Unconditionally Secure Proactive Secret Sharing Scheme with Combinatorial Structures

D.R. Stinson and R. Wei
Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
{dstinson, rwei}@cacr.math.uwaterloo.ca

Abstract

Verifiable secret sharing schemes (VSS) are secret sharing schemes dealing with possible cheating by the participants. In this paper, we propose a new unconditionally secure VSS. Then we construct a new proactive secret sharing scheme based on that VSS. In a proactive scheme, the shares are periodically renewed so that an adversary cannot get any information about the secret unless he is able to access a specified number of shares in a short time period. Furthermore, we introduce some combinatorial structure into the proactive scheme to make the scheme more efficient. The combinatorial method might also be used to improve some of the previously constructed proactive schemes.

1 Introduction

One important topic in cryptography is how to securely share a secret among a group of people. In some cases, many people need to share the power to use a cryptosystem. Thus some secret information should be shared by a group so that the cryptosystem can be used only if it is permitted by a specified subset of the group. The study of how to keep a secure backup of a secret key and how to recover it securely has been first studied by Blakley [4] and Shamir [23] independently. Shamir proposed a polynomial threshold scheme. In a (t, n)-threshold scheme, a secret value is shared by n participants such that any t of the participants can reconstruct the secret value by putting their shares together, but any $t-1$ participants cannot get any information about the secret value. In such a scheme, an adversary needs to compromise at least t locations in order to learn the secret, and corrupt at least $n-t-1$ locations to destroy the secret.

In many situations, such as cryptographic master keys, data files, legal documents, etc., a secret value needs to be stored for a long time. In these situations, an adversary may attack the locations one by one and eventually get the secret or destroy it. To prevent such an attack, proactive secret sharing schemes are proposed. Proactive security for secret sharing was first suggested by Ostrovsky and Yung in [18]. In [18] they presented, among other things, a proactive polynomial secret sharing scheme. Proactive security refers to security and availability in the the presence of a mobile adversary. Herzberg et al. [15] specialized this notion to robust secret sharing schemes and gave a detailed efficient proactive secret sharing scheme. In their scheme, a secret value is shared by n servers. The mobile adversary is able to attack all the servers during a long period of time. However, since the corrupted servers can be rebooted, in any time period there are only a subset of servers that are corrupted. "Robust" means that in any time period, the servers can reconstruct the secret value correctly.

The scheme in [15] is based on Shamir's polynomial threshold scheme, thus most aspects of the scheme are unconditionally secure. However, their scheme depends on verifiable secret sharing schemes based on [9, 19] which depend on some cryptographic assumptions.The security of the scheme in [9] is based on the hardness of solving discrete logarithm. In the scheme of [19], the privacy of the secret is unconditionally secure, but the correctness of the shares depends on a computational assumption. In a sense, these two schemes complement each other.

The purpose of this paper is to provide a new proactive secret sharing scheme which is unconditionally secure, i.e., the security of any part of the scheme is not based on any cryptographic assumption. Let S be the set of possible secret values, where $|S| = q$. Then unconditional security of the scheme means that at any time the adversary cannot guess the shared secret $s \in S$ with probability better than $\frac{1}{q}$.

We first propose an unconditionally secure verifiable secret sharing scheme. This scheme is an improvement of the absolute VSS in [3]. Then we propose several protocols to make it proactive. Following from the method of [15], the lifetime of the secret is divided into periods of time in the proactive scheme. In each time period, the n shares will be renewed while the secret remains the same. In this way, a mobile adversary who is able to attack (learn or corrupt) at most b shares (where $n > 3b + 1$) in a time period cannot learn any information about the secret in the long lifetime. This scheme is also robust, i.e., the secret can be reconstructed at any time.

Furthermore, we introduce some combinatorial structures in the scheme so that the scheme will be more efficient. With the combinatorial structure, most of the computation of the system will depend on the parameter b. Thus there is a "trade-off" between the computation and the value of b: when b is smaller (the ability of the adversary is more limited), the computation takes less time. Thus our scheme is more efficient in the situation when the number of the possible corrupted servers are much smaller as compared to the total number of the servers in the system. On the other hand, our combinatorial method might be easily adapted to the scheme of [15] to make the scheme more efficient.

The rest of this paper is arranged as follows. In Section 2 we give some preliminaries and the main settings of the system. Section 3 describes our new verifiable secret sharing scheme. We also propose an anonymous VSS in a subsection. Section 4 describes the proactive scheme without combinatorial structure. Section 5 introduces the combinatorial structure and describes how to apply it to the proactive scheme.

2 Preliminaries

2.1 Previous work

Proactive refers to the security of the scheme in the presence of a mobile adversary who may corrupt all participants of the scheme throughout the lifetime of the system but cannot corrupt too many participants during any short period of time. Such a mobile adversary was first considered by Ostrovsky and Yung in [18].

The motivation of [18] is to combat mobile viruses. The scheme requires the participants to constantly exchange messages and to be able to erase parts of its memory. A polynomial secret sharing proactive scheme is proposed which uses the verifiable secret sharing scheme of [22].

Herzberg et al. [15] further discussed proactive secret sharing schemes and gave a detailed practical scheme. In their scheme the lifetime is divided into periods of time. At the beginning of each time period, the share holders engage in an interactive update protocol which includes a share recovery protocol and a share renewal protocol. At the end of the period, each shareholder holds completely new shares of the same secret. The secret will not be computed during the update phase while it can be reconstructed at any time. They used the polynomial-based method from [18] for the renewal protocol. They also proposed a polynomial-based method for share recovery protocol. The verifiable secret sharing schemes they used are from [9, 19].

There are also many papers that discuss proactive security, see e.g., [6, 14, 11, 21] and their references. Our discussion will mainly follow the papers [15, 18].

2.2 The setting

We will follow the setting of the scheme in [15, 18]. We assume that there is a system of n servers $P_1, P_2, \cdots, P_n$, which are connected to a common broadcast channel such that messages sent through this channel instantly reach every server. We also assume that the system is synchronized, i.e., the servers can access a common global clock, and that each server has a local source of randomness. To make things simpler, we assume that there are private channels between each pair of servers and that messages sent by broadcast are safely authenticated. With these assumptions, we are able to focus on the proactive scheme itself.

There is an adversary which can corrupt b servers during any time period. Corrupting a server means learning the secret information in the server, modifying its data, sending out wrong message, changing the intended behavior of the server, disconnecting it, and so on. Since the server can be rebooted, the adversary is a mobile one.

A secret value $s \in GF(q)$ will be shared by the servers through the scheme. The value of s needs to be maintained for a long period of time. The life time is divided into time periods which are determined by the global clock. At the beginning of each time period the servers engage in an interactive update protocol. The update protocol will not reveal the value of s. At the end of the period the servers hold new shares of s. The mobile adversary who correpts b servers in a time period cannot get any information about the secret value s. The system can reproduce s in the presence of the mobile adversary at any time.

We consider unconditional security in this paper, which means that the adversary cannot guess the secret with probability better than $\frac{1}{q}$ if the secret $s \in GF(q)$.

3 Verifiable secret sharing

Since secret sharing schemes were proposed initially by Shamir [23] and Blakley [4], research on this topic has been extensive. In the "classic" secret sharing schemes, there are assumed to be no faults in the system. Tompa and Woll [26], and McEliece and Sarwate [17] first considered schemes with faulty participants and gave partial solutions for that problem. In their schemes, the dealer is always assumed honest. Chor et al. [8] first defined the complete notion of Verifiable Secret Sharing (VSS), and gave a solution which is based on some cryptographic assumption. In a VSS, each holder of a share can verify that the share is consistent with the other shares. Thus both the dealer and other participants can be verified in such a scheme. There are two aspects of the security in a VSS. One is the security of the secret and the other is the security of the verification.

There are many papers which have discussed VSS recently. Most schemes use zero-knowledge proofs, e.g., [3, 7, 10, 13, 20, 22]. Others use cryptographic assumptions such as the hardness of discrete logarithm, see [9, 19]. [12] proposed a simple and efficient VSS, but it based on some "collision resistance" assumption. On the other hand, many known VSS are not easy to adapt for proactive property.

The VSS in [22, 19, 9] are used in proactive schemes in [18, 15]. [18] used the VSS from [22] which used some zero-knowledge proofs. [15] used the VSS of Feldman [9] and Pedersen [19]. The security of the scheme in [9] is based on the hardness of solving discrete logarithm. In the scheme of [19], the privacy of the secret is unconditionally secure, but the verification depends on a computational assumption.

In [3] it was shown that in any unconditionally secure VSS, $b < \frac{n}{3}$. Thus the VSS with $b \geq \frac{n}{3}$ will either depend on some cryptographic assumption or have small propability of errors. In this section, we will propose an unconditionally secure VSS with $b \leq \frac{n-2}{3}$, which is simpler and more efficient than the scheme in [3]. Moreover, our scheme is easy to transform into a proactive scheme.

3.1 Definition

Now we give a formal definition of VSS as follows.

Suppose there are a dealer D and n other participants $P_1, P_2, \cdots, P_n$ all connected by private communication channels. They also have access to a broadcast channel. There is a static adversary A that can corrupt up to b of the participants including the D. Here static means that the b participants controlled by the adversary are fixed.

Let π be a protocol consisting two of phases $Share$ and *Reconstruct*. Let S be the set of possible secret values. At the beginning of $Share$, the dealer inputs a secret $s \in S$. At the end of $Share$ each participant P_i is instructed to output a Boolean value ver_i. At the end of *Reconstruct* each participant is instructed to output a value in S.

The protocol π is an unconditionally secure *Verifiable Secret Sharing* protocol if the following properties are hold:

1. A good player P_i outputs $ver_i = 0$ at the end of $Share$ if and only if the share he received is not consistent with the other good players;

2. If the dealer is good, then $ver_i = 1$ for every good P_i;

3. If all good players P_i output $ver_i = 1$ at the end of $Share$, then there exists an $s' \in S$ such that the event that all good P_i output s' at the end of *Reconstruct* is fixed at the end of $Share$ and $s' = s$ if the dealer is good;

4. If $|S| = q$ and s is chosen randomly from S, and the dealer is good, then any adversary A cannot guess at the end of $Share$ the value s with probability better than $\frac{1}{q}$.

3.2 The new VSS

In this subsection we provide a new unconditionally secure VSS which will be used in our proactive scheme later.

Suppose there is a dealer D and n participants P_i, $1 \leq i \leq n$, where $n \geq t + 2b + 1$ and $t > b$. Let $S = GF(q)$ be a finite field and let ω be a primitive element in $GF(q)$. In the following protocol, all the computations are in the field $GF(q)$. We first state the share phase as follows.

Share

1. When D wants to share a secret value $s \in S$, he chooses a random symmetric polynomial
$$f(x, y) = \sum_{i=0}^{t-1}\sum_{j=0}^{t-1} a_{ij}x^i y^j,$$
where $a_{00} = s$ and $a_{ij} = a_{ji}$ for all i, j. Then, for each k, D sends $h_k(x) = f(x, \omega^k)$ to P_k through a private channel.

2. After receiving $h_k(x)$, each P_k sends $h_k(\omega^l)$ to P_l for $1 \leq l \leq n, (l \neq k)$ through a private channel.

3. Each P_l checks whether $h_k(\omega^l) = h_l(\omega^k)$ for $1 \leq k \leq n, (l \neq k)$. If P_l finds, for at least $n - b - 1$ values of k, that $h_k(\omega^l) = h_l(\omega^k)$, then P_l accepts the shares and outputs $ver_l = 1$, and P_l also records all the P_k for those k which causes $h_k(\omega^l) \neq h_l(\omega^k)$ in a $list_l$. Otherwise, P_l refuses the shares and outputs $ver_l = 0$.

Next we consider the reconstruct phase. Note that although the adversary is static, he could provide correct information in $Share$ phase but wrong information in *Reconstruct* phase.

Reconstruct

1. For each P_i with $ver_i = 1$, P_i sends $h_i(0)$ to P_k, where P_k is not in the $list_i$.

2. After receiving $h_i(0)$, P_k computes a polynomial $f_k(0, y)$ such that $f_k(0, \omega^i) = h_i(0)$ for at least $n - b$ of the data he received.

3. P_k computes and output $s' = f_k(0,0)$.

We prove that the protocol is an unconditionally secure VSS in the following theorem.

Theorem 3.1 *The above scheme is an unconditionally secure verifiable secret sharing scheme.*

Proof We prove that the above scheme satisfies the conditions of the VSS as follows.

1. If a good player P_i output $ver_i = 0$, then $h_i(\omega^l) \neq h_l(\omega^i)$ for at least $b+1$ values of l. Since we assume that at least $n-b-1$ other players are good, $h_i(x)$ is not consistent with at least one good player. On the other hand, if the good player P_i's share is incorrect, then his share is consistent with at most $t-1$ good players, since shares lie on a polynomial of degree $\leq t-1$. Since $b+t-1 < n-b-1$, P_i will output $ver_i = 0$.

2. If the dealer is good, then the good player receives $f(x, \omega^i)$. Since $f(x,y)$ is symmetric, $f(\omega^l, \omega^i) = f(\omega^i, \omega^l)$ for all good players P_l. Thus $ver_i = 1$.

3. If all the good players P_i output $ver_i = 1$, then all of their shares are consistent with each other, since $n-b-1 > b$. Therefore they will construct the same polynomial in the *Reconstruct* protocol, when $n \geq t+2b$. We can use the algorithms from [24] to do this efficiently. If the dealer is good, then a good player P_k will have $f_k(0,y) = f(0,y)$.

4. Suppose the adversary controls b shares. Without loss of generality, we assume that he knows $h_1(x), h_2(x), \cdots, h_b(x)$. It is easy to show (see, e.g., [5]) that for any value $s' \in GF(q)$, we can find $b_{ij} \in GF(q)$, where $b_{00} = s', b_{ij} = b_{ji}, 0 \leq i,j \leq t-1$ such that if

$$f'(x,y) = \sum_{i=0}^{t-1}\sum_{j=0}^{t-1} b_{ij}x^i y^j,$$

then $f'(x, \omega^k) = h_k(x)$ for $k = 1, 2, \cdots, b$. □

Remark. This scheme is modified from Blom's key predistribution scheme (see [5] for the details). For simplicity, our description used a Reed-Solomon code instead of general MDS codes. It is straightforward to generalize our scheme by using MDS codes.

3.3 An example

We display a toy example in this subsection. Let $q = 13, \omega = 2, n = 8, t = 3$ and $b = 2$. Suppose the dealer D is good. First D selects a polynomial as follows:

$$f(x,y) = 3 + 9x + 2x^2 + 9y + 2y^2 + 8xy + 11xy^2 + 11x^2y + 4x^2y^2.$$

Then D sends the vector (v_1, v_2, v_3) to the players as follows, each of which determines a polynomial $v_1 + v_2x + v_3x^2$:

$$\begin{aligned}
h_1 &\longleftarrow (3,4,1)\\
h_2 &\longleftarrow (6,9,6)\\
h_3 &\longleftarrow (8,10,8)\\
h_4 &\longleftarrow (9,2,6)\\
h_5 &\longleftarrow (12,11,4)\\
h_6 &\longleftarrow (9,12,8)\\
h_7 &\longleftarrow (6,11,9)\\
h_8 &\longleftarrow (12,10,10)
\end{aligned}$$

Suppose that P_1 received a wrong polynomial, say, $h_1'(x) = 2 + 6x + 3x^2$. Assume that P_i, $1 \le i \le 6$, are good players. P_1 will receive the values $9, 8, 11, 11, 0$ from P_2, P_3, P_4, P_5, P_6 respectively and find $h_1'(\omega^4) = 8 \ne 11$, $h_1'(\omega^5) = 3 \ne 11$ and $h_1'(\omega^6) = 12 \ne 0$. Note that there are at most two good players' data which will agree with $h_1'(x)$, since the degree of $h_1'(x)$ is two. So P_1 will output $ver_1 = 0$, since P_1 has at most four values agreeing with $h_1'(x)$.

If there are at most two bad players, then each good player's share will be confirmed by the other 5 good players, so they will all output 1 at the end of $Share$.

At the *Reconstruct* phase, there are at most two incorrect values in the total eight values, so the good player can compute the secret value using the methods of [24].

3.4 VSS without dealer

Secret sharing without dealer means that there is no dealer in the scheme, who knows and distributes the secret. Secret sharing without dealer is first considered in [16]. One such secret sharing scheme is considered in [19].

We can remove the dealer from our scheme as follows. The other properties of the scheme are the same as in the previous subsection.

Share

1. Each P_k chooses an independent random symmetric polynomial

$$f^{(k)}(x, y) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} a_{ij} x^i y^j,$$

   where $a_{00} = s_k$ and $a_{ij} = a_{ji}$ for all i, j. Then P_k sends $h_l^{(k)}(x) = f^{(k)}(x, \omega^l)$ to P_l through a private channel.

2. After receiving $h_l^{(k)}(x)$, each P_l sends $h_l^{(k)}(\omega^m)$ to P_m for $1 \le m \le n$ through a private channel.

3. The P_m checks whether $h_m^{(k)}(\omega^l) = h_l^{(k)}(\omega^m)$ for $1 \le l \le n$. If P_m finds that for at least $b + 1$ values of l, $h_m^{(k)}(\omega^l) \ne h_l^{(k)}(\omega^m)$, then P_m broadcasts an accusation of P_k and put k in a list of rejected shares $list_m$.

4. Each player P_m makes a list $\mathcal{L}$ which contains all l such that P_l is accused by at least $b + 1$ participants of the system and defines the share

$$h_m = \sum_{l \notin \mathcal{L}} h_m^{(l)}(x)$$

5. If $list_m \subseteq \mathcal{L}$, then P_m outputs $ver_m = 1$. Otherwise, P_m refuses the shares and outputs $ver_m = 0$.

The reconstruct phase is the same as the previous scheme. Note that in this scheme the shared secret is

$$s = \sum_{i \notin \mathcal{L}} s_i.$$

In this scheme, each player in turn plays the part of the dealer. Thus the security of scheme follows from Theorem 3.1. We need only to show that each good player has the same list $\mathcal{L}$, which is obvious.

Remark. As we indicated before, our VSS is modified from Blom's key predistribution scheme. In the original scheme, there is a dealer to construct the schemes. Using the methods of this section, we obtain a key predistribution scheme without dealer.

4 New proactive scheme

In this section, we describe our proactive secret sharing scheme without combinatorial structure. We will add combinatorial structures in this scheme to improve the efficiency of the scheme in next section.

4.1 Initialization

In the initial step, we assume that there is a dealer who set up the scheme. After the initialization phase, the dealer will no longer be needed.

In the initialization, we use the *share* phase of the VSS described in the Section 3, but we assume that $t > b + 1$. The first two steps are the same. Then we do the following.

3′. Each P_l checks whether $h_k(\omega^l) = h_l(\omega^k)$ for $1 \le k \le n$. If P_l finds that for at least $n - b - 1$ of the $n - 1$ values of k, $h_k(\omega^l) = h_l(\omega^k)$, then P_l accepts the shares and outputs $ver_l = 1$, and P_l also broadcasts accusations against all the P_k for those k which causes $h_k(\omega^l) \ne h_l(\omega^k)$ in a $list_l$. Otherwise, P_l refuses the shares and outputs $ver_l = 0$.

4. If at least $n - b$ of the servers output "1", then the dealer erases all the information about the scheme. Otherwise, the dealer reboots the whole system and initializes the system again.

After the system is successfully initialized, the system will determine the bad servers as follows. A server will be determined to be bad if and only if it is accused by at least $n - b$ players in the system. Thus at the end of the initialization phase, every good server holds the same list of bad servers $\mathcal{L}$. Note that some accusations might be made by bad servers, but we assume that $n - b$ of the servers are good.

4.2 Share renewal

In the share renewal phase, all good servers do the following:

1. Each server P_l selects a random symmetric polynomial
$$r^{(l)}(x, y) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} r_{ij} x^i y^j,$$
where $r_{00} = 0$ and $r_{ij} = r_{ji}$ for all i, j.

2. P_l sends $h_k^{(l)}(x) = r^{(l)}(x, \omega^k)$ to P_k for $k = 1, 2, \cdots, n$ by a private channel and broadcasts $h_0^{(l)}(x) = r^{(l)}(x, 0)$.

3. P_k checks whether $h_0^{(l)}(0) = 0$ and $h_k^{(l)}(0) = h_0^{(l)}(\omega^k)$. If the conditions are satisfied, then P_k computes and sends P_m the value $h_k^{(l)}(\omega^m)$. Otherwise P_k broadcasts an accusation of P_l.

4. P_m checks whether $h_m^{(l)}(\omega^k) = h_k^{(l)}(\omega^m)$ for the values of l not accused by $n - b$ servers of the system. If the equation is not true for more than b values of k, then P_m 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 that P_l is accused by, P_l broadcasts $h_i^{(l)}(x)$. Then server P_k checks whether $h_i^{(l)}(\omega^k) = h_k^{(l)}(\omega^i)$ 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 bad servers $\mathcal{L}$ by including all values l for which P_l is accused by at least $b + 1$ servers, or found bad in the previous step. Then P_m updates its shares as
$$h_m(x) \longleftarrow h_m(x) + h_m^{(k)}(x)$$
for all $k \notin \mathcal{L}$.

Remark. We can remove the private channels in step 2, since our scheme is also a key predistribution scheme and the server P_i and P_j can use $h_i(j) = h_j(i)$ as a key to communicate securely.

To check the security of the renewal phase, first we note that any coalition of at most b servers cannot get any information about any shares except their own. In fact, a server P_i only knows $h_i^{(l)}(x)$ and $h_0^{(l)}(x)$. Since $b < t - 1$, the coalition of b servers knows at most $t - 1$ polynomials which cannot reveal $r^{(l)}(x, y)$ (see, e.g., [5]). Secondly, from the protocol we know that every good server should have the same list $\mathcal{L}$. Therefore, the good servers will keep consistent shares after renewal.

Note that a good server P_l can be accused by at most b servers. In this case, P_l will broadcast b polynomials in its defense. Thus P_l will broadcasts total $b + 1$ polynomials. Since $t > b + 1$, these information will not reveal $r^{(l)}(x, y)$. On the other hand, suppose P_l gives P_i a wrong share, i.e., the share P_i received is not consistent with at least $\frac{n-b}{2}$ other servers (the majority of good servers). Then P_i will accuse P_l in step 4, since $\frac{n-b}{2} > b$. If P_l broadcasts a correct share in the defence, then P_i can correct his share. Otherwise, P_l will be found to be bad.

4.3 Recover a share

When a server is corrupted or replaced, it needs to be rebooted and thus it needs to recover the secret shares.

We first provide a protocol, to detect the corrupted servers, which we call detection.

Detection

1. P_l computes and sends $h_l(\omega^k)$ to P_k for $k = 1, 2, \cdots, n$ by private channels.
2. P_k checks whether $h_l(\omega^k) = h_k(\omega^l)$. P_k then broadcasts an accusation $list_k$ which contains those l such that $h_l(\omega^k) \neq h_k(\omega^l)$ or $h_l(\omega^k)$ was not received.
3. Each good server updates the list $\mathcal{L}$ so that it contains those l accused by at least $b+1$ servers of the system.

After running *Detection*, the system will recover the shares for all server P_l, where $l \in \mathcal{L}$. The recovery protocol is as follows.

1. For each $l \in \mathcal{L}$, every good server P_i computes and sends $h_i(\omega^l)$ to P_l.
2. Upon receiving the data, P_l computes a polynomial $h_l(x)$ such that $h_l(\omega^k) = h_k(\omega^l)$ for the majority of k it received, using the algorithms of [24]. P_l sets $h_l(x)$ as its shares.

4.4 Reconstruct the secret

The reconstruction protocol is similar to the *Reconstruction* of VSS introduced in Section 3. We need only to change the first step as follows.

1' For each good server P_i, P_i sends $h_i(0)$ to P_k, where k is not in the list $\mathcal{L}$.

5 Combinatorial structure

In this section, we will introduce some combinatorial structure into our scheme. The combinatorial structure provides a predetermined arrangement of the servers which permits the possibility of reducing the computation of the scheme.

5.1 Set systems

A set system is a pair $(X, \mathcal{B})$, where X is a set of n points and $\mathcal{B}$ is a collection of subsets of X called blocks.

We will use a set system with the following properties, where $t \leq \frac{n}{2} - 1$:

1. $|B| \geq t$ for any $B \in \mathcal{B}$.
2. For any subset $F \subset X$ with $|F| \leq t-1$, there exists a $B \in \mathcal{B}$ such that $F \cap B = \emptyset$.

It is easy to see that such a set system exists. For example, we can choose $\mathcal{B}$ to be all the t-subsets of X. However, there are often better set systems (i.e., set systems containing fewer blocks). The following definition is from [25].

Definition 5.1 *A set system* $(X, \mathcal{B})$ *is an* (i,j)-disjunct system *provided that, for any* $P, Q \subseteq X$ *such that* $|P| \leq i$, $|Q| \leq j$ *and* $P \cap Q = \emptyset$, *there exists a* $B \in \mathcal{B}$ *such that* $P \subseteq B$ *and* $Q \cap B = \emptyset$. *An* (i,j)*-disjunct system,* $(X, \mathcal{B})$*, will be denoted as an* (i,j)*-DS*(n,s) *if* $|X| = n$ *and* $|\mathcal{B}| = s$.

It is easy to see that we can use $(t, t-1)$-DS(n,s) as our needed set system. [25] provided some efficient explicit constructions for (i,j)-DS(n,s) in which s is $O((ij)^{\log^*(n)}(\log n))$.

5.2 Applying set system to the proactive VSS

Now we will use the set system to improve our proactive scheme. Suppose $(X, \mathcal{B})$ is a set system satisfying the conditions of subsection 5.1, where $X = \{1, 2, \cdots, n\}$, and $\mathcal{B} = \{B_1, B_2, \cdots, B_s\}$. The set system is published so that each participant can consult it.

The idea of using the set system is to reduce the computations for the share renewal and share recover protocols. In the scheme of Section 4, share renewal and share recover used the data from all the participants. However, these operations can be carried out using the data from t good servers.

Note that in our scheme, in any phase there is a list $\mathcal{L}$ containing all the bad servers. By the property of the set system, there is a block B which contains only good servers. If the system can determine one of the "good" blocks, then the system can renew the shares or recover the shares only using the data from these servers. We will call these servers the members of an *executive committee*.

For a list $\mathcal{L}$ of bad servers, the system can decide following list of blocks: $B_{i_1}, B_{i_2}, \cdots, B_{i_e}$, such that $B_{i_j} \cap \mathcal{L} = \emptyset, j = 1, 2, \cdots, e$ and $1 \leq i_1 < i_2 < \cdots < i_e \leq s$. These blocks are called *executive committee candidates*. Note that the adversary is mobile, therefore we cannot guarantee that these candidates contain only good servers in the next time period.

The proactive secret sharing scheme with combinatorial structure works as follows. The initialization is the same as that in Section 4. In each time period the system does the following.

1. Run *Detection* to obtain the list $\mathcal{L}$ of bad servers and the executive committee candidates: $B_{i_1}, B_{i_2}, \cdots, B_{i_e}$.
2. If an executive committee has not been found, then for next executive committee candidate B, each $P_g \in B$ does the following:
   (a) Selects a random symmetric polynomial
   $$r^{(g)}(x,y) = \sum_{i=0}^{t-1}\sum_{j=0}^{t-1} r_{ij}x^i y^j,$$
   where $r_{00} = 0$ and $r_{ij} = r_{ji}$ for all i, j, and sends $h_k^{(g)}(x) = r^{(g)}(x, \omega^k)$ to P_k for $k = 1, 2, \cdots, n, k \neq g$ by private channel and broadcasts $h_0^{(g)}(x) = r^{(g)}(x, 0)$.
   (b) P_k checks whether $h_0^{(g)}(0) = 0$ and $h_k^{(g)}(0) = h_0^{(g)}(\omega^k)$ for $g \in B$. If the conditions are satisfied, then P_k computes and sends P_m the value $h_k^{(g)}(\omega^m)$. Otherwise P_k broadcasts an accusation of P_g.

(c) P_m checks whether $h_m^{(g)}(\omega^k) = h_k^{(g)}(\omega^m)$ for $g \in B$. If the equation is not true, then P_m broadcasts an accusation of P_g.

(d) A member in B is accused by at least $b+1$ servers is bad. If a member in B is accused by at most b servers, then it can defend itself. If no member in B is bad, then B is found to be the executive committee.

3. The system runs the recovery protocol to recover the shares for the servers in $\mathcal{L}$.

4. Each server P_m updates its shares as

$$h_m(x) \longleftarrow h_m(x) + h_m^{(g)}(x)$$

for all $g \in B$.

The reconstruction protocol is the same as that in Section 4.

5.3 Applying combinatorial structures to other schemes

The proactive secret sharing scheme proposed by Herzberg et al. in [15] is similar to our scheme in Section 4. Thus it is straightforward to modify our method with combinatorial structures to their scheme. In general, suppose a proactive secret sharing scheme has the following properties:

1. Information from any t good servers can be used to renew shares and recover shares.
2. A VSS exists in which any server can use this VSS to send data which can be verified by the system.
3. There is a detection protocol to find the bad servers.
4. There is a defence protocol so that an accused server can be determined bad or good by the system.
5. There are renewal and share recovery protocols.
6. There is set system $(X, \mathcal{B})$ satisfying the conditions of Subsection 5.1.

Then we can use the following scheme for renewal and share recovery protocols.

1. Run the detection protocal to obtain a list $\mathcal{L}$ of bad servers and the executive committee candidates: $B_{i_1}, B_{i_2}, \cdots, B_{i_e}$.
2. If executive committee has not been found, then for next executive committee candidate B, each $P_g \in B$ does:

   (a) Send recovery information rc_k^g to P_k for each $k \in \mathcal{L}$ in the system and send renewal information rn_l^g to P_l for each l in the system by VSS.

   (b) The system checks the correctness of rc_k^g and rn_l^g. If some mistake is found, then P_g is accused.

(c) A member in B is accused, then it can defend itself and the system can decide whether it is bad. If no member in B is bad, then B is defined to be the executive committee.

3. The $P_k \in \mathcal{L}$ recovers its share using $\{rc_k^g : g \in B\}$.

4. Each server P_l renews its share using $\{rn_l^g : g \in B\}$.

References

[1] N. Alon, Z. Galil and M. Yung, Efficient dynamic-resharing "verifiable secret sharing" against mobile adversary, European Symposium on Algorithms (ESA) 95, LNCS 979, 523-537.

[2] J. C. Benaloh, Secret sharing homomorphisms: keeping shares of a secret secret, Advances in Cryptology-Crypto'86, LNCS 263, 1987, 251-260.

[3] M. Ben-Or, S. Goldwasser and A. Wigderson, Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations, Proc. 20th Annual Symp. on the Theory of Computing, ACM, 1988, 1-10.

[4] G.R. Blackley, Safeguarding cryptographic keys. Proc. Nat. Computer Conf. AFIPS Conf. Proc., 1979, 313-317.

[5] R. Blom, An optimal class of symmetric key generation systems, Eurocrypt'84, LNCS 209, (1985), 335-338.

[6] R. Canetti and A. Herzberg, Maining security in the presence of transient faults, Crypto'94, Springer-Verlag, 1994.

[7] D. Chaum, C. Crepeau and I. Damgard, Multiparty Unconditionally Secure Protocols, Proc. 20th Annual Symp. on the Theory of Computing, ACM, 1988, 11-19.

[8] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, Verifiable Secret Sharing and Achieving Simultaneity in Presence of Faults, Proc. 26th Annual Symp. on the Foundations of Computing Science, IEEE, 1985, 383-395.

[9] P. Feldman, A Practical Scheme for Non-Interactive Verifiable Secret sharing, Proc. 28th Annual Symp. on the Foundations of Computing Science, IEEE, 1987, 427-437.

[10] P. Feldman and S. Micali, An Optimal Algorithm for Synchronous Byzantine Agreement, Proc. 20th Annual Symp. on Theory of Computing, ACM, 1988, 148-161.

[11] Y. Frankel, P. Gemmel, P. D. MacKenzie and M. Yung, Proactive RSA, Crypto'97, LNCS 1294, 440-452.

[12] R. Gennaro, M. O. Rabin and T. Rabin, Simplified VSS and fast-track multiparty computations with applications to threshold cryptography, Proc. of 17th ACM Symp. on Principles of Distributed Computing, (1998), 101-111.

[13] O. Goldreich, S. Micali and A. Wigderson, Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems, Journal of the ACM, 38(1991), 691-729.

[14] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk and M. Yung, Proactive public key and signature systems, The 4th ACM Symp. on Comp. and Comm. Security, April 1997.

[15] A. Herzberg, S. Jarecki, H. Krawczyk and M. Yung, Proactive secret sharing or: How to cope with perpetual leakage, Crypto'95, 339-352.

[16] I. Ingemarsson and G. J. Simmons, A protocol to set up shared secret schemes without the assistance of a mutually trusted party, Eurocrypt'90, LNCS 473, 1990, 266-282.

[17] R. J. McEliece and D. V. Sarwte, On Sharing Secrets and Reed-Solomon Codes, Communications of the ACM, 24(1981), 583-584.

[18] R. Ostrovsky and M. Yung, How to withstand mobile virus attacks, ACM Symposium on principles of distributed computing, 1991, 51-59.

[19] T. P. Pedersen, Non-interactive and information-theoretic secret sharing, Advances in Cryptology - Crypto'91, LNCS 576, 1991, 129-140.

[20] T. Rabin, Robust sharing of secrets when the dealer is honest or faulty, Journal of the ACM, 41(1994), 1089-1109.

[21] T. Rabin, A simplified approach to threshold and proactive RSA, Crypto'98, LNCS 1462, 1998, 89-104.

[22] T. Rabin and M. Ben-Or, Verifiable secret sharing and multiparty protocols with honest majority, Proc. 21st Annual Sympo. on the Theory of Computing, ACM, 1989, 73-85.

[23] A. Shamir, How to share a secret, Commun. ACM, 22(1979), 612-613.

[24] D. R. Stinson and R. Wei, Determining the maximum consistent set of shares in a threshold scheme, submitted.

[25] D. R. Stinson, R. Wei and L. Zhu, New constructions for perfect hash families and related structures using combinatorial designs and codes, J. Combinatorial Designs, to appear.

[26] M. Tompa and H. Woll, How to share a secret with cheaters, Journal of Cryptology, 1(1988), 133-138.