; if $U_j \in C_h$, then S_i computes a value $y_{i,j}^h$, using a public known function. Otherwise, S_i sets $y_{i,j}^h = \perp$ (a special value which does convey no information about $\overline{k}_h$). Finally, S_i sends the value $y_{i,j}^h$ to U_j. Note that a bad server can either refuse to replay or it may send some incorrect value.

Key-computation phase: Having received the values from the servers, each user U_j in C_h computes $\overline{k}_h$ from a certain majority of the values received.
Roughly speaking, a Verifiable DKDC must satisfy the following properties:

- **Correct and Verifiable Initialization Phase.** When the initialization phase successfully terminates, any good server S_i must be able to identify the subsets of good servers and to compute his private information.
- **Consistent Key Computation.** Each user in a conference $C_h \subseteq U$ must be able to compute the *same* conference key, after interacting with a subset of good servers.
- **Conference Key Security.** A conference key must be secure against attacks performed by a coalition of bad servers, coalition of users, and coalition of both.

Or in a more precise way:

Definition 4. *[11, 10] Let $U = \{U_1, \ldots, U_m\}$ be a set of users and let $S = \{S_1, \ldots, S_n\}$ be a set of servers. Let $\mathcal{C}$ be the family of conferences and let F be*

the family of tolerated coalitions. A verifiable $((\Gamma, \Delta), \Gamma_A^{\perp}, m, n, \mathcal{C})$*-Distributed Key Distribution Scheme is a protocol which enables each user of* $C_h \in \mathcal{C}$ *to compute a common key* $\overline{k}_h$ *interacting with set of servers of the network. More precisely, the following properties are satisfied:*

- *After the initialization phase, each good server computes his private information and verifies its consistency with the information received and stored by other good servers. At least a set of servers successfully completes this phase and each of them constructs the same (public) list* $\mathcal{G}$ *containing the identities of the good servers.*
- *After the initialization phase, each good server is able to answer the key-request messages.*
- *Each user in* $C_h \in \mathcal{C}$ *can compute the common key* $\overline{k}_h$ *by contacting the servers in* $\mathcal{G}$*. At least one subset of the good servers* $\mathcal{G}$ *from* $(3 * \Gamma_A)^{\perp}$ *gives good answers, from which the user reconstructs the key.*
- *Each conference key is completely secure against coalition of users in* F*; coalitions of set of servers* ($\notin \Gamma_A^{\perp}$)*; and joint coalitions of users and servers.*

3 A VSS

The main component of our $((\Gamma, \Delta), \Gamma_A^{\perp}, m, n, \mathcal{C})$-VDKDS is a VSS. Since secret sharing were proposed initially by Shamir [24] and Blakley [1], research on this topic has been extensive. In the "classic" secret sharing schemes, there are assumed to be no faults in the system. Chor et.al. [6] first defined the complete notion of VSS. In this section we provide a slightly modified version of unconditionally secure VSS proposed by Stinson and Wei in [23], with improved round complexity by applying the technique described in [15], but for the general access structure. For the precise definition of the VSS one can see [11, 10, 20, 23].

3.1 Distribution (Share) Phase

Let $s \in K$ be a secret.

1. The dealer $\mathcal{D}$ chooses a random symmetric matrix $R \in K^{d,d}$, subject to s in its upper left corner. He sends $\mathbf{v}_{\varphi(k)} = M_{\varphi(k)}R$ (the row-vectors) to P_k.
2. Then each P_i generates and sends to every P_k random values $r_{\varphi(i),\varphi(k)} \in K^{|\varphi(i)|,|\varphi(k)|}$ through a private channel.
3. After receiving $r_{\varphi(i),\varphi(k)}$, each P_k broadcasts the values $M_{\varphi(i)}\mathbf{v}^T_{\varphi(k)} + r_{\varphi(i),\varphi(k)} + r^T_{\varphi(k),\varphi(i)}$ for each $i \neq k$.
4. Each P_i computes the minimum subset $\mathcal{G} \subset \{P_1, \ldots, P_n\}$, such that any ordered pair $(e, k) \in \mathcal{G} \times \mathcal{G}$ is *consistent*, i.e. such that $M_{\varphi(e)}\mathbf{v}^T_{\varphi(k)} + r_{\varphi(e),\varphi(k)} + r^T_{\varphi(k),\varphi(e)} = (M_{\varphi(k)}\mathbf{v}^T_{\varphi(e)} + r_{\varphi(k),\varphi(e)} + r^T_{\varphi(e),\varphi(k)})^T = \mathbf{v}_{\varphi(e)}M^T_{\varphi(k)} + r^T_{\varphi(k),\varphi(e)} + r_{\varphi(e),\varphi(k)}$. If $\mathcal{G} \in \Gamma_A^{\perp}$, then P_i outputs $ver_i = 1$ otherwise P_i outputs $ver_i = 0$.

It is obvious that every good participant computes the same subset $\mathcal{G}$ at the end of *Share*. Next we consider the reconstruction phase. Note that although the adversary is static, he could provide correct information in *Share* phase but wrong information in *Reconstruction* phase. It means that the adversary access structure in the reconstruction phase is $(2 * \Gamma_A)^{\perp}$.

3.2 Reconstruction Phase

1. Each player P_i sends $\langle \varepsilon^T, \mathbf{v}_{\varphi(i)} \rangle$ to P_k, where $i, k \in \mathcal{G}$, the set of good participants after the distribution phase.
2. After receiving the information, P_k computes λ, such that $M^T_{\varphi(\widetilde{\mathcal{G}})} \lambda = \varepsilon$, for some group $\widetilde{\mathcal{G}} \subset \mathcal{G}$ and $\widetilde{\mathcal{G}} \in (2 * \Gamma_A)^{\perp}$.
3. Denote by R_1 the first column in R, hence $s = \langle R_1, \varepsilon \rangle = \langle R_1, M^T_{\varphi(\widetilde{\mathcal{G}})} \lambda \rangle = \langle M_{\varphi(\widetilde{\mathcal{G}})} R_1, \lambda \rangle = \langle (M_{\varphi(\widetilde{\mathcal{G}})} R)_1, \lambda \rangle = \langle (\mathbf{v}_{\varphi(\widetilde{\mathcal{G}})})_1, \lambda \rangle$, where $(\mathbf{v}_{\varphi(\widetilde{\mathcal{G}})})_1$ is the column-vector of the first coordinates of each share, i.e. $\langle \varepsilon^T, \mathbf{v}_{\varphi(\widetilde{\mathcal{G}})} \rangle$.

Note that the joint information held by the players in $\mathcal{G}$ is $\mathbf{v}_{\varphi(\mathcal{G})} = M_{\varphi(\mathcal{G})} R$. It can be shown that the security of the protocol remains the same see [11, 10, 15, 20, 23].

The following theorem, proved in [20], gives sufficient conditions for existence of a unconditionally secure verifiable secret sharing scheme.

Theorem 2. *[20] The scheme is a unconditionally secure verifiable secret sharing scheme if the following condition is satisfied:*
i) $(2 * \Gamma_A)^{\perp} \subseteq \Gamma$.

The following result of Fehr and Maurer [13] proves that the conditions are also necessary.

Theorem 3. *The very strong robustness property for VSS is fulfilled if and only if* $P \notin \Delta * (2 * \Delta_A) = \Delta * \Delta_A * \Delta_A$.

4 Proactivity

Proactive security for secret sharing was first suggested by Ostrovski and Yung in [21], where they presented, among other things, a proactive polynomial secret sharing scheme. The polynomial proactive secret sharing scheme proposed in [21] uses the verifiable secret sharing scheme from [22]. Proactive security refers to security and availability in the presence of a mobile adversary. Herzberg et.al. [16] further specialized this notion to robust secret sharing schemes and gave a detailed efficient proactive secret sharing scheme. *Robust* means that in any time period, the shareholders can reconstruct the secret value correctly. There are also many papers that discuss proactive security, see e.g. the references in [16, 21–23].

The secret value needs to be maintained for a long period of time. Then the life time is divided into time periods which are determined by the global clock.

At the beginning of each time period the server engages in an interactive update protocol. The update protocol will not reveal the value of the secret. At the end of the period the servers hold new shares of the secret. We distinguish the following phases in each time period [16]. At the beginning we have *Distribution* or *Recovery*, during the period *Renewal* and at the end *Reconstruct* or *Detection* followed by *Recovery* for the beginning of the next period.

The first information theoretic unconditionally secure proactive VSS was proposed by Stinson and Wei in [23], were proactivity was added to the basic VSS described above. A generalization of that scheme to general access structure has subsequently been given in [20]. In [11, 10] D'Arco and Stinson found an attack to break the Renewal procedure given in [20, 23]. They also proposed a new variation of the scheme based on two different approaches for adding proactive security to VSS. The first technique uses symmetric polynomial and the second relies on the use of generic non-symmetric polynomial. The purpose of this section is to show an attack on the unconditionally secure proactive SSS with symmetric polynomials from [10] and to propose a slightly modified scheme that resists the attack and has better information rate (see also [11]).

First, we make the following remarks to the proposed in [10] solutions. In the non-symmetric scheme of D'Arco and Stinson besides the share (of length t) the servers should keep also the *verification share* of length t. So, the information which is kept by them is doubled, hence the information rate of the new scheme is reduced twice. In the symmetric scheme the servers should keep the share (of length t) and the *verification share* of length n, where $n > t + 3b$. Thus the information which is kept by them increases more than twice, hence the information rate of the new scheme is reduced more than twice.

4.1 Attack against proactivity

Now we start with the analysis of the Renewal phase in [10], which is as follows:

Renewal phase

1. Each server P_ℓ selects a random symmetric polynomial (i.e. $r_{i,j} = r_{j,i}$).

$$r^{(\ell)}(x, y) = \sum_{i=0}^{t-2} \sum_{j=0}^{t-2} r_{i,j} x^i y^j.$$

2. P_ℓ sends $h_k^{(\ell)}(x) = r^{(\ell)}(x, \omega^k)$ to P_k for $k = 1, 2, \ldots, n$ by a private channel.
3. After receiving $h_k^{(\ell)}(x)$, each P_k sends $h_k^{(\ell)}(\omega^m)$ to P_m for $k = 1, 2, \ldots, n$.
4. P_m checks whether $h_k^{(\ell)}(\omega^m) = h_m^{(\ell)}(\omega^k)$ for $k = 1, 2, \ldots, n$ and $k \neq m$. If P_m finds that the equality is not true, then he broadcasts an accusation of P_ℓ.
5. If P_ℓ is accused by at most b servers, then he can defend himself as follows. For those P_i he is accused by, P_ℓ broadcasts $h_i^{(\ell)}(x)$. Then, the server P_k checks whether $h_k^{(\ell)}(\omega^i) = h_i^{(\ell)}(\omega^k)$ and broadcasts "yes" or "no". If there are at least $n - b - 2$ servers broadcasting yes, then P_ℓ is not a bad server.

6. P_m updates the list of good servers $\mathcal{G}$ (i.e., the values ℓ for which P_ℓ is accused by at least $b+1$ servers, or found bad in the previous step are not in $\mathcal{G}$). Then, P_m updates its shares as

$$h_m(x) \longleftarrow h_m(x) + \omega^m h_m^*(x),$$

where $h_m^*(x) = \sum_{\ell \in \mathcal{G}} h_m^{(\ell)}(x)$. Moreover, P_m updates a verification vector V_m by computing

$$V_m[j] \longleftarrow V_m[j] + h_m^*(\omega^j).$$

First, note that instead of verification share $V_m[j]$ for $j = 1, 2, \ldots, n$ one can use a polynomial $V_m(x)$ of degree $t-2$, such that $V_m(\omega^j) = V_m[j]$. In fact we can change in step 6. $V_m(x) \longleftarrow V_m(x) + h_m^*(x)$. In this way the size of the verification share become $t-1$.

Unfortunately the information from the share and verification share of server P_i allows the attacker to calculate the initial share of P_i, obtained from the Dealer during the Distribution (Share) phase. Indeed, after q executions of *Renewal* P_i possesses

$$h_i(x) = h_i^0(x) + \omega^i \sum_{p=1}^{q} h_i^{*,p}(x)$$

and

$$V_i(x) = \sum_{p=1}^{q} h_i^{*,p}(x).$$

Subtracting $\omega^i V_i(x)$ from $h_i(x)$ the attacker obtain the initial share $h_i^0(x)$. The consequence is that if a passive adversary breaks into t servers once, even in different periods, he collects t initial shares and hence he can recover the secret.

4.2 Modification of the Scheme

First we will consider the threshold case. Basically, the problem in the above procedure is due to the asymmetry in the renewal polynomial. Indeed, we have

$$r(x, y) \longleftarrow r(x, y) + y r^*(x, y)$$

where $r^*(x, y) = \sum_{\ell \in \mathcal{G}} r^{(\ell)}(x, y)$. Note that $r(0, 0)$ is not changed, so the secret stays the same. Also $r(0, y)$ is changed randomly so the adversary is not able to calculate the new values.

To be able to perform a pair-wise check one need a symmetry, that is why the servers keep two shares: one is the actual and the other is the verification share, which collects the asymmetry in the protocol from [10].

We propose to keep the symmetry in the renewal polynomial:

$$r(x, y) \longleftarrow r(x, y) + (x + y) r^*(x, y).$$

Hence in the **Renewal phase** for the threshold case we need to modify only step 6.

6′. P_m updates the list of good servers $\mathcal{G}$ (i.e., the values ℓ for which P_ℓ is accused by at least $b+1$ servers, or found bad in the previous step are not in $\mathcal{G}$). Then, P_m updates its shares as

$$h_m(x) \longleftarrow h_m(x) + (x+\omega^m)h^*_m(x),$$

where $h^*_m(x) = \sum_{\ell\in\mathcal{G}} h^{(\ell)}_m(x)$.

Now we do not need verification share any more.

For general access structure the modification of the Renewal phase of the proactive SSS in [20] will be as follows:

Renewal phase

1. Each server $P_e \in \mathcal{G}$ selects a random $(d-1)\times(d-1)$ symmetric matrix $R^{(e)}$ and using it constructs two symmetric $d\times d$ matrix $R^{(e,1)}, R^{(e,2)}$. $R^{(e,1)}$ is constructed by adding zero column and zero row as last row and column and $R^{(e,2)}$ is constructed by adding zero column and zero row as first row and column.
2. After that P_e sends $\mathbf{v}^{(e,1)}_{\varphi(k)} = M_{\varphi(k)}R^{(e,1)}$ and $\mathbf{v}^{(e,2)}_{\varphi(k)} = M_{\varphi(k)}R^{(e,2)}$ to all P_k by a private channel.
3. Each P_k checks whether the last column of $\mathbf{v}^{(e,1)}_{\varphi(k)}$ is zero-column and whether the first column of $\mathbf{v}^{(e,2)}_{\varphi(k)}$ is zero-column too. If these conditions are not satisfied P_k broadcasts an accusation to P_e, otherwise P_k computes $\mathbf{v}^{(e)}_{\varphi(k)}$ as the sum of the right shift of the coordinates of $\mathbf{v}^{(e,1)}_{\varphi(k)}$ and the left shift of the coordinates of $\mathbf{v}^{(e,2)}_{\varphi(k)}$. i.e. if we denote $\mathbf{v}^{(e,2)}_h = (0, (\mathbf{v}^{(e,2)}_h)_1, \ldots, (\mathbf{v}^{(e,2)}_h)_{d-1})$ and $\mathbf{v}^{(e,1)}_h = ((\mathbf{v}^{(e,1)}_h)_1, \ldots, (\mathbf{v}^{(e,1)}_h)_{d-1}, 0)$ then $\mathbf{v}^{(e)}_h = ((\mathbf{v}^{(e,2)}_h)_1, (\mathbf{v}^{(e,2)}_h)_2 + (\mathbf{v}^{(e,1)}_h)_1, \ldots, (\mathbf{v}^{(e,2)}_h)_{d-1} + (\mathbf{v}^{(e,1)}_h)_{d-2}, (\mathbf{v}^{(e,1)}_h)_{d-1})$, where $h \in \varphi(k)$. Finally, P_k computes and sends to P_j the values $M_{\varphi(j)}(\mathbf{v}^{(e,1)}_{\varphi(k)})^T$, $M_{\varphi(j)}(\mathbf{v}^{(e,2)}_{\varphi(k)})^T$ and $M_{\varphi(j)}(\mathbf{v}^{(e)}_{\varphi(k)})^T$.
4. P_j checks whether $M_{\varphi(j)}(\mathbf{v}^{(e)}_{\varphi(k)})^T = \mathbf{v}^{(e)}_{\varphi(j)}M^T_{\varphi(k)}$, $M_{\varphi(j)}(\mathbf{v}^{(e,1)}_{\varphi(k)})^T = \mathbf{v}^{(e,1)}_{\varphi(j)}M^T_{\varphi(k)}$ and $M_{\varphi(j)}(\mathbf{v}^{(e,2)}_{\varphi(k)})^T = \mathbf{v}^{(e,2)}_{\varphi(j)}M^T_{\varphi(k)}$ for the values of e not accused by some set of servers from $(2*\Gamma_A)^\perp$ (in step 3). If the set of values of k, for which the equations are not true, belongs to $(2*\Gamma_A)^\perp$, then P_j broadcasts an accusation of P_e.
5. If P_e is accused by some set of servers from $(2*\Gamma_A)^\perp$ (from steps 3 and 4), then he can defend himself as follows. For those P_i that P_e is accused by, P_e broadcasts $\mathbf{v}^{(e,1)}_{\varphi(i)}$ and $\mathbf{v}^{(e,2)}_{\varphi(i)}$. Then all servers P_k check whether $M_{\varphi(i)}(\mathbf{v}^{(e)}_{\varphi(k)})^T = \mathbf{v}^{(e)}_{\varphi(i)}M^T_{\varphi(k)}$, $M_{\varphi(i)}(\mathbf{v}^{(e,1)}_{\varphi(k)})^T = \mathbf{v}^{(e,1)}_{\varphi(i)}M^T_{\varphi(k)}$ and $M_{\varphi(i)}(\mathbf{v}^{(e,2)}_{\varphi(k)})^T = \mathbf{v}^{(e,2)}_{\varphi(i)}M^T_{\varphi(k)}$ and broadcasts *"yes"* or *"no"*. If the set of servers broadcasting *"yes"* is from $(2*\Gamma_A)^\perp$, then P_e is not a bad server.

6. P_j updates the list of bad servers $\mathcal{L}$ by including all values e for which P_e is accused by at least one set from $(2 * \Gamma_A)^{\perp}$ or found bad in the previous step. Then P_j updates its shares as $\mathbf{v}_{\varphi(j)} \longleftarrow \mathbf{v}_{\varphi(j)} + \sum_{e \notin \mathcal{L}} \mathbf{v}^{(e)}_{\varphi(j)}$.

Because of the symmetry all other procedures are the same as in [23] for the threshold case and as in [20] for the general access structure. Note that the information rate of the new scheme is optimal and equal to the rates in [20, 23].

The following theorem, which is proved in [20], gives sufficient conditions for the existence of an unconditionally secure proactive secret sharing scheme.

Theorem 4. *[20] The scheme is a unconditionally secure proactive secret sharing scheme if the following conditions are satisfied:*
ii) $(3 * \Gamma_A)^{\perp} \subseteq \Gamma$.
iii) For each group $N \in \Gamma^{-}$ *the number of rows* $|\varphi(N)|$ *for the group is equal to number of columns of matrix* M.

5 A Proactive Verifiable DKDS

Using LSSS as a primitive and based on the linearity of the system we can build a DKDS. If we use a VSS instead of LSSS we can set up a Verifiable DKDS. Finally, if as primitive we use a proactive VSS we can build a Proactive VDKDS. The only difference between LSSS and VSS appears in the Set up phase. A straightforward solution to gain proactive security could be directly to apply, at the beginning of each time period the procedures *Detection*, *Recovery* and *Renewal* for each of the secrets.

We assume that a Dealer $\mathcal{D}$ initializes the system, but as it is noted in [11, 10], it is also possible the system to be initialized without the Dealer. The scheme proposed in this section provides ℓ-wise independent conference keys (as in [11, 10]), i.e. the ℓ-th conference key is uniformly distributed over the set of possible values, even if an adversary already knows $\ell - 1$ conference keys. The Set up phase is as follows.

5.1 Set up Phase

1. Let $\ell_{\widetilde{F}}$ be the maximum number of conference keys that a group $\widetilde{F}$ can compute. Assume that $\ell > max\{\ell_{\widetilde{F}}; \widetilde{F} \in F\}$. The Dealer $\mathcal{D}$ chooses a *random* secret column vector $k = (k_1, \ldots, k_\ell)$ and publishes an $\ell \times \ell$ matrix N, consisting of linearly independent row vectors, i.e. $rank(N) = \ell$. The conference key for C_s is then defined by $\overline{k}_s = \langle k^T, N_s \rangle$.
2. Then for each coordinate of the vector k the Dealer runs ℓ independent copies of the proactive VSS Σ_z described before, where the secret that each proactive VSS Σ_z distributes among the servers is k_z for $z = 1, \ldots, \ell$.
3. Each server S_i stores ℓ packets of shares $\mathbf{v}_{\varphi(i),k_z}$ sent by the Dealer during the executions of the Share Phase of the Σ_z's, and publishes the list of good servers $\mathcal{G} \in \Gamma_A^{\perp}$ he has found.

In a VSS the reconstruction of the secret is done by the participants (i.e. the servers in our settings) while in DKDS each user of a given conference contacts the servers, receives some information and computes the common key by applying a public function to the values received. Basically, the values sent by the servers must enable them to compute a single key, namely, the one the user is asking for.

5.2 Key Request and Key Computation Phase

1. User $U_j \in C_s$ asks a subset of good servers from $(2 * \Gamma_A)^{\perp}$ for the key $\overline{k}_s$.
2. Each server S_i computes temporary shares $\mathbf{v}_{\varphi(i),s} = \sum_{z=1}^{\ell} (N_s)_z \mathbf{v}_{\varphi(i),k_z}$ and sends the first column of $\mathbf{v}_{\varphi(i),s}$ to $U_j \in C_s$ i.e. $(\mathbf{v}_{\varphi(i),s})_1 = \langle \mathbf{v}_{\varphi(i),s}, \varepsilon^T \rangle$.
3. U_j computes λ, such that $M^T_{\varphi(\widetilde{\mathcal{G}})} \lambda = \varepsilon$, for some group $\widetilde{\mathcal{G}} \subset \mathcal{G}$ and $\widetilde{\mathcal{G}} \in (3 * \Gamma_A)^{\perp}$. Finally, he recovers $\overline{k}_s = \langle \lambda, (\mathbf{v}_{\varphi(\widetilde{\mathcal{G}}),s})_1 \rangle$.

Correctness. The correctness of the construction can be shown as follows: According to step 1. in the Set up Phase $\overline{k}_s = \langle k^T, N_s \rangle = \sum_{z=1}^{\ell} (N_s)_z k_z$ but from the Reconstruct Phase of VSS we have that

$$\begin{aligned} &\langle \lambda \, , (\mathbf{v}_{\varphi(\widetilde{\mathcal{G}}),k_z})_1 \rangle = \langle \lambda, (M_{\varphi(\widetilde{\mathcal{G}})} R^{k_z})_1 \rangle = \langle \lambda, M_{\varphi(\widetilde{\mathcal{G}})} (R^{k_z})_1 \rangle \\ &= \langle (R^{k_z})_1, M^T_{\varphi(\widetilde{\mathcal{G}})} \lambda \rangle = \langle (R^{k_z})_1, \varepsilon \rangle = k_z. \end{aligned}$$

Hence

$$\begin{aligned} \overline{k}_s &= \sum_{z=1}^{\ell} (N_s)_z \langle \lambda, M_{\varphi(\widetilde{\mathcal{G}})} (R^{k_z})_1 \rangle = \langle \lambda, \sum_{z=1}^{\ell} (N_s)_z M_{\varphi(\widetilde{\mathcal{G}})} (R^{k_z})_1 \rangle \\ &= \langle \lambda, M_{\varphi(\widetilde{\mathcal{G}})} \sum_{z=1}^{\ell} (N_s)_z (R^{k_z})_1 \rangle = \langle M^T_{\varphi(\widetilde{\mathcal{G}})} \lambda, \sum_{z=1}^{\ell} (N_s)_z (R^{k_z})_1 \rangle \\ &= \langle \varepsilon, \sum_{z=1}^{\ell} (N_s)_z (R^{k_z})_1 \rangle = \langle \varepsilon, (R^s)_1 \rangle. \end{aligned}$$

So, we can think for the secret conference key $\overline{k}_s$ as a secret distributed with VSS using the temporary random symmetric matrix $R^s = \sum_{z=1}^{\ell} (N_s)_z R^{k_z}$. That is why in step 2. in Key Request phase the server S_i needs to compute the temporary shares $\mathbf{v}_{\varphi(i),s}$.

On the other hand we have:

$$\begin{aligned} \overline{k}_s &= \sum_{z=1}^{\ell} (N_s)_z \langle \lambda, M_{\varphi(\widetilde{\mathcal{G}})} (R^{k_z})_1 \rangle = \sum_{z=1}^{\ell} (N_s)_z \langle \lambda, (M_{\varphi(\widetilde{\mathcal{G}})} R^{k_z})_1 \rangle \\ &= \sum_{z=1}^{\ell} (N_s)_z \langle \lambda, (\mathbf{v}_{\varphi(\widetilde{\mathcal{G}}),k_z})_1 \rangle = \langle \lambda, \sum_{z=1}^{\ell} (N_s)_z (\mathbf{v}_{\varphi(\widetilde{\mathcal{G}}),k_z})_1 \rangle \end{aligned}$$

$$= \langle \lambda, (\sum_{z=1}^{\ell} (N_s)_z \mathbf{v}_{\varphi(\widetilde{\mathcal{G}}),k_z})_1 \rangle = \langle \lambda, (\mathbf{v}_{\varphi(\widetilde{\mathcal{G}}),s})_1 \rangle.$$

Thus the user U_j is able to restore the secret conference key in step 3. of the Key Computation Phase.

6 Conclusions

In this paper we have shown how to set up a Robust Unconditional Proactive Verifiable DKDS, enabling a set of servers to jointly realize a Key Distribution Center. We have used unconditionally secure proactive VSS based on a general access structure as a building block. Basically, we can use only the VSS based on a general access structure (as a building block) and the structure of the DKDS will stay the same.

We have also revised the unconditionally secure VSSs from [10, 20, 23], proposing a modified version which is proactively secure. Since the proactivity, considered as security property, can be useful in several settings in which the adversary is mobile, the applicability of such schemes has independent interest of the specific application to key distribution that has been addressed in this paper.

It is clear that using the linear unconditional Proactive Verifiable DKDC as a base and the homomorphic properties of Diffie-Helman or ElGamal cryptosystem one can build a computational secure Proactive Verifiable DKDC. Using the ideas in [9] they can be made more efficient.

7 Acknowledgements

The authors would like to thank Paolo D'Arco and Dough Stinson for the fruitful discussions and comments.

References

1. **G. R. Blakley**, Safeguarding cryptographic keys, *AFIPS Conference Proc.* 48, 1979, pp. 313-317.
2. **G. R. Blakley, G. A. Kabatianskii**, Linear Algebra Approach to Secret Sharing Schemes, Springer Verlag *LNCS* 829, 1994, pp. 33-40.
3. **C. Blundo, P. D'Arco, V. Daza, C. Padro**, Bounds and Constructions for Unconditionally Secure Distributed Key Distribution Schemes for General Access Structures, *Proc. of the Information Security Conference* (ISC 2001), *LNCS* 2200, 2001, pp. 1-17.
4. **C. Blundo, P. D'Arco, C. Padro**, A ramp model for distributed key distribution schemes *WCC* 2001, pp. 92-102.
5. **E. F. Brickell**, Some ideal secret sharing schemes, *J. of Comb. Math. and Comb. Computing* 9, 1989, pp. 105-113.
6. **B. Chor, S. Goldwasser, S. Micali, B. Awerbuch**, Verifiable secret sharing and achieving simultaneity in the presence of faults, *Proc. of the IEEE 26th Annual Symp. on Foundations of Computer Science* 1985, pp. 383-395.

7. **R. Cramer**, Introduction to Secure Computation. In Lectures on Data Security - Modern Cryptology in Theory and Practice, *LNCS* 1561, 1999, pp. 16-62.
8. **P. D'Arco**, On the Distribution of a Key Distribution Center, *Proc. of ICTCS 2001*, *LNCS* 2202, 2001, pp. 357-369.
9. **V. Daza, J. Herranz, C. Padro, G. Saez** A distributed and computationally secure key distribution scheme, *Cryptology ePrint Archive*, Report 2002/069,
10. **P. D'Arco, D. Stinson**, On Unconditionally Secure Proactive Secret Sharing Scheme and Distributed Key Distribution Centers, *unpublished manuscript*, May 2002.
11. **P. D'Arco, D. Stinson**, On Unconditionally Secure Robust Distributed Key Distribution Centers, to appear in *ASIACRYPT 2002*.
12. **M. van Dijk**, A Linear Construction of Secret Sharing Schemes, *DCC* 12, 1997, pp. 161-201.
13. **S. Fehr, U. Maurer**, Linear VSS and Distributed Commitments Based on Secret Sharing and Pirwise Checks, *Proc. CRYPTO 2002*, Springer Verlag LNCS 2442, pp. 565-580.
14. **S. Fehr, V. Nikov, S. Nikova**, private communication.
15. **R. Gennaro, Y. Ishai, E. Kushlevitz, T. Rabin**, The round complexity of Verifiable Secret Sharing and Secure Multicasting, *Proc. STOC 2001*.
16. **A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung**, Proactive secret sharing or: How to cope with perpetual leakage, *Proc. CRYPTO 1995*, Springer Verlag LNCS 963, pp. 339-352.
17. **M. Hirt, U. Maurer**, Player Simulation and General Adversary Structures in Perfect Multiparty Computation, *J. of Cryptology* 13, 2000, pp. 31-60.
18. **M. Karchmer, A. Wigderson**, On Span Programs, *Proc. of 8-th Annual Structure in Complexity Theory Conference*, San Diego, California, 18-21 May 1993. IEEE Computer Society Press, pp. 102-111.
19. **M. Naor, B. Pinkas and O. Reingold**, Distributed Pseudo-random Functions and KDCs, *EuroCrypt'99*, *LNCS* 1592, 1999, pp. 327-346.
20. **V. Nikov, S. Nikova, B. Preneel, J. Vandewalle**, Applying General Access Structure to Proactive Secret Sharing Schemes, *Proc. of the 23rd Symposium on Information Theory in the Benelux*, May 29-31, 2002, Universite Catolique de Lovain (UCL), Lovain-la-Neuve, Belgium, pp. 197-206, *Cryptology ePrint Archive*: Report 2002/141.
21. **R. Ostrovsky, M. Yung**, How to withstand mobile virus attack, ACM Symposium on principles of distributed computing, 1991, pp. 51-59.
22. **T. Rabin, M. Ben-Or**, Verifiable secret sharing and multiparty protocols with honest majority, *Proc. of the 21st Annual ACM Symp. on Theory of Computing* 1989, pp. 73-85.
23. **D.R. Stinson, R. Wei**, Unconditionally Secure Proactive Secret Sharing Scheme with combinatorial Structures, *SAC'99*, Springer Verlag LNCS 1758, pp. 200-214.
24. **A. Shamir**, How to share a secret, *Communications of the ACM* 22, 1979, pp. 612-613.