Construction of a Secret Sharing Scheme with Multiple Extra Functionalities

by

Ivan Minevskiy
Honours Student

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE

in

The Faculty of Science

(Department of Computer Science)

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
March 2004
© Ivan Minevskiy, 2004
Abstract

In many circumstances, secret sharing has to provide more flexibility and functionality than the fundamental \((t,n)\) threshold schemes. The requirements for different extra functionalities of a secret sharing scheme are often contradictory to each other, which makes construction of a scheme with several additional features a challenge.

We will introduce a basic threshold secret sharing scheme and will incrementally add several extra functionalities to it. Then we will critically analyze the strengths and weaknesses introduced into the scheme by each of these features. The added extra functionalities will allow to renew and recover shares, and to enroll/disenroll shareholders. We will discuss a method for upgrading the threshold access structure of a scheme to a general access structure.
## Contents

Abstract........................................................................................................................................ ii

Contents.......................................................................................................................................... iii

1. Introduction ................................................................................................................................... 1

2. Overview of area............................................................................................................................. 2

   2.1 Threshold schemes and common definitions ............................................................................. 2
   2.2 Shamir’s (t,n) secret sharing scheme ......................................................................................... 4
       2.2.1 Properties of Shamir’s threshold scheme ........................................................................... 5
   2.3 Blakley’s secret sharing scheme .................................................................................................. 5
   2.4 Generalized secret sharing schemes and access structures ....................................................... 6
   2.5 Cheater detection ..................................................................................................................... 8
   2.6 Secret sharing schemes with extended capabilities ................................................................. 8

3. Proposed secret sharing scheme ................................................................................................... 9

   3.1 Secret sharing and reconstruction .............................................................................................. 9
       3.1.1 Secret sharing phase ........................................................................................................... 9
       3.1.2 Secret reconstruction phase ................................................................................................... 9
       3.1.3 Discussion ....................................................................................................................... 10
   3.2 Renewal of shares ..................................................................................................................... 10
       3.2.1 Discussion ....................................................................................................................... 11
   3.3 Enrollment of new shares ......................................................................................................... 11
       3.3.1 Discussion ....................................................................................................................... 12
   3.4 Recovery of lost shares ............................................................................................................. 12
   3.5 Disenrollment of shares ............................................................................................................ 13
   3.6 General access structure ......................................................................................................... 13
   3.7 Comparison of the proposed scheme with Shamir’s and Blackley’s ........................................ 16
       3.7.1 Proposed scheme ............................................................................................................. 16
       3.7.2 Shamir’s scheme ............................................................................................................. 16
       3.7.3 Blakley’s scheme ............................................................................................................ 17
       3.7.4 Conclusion ..................................................................................................................... 18

Appendix: User Manual ................................................................................................................... 19

   1. SSSDealer .................................................................................................................................... 19
       1.1 Secret construction ................................................................................................................ 19
       1.2 Secret reconstruction ............................................................................................................ 21
       1.3 Renewal of shares ................................................................................................................ 22
       1.4 Enrollment of new shareholders .......................................................................................... 23
       1.5 Disenrollment of shareholders ............................................................................................ 24
       1.6 Utilities for TEXT<>NUMBER conversions ........................................................................ 25
   2. SSSClient ..................................................................................................................................... 26
       2.1 Loading shares ..................................................................................................................... 26
       2.2 Renewal of shares ................................................................................................................ 26
       2.3 Participating in enrollment .................................................................................................. 27
   3. About ......................................................................................................................................... 27

Bibliography ..................................................................................................................................... 28
1. Introduction

Secret sharing is concerned with the problem of how to distribute a secret among a group of participating individuals, or shareholders, so that only a pre-defined composition of shareholders are capable of reconstructing the secret by combining the parts of the secret, or shares, distributed to them. There are many practical applications of secret sharing schemes. For example, to transfer money from a bank a manager and a clerk need to cooperate. Perhaps, a ballistic missile should be launched only if three officers authorize the action. In this way the authority to initiate an action, e.g. the launch of a missile, is divided for the purpose of greater security.

In communications networks that require security, it is important that secrets be protected by more than one key. Furthermore, a system of several keys that can be combined in multiple ways may allow for the recovery of a unique secret regardless of how they are combined. Schemes that have a group of participants that can recover a secret are known as Secret Sharing Schemes.

The original motivation for secret sharing was the following. Backup copies are created to protect cryptographic keys from loss or corruption. The greater the number of copies made, the greater the risk of security exposure; the smaller the number, the greater the chance that all of them are lost. Secret sharing schemes allow improving the level of protection without increasing the risk of exposure.
2. Overview of area

The idea of secret sharing is to start with a secret, divide it into pieces called shares, which are then distributed among participating individuals by the dealer. Only certain groups (authorized subsets of participants) can reconstruct the original secret. More formally, a Secret Sharing Scheme (SSS) is a method whereby \( n \) pieces of information, called shares or shadows, are assigned to a secret key \( K \) in such a way that

1. the secret key can be reconstructed from certain authorized groups of shares, and
2. the secret key cannot be reconstructed from unauthorized groups of shares.

Further in this paper we are going to denote participating individuals as shareholders, participants, or players interchangeably.

2.1 Threshold schemes and common definitions

The first threshold schemes were independently invented by both Adi Shamir[10] and George Blackley[2] in 1979. Shamir[10] states that threshold schemes can be very helpful in the management of cryptographic keys. To protect data, we would encrypt it. However, to protect the encryption key, we need a different method. The most secure key management scheme keeps the key in a single place. This sort of scheme may not always be appropriate. For instance, in a single case of misfortune the key may be rendered inaccessible. An obvious solution to this may be to make multiple copies of the key. This, however, also increases the risk associated with keeping multiple keys secret. By using Shamir’s [10] threshold scheme concept, we can get a very robust key management scheme.

Threshold schemes are ideally suited to situations where a group of mutually suspicious individuals with conflicting interests must cooperate [10].

We will now use the definition outlined in [14] to define a threshold secret sharing scheme.

**Definition 2.1.** Let \( t \) and \( n \) be positive integers, \( t \leq n \). A \((t,n)\)-threshold scheme is a method of sharing a secret, \( S \), among a set of \( n \) participants (denoted by \( P \)), in such a way that any \( t \) participants can compute the value of \( S \), but no group of \((t - 1)\) participants can do so.

The values of shares are chosen by a special participant, which is referred to by [14] as the dealer. When dealer wants to share \( S \) among the participants in \( P \), she gives each participant some partial information referred to earlier as a share. The shares should be distributed secretly, so no participant knows the share given to any other participant. At some later time, a subset of participants \( B \subseteq P \) will pool their shares in an attempt to compute \( S \). Alternatively they could give
their shares to a trusted authority, or *combiner*, who will perform the computation on their behalf. It is assumed that the combiner is an algorithm which only performs the task of reconstructing the secret. If $|B| \geq t$, then they should be able to compute the value of $S$ as a function of the shares they collectively hold. Furthermore, if $|B| < t$, then they should obtain no information about the value of $S$.

**Definition 2.2.** A *perfect* threshold scheme is a threshold scheme in which knowing only $(t - 1)$ or fewer shares reveals no information about $S$ whatsoever, in the information-theoretic sense [8].

The efficiency of a secret sharing scheme is measured by its information rate. The information rate was studied by Brickell and Stinson [3]. It is a measure of the amount of information that the participants need to keep secret in a secret sharing scheme.

**Definition 2.3.** The *information rate* for a particular shareholder is the bit-size ratio (size of the shared secret) / (size of that user’s share). The information rate for a secret sharing scheme itself is the minimum such rate over all participants [8].

In any perfect secret sharing scheme the following holds for all shares: (size of a share) $\geq$ (size of the shared secret) [8]. Hence, every perfect secret sharing scheme must have an information rate $\leq 1$. If any user $P_i$ had a share of bit-size less than that of the secret, knowledge of the shares (excepting that of $P_i$) corresponding to any authorized subset to which $P_i$ belonged would reduce the uncertainty in the secret to at most that in $P_i$’s share. Thus, by definition, the scheme would not be perfect.

**Definition 2.4** Secret sharing schemes with information rate 1 (see Definition 2.3) are called *ideal* [8].

For example, Shamir’s threshold scheme is an ideal secret sharing scheme. It will be described in details in the next section.

Define an *access structure* to be a set of all subsets of participants able to reconstruct the secret. Besides $(t,n)$-threshold access structures, more general access structures exist in secret sharing. These will be considered in Section 2.4. General access structures are useful in situations where the trust-status of the shareholders is not uniform. For example, in the bank scenario described earlier, it might be considered more secure to authorize either the bank manager, or any two out of three senior employees to transfer money.
2. Overview of area

2.2 Shamir’s \((t,n)\) secret sharing scheme

Shamir’s [10] \((t,n)\)-threshold scheme was published in 1979. It is based on Lagrange polynomial interpolation. For an integer \(m > 1\), we denote by \(\mathbb{Z}_m\) the set \(\{0, 1, \ldots, m - 1\}\). All arithmetic operations over \(\mathbb{Z}_m\) are done \textit{mod} \(m\). All further calculations in this paper are done in \(\mathbb{Z}_p\) where \(p\) is a prime number bigger than the secret.

A \((t,n)\)-threshold Shamir’s scheme is constructed by the trusted party (dealer) Tom. He begins with a secret integer \(S \geq 0\) that he wishes to distribute among \(n\) shareholders. Next he chooses a prime \(p > \max(S,n)\), defines \(a_0 = S\), and selects \((t-1)\) random, independent coefficients \(a_1, \ldots, a_{t-1}\), \(0 \leq a_j \leq p-1\), defining a random polynomial over \(\mathbb{Z}_p\),

\[
    f(x) = \sum_{j=0}^{t-1} a_j x^j
\]  

(2.1)

Thus, \(f(x)\) is of degree at most \((t-1)\). Now Tom chooses \(n\) distinct public points \(x_i \in \mathbb{Z}_p\), \(1 \leq i \leq n\), computes shares \(S_i = f(x_i) \mod p\), \(1 \leq i \leq n\), and securely transfers every share \(S_i\) to its shareholder, along with public index \(i\). The secret is \(S = f(0)\).

When \(t\) participants agree to cooperate, the combiner Clara takes their shares and tries to recover the secret polynomial \(f(x)\). Shares provide \(t\) distinct points \((x_i, S_i)\) on the curve \(f(x)\). With these points the following system of equations can be constructed:

\[
    \begin{align*}
    S_1 &= a_0 + a_1 x_1 + \ldots + a_{t-1} x_1^{t-1} \\
    S_2 &= a_0 + a_1 x_2 + \ldots + a_{t-1} x_2^{t-1} \\
    &\vdots \\
    S_t &= a_0 + a_1 x_t + \ldots + a_{t-1} x_t^{t-1}
    \end{align*}
\]  

(2.2)

Since the Vandermonde determinant \(\Delta \neq 0\), the system (2.2) has a unique solution for \((a_0, \ldots, a_{t-1})\)

\[
    \Delta = \begin{vmatrix}
    1 & x_1 & \cdots & x_1^{t-1} \\
    1 & x_2 & \cdots & x_2^{t-1} \\
    \vdots & \vdots & \ddots & \vdots \\
    1 & x_t & \cdots & x_t^{t-1}
    \end{vmatrix}
\]  

(2.3)

The Lagrange interpolation formula allows to determine the polynomial \(f(x)\) of degree \((t-1)\) from \(t\) distinct points \((x_i; S_i)\), thus
2. Overview of area

The Lagrange Interpolation Formula can be simplified to a linear expression because we are only interested in the constant term of the polynomial. Recall that \( S = a_0 = f(0) \). Then the shared secret \( S \) can be written as

\[
S = \sum_{i=1}^{t} c_i S_i, \quad \text{where} \quad c_i = \prod_{1 \leq j \neq i} \frac{x_j}{x_j - x_i}
\]

If Clara knows only \((t - 1)\) shares, she cannot find the unique solution for \( S = a_0 \) because the system (2.2) contains \((t - 1)\) equations with \( t \) unknowns. Moreover, each group member may compute \( S \) as a linear combination of \( t \) shares \( S_i \), since the \( c_i \) are non-secret constants and may be pre-computed.

### 2.2.1 Properties of Shamir’s threshold scheme

Shamir’s secret sharing scheme has the following properties [8]:

1. **perfect.** Given knowledge of any \( t - 1 \) or fewer shares, all values \( 0 \leq S \leq p - 1 \) of the shared secret remain equally probable (see Definition 2.2).
2. **ideal.** The size of one share is the size of the secret (see Definition 2.4).
3. **extendable for new users.** New shares may be computed and distributed without affecting existing shares.
4. **no unproven assumptions.** Unlike many cryptographic schemes, its security does not rely on any unproven assumptions (e.g., about the difficulty of number-theoretic problems).

### 2.3 Blakley’s secret sharing scheme

George Blakley independently developed a secret sharing scheme equivalent to Shamir’s and it was also published in 1979. Instead of polynomial interpolation Blakley used geometric approach based on hyperplanes in \( n \) dimensional space as his implementation vehicle. And he used the term “shadow” to refer to the shared representations of the secret value, rather than “share”, as used by Shamir. Shadows were represented by equations of hyperplanes, each pair of which intersected in a line and finally in a point in \( n \) dimensional space, representing the secret value.

In Blakley’s scheme the secret is a point in \( k \)-dimensional space and \( n \) shadows are constructed, with each shadow defining an affine hyperplane in this space. An affine hyperplane
is the set of solutions $x = (x_1, \ldots, x_k)$ to an equation of the form $a_1x_1 + \ldots + a_kx_k = b$. By finding the intersection of any $k$ of these planes, the secret (point of the intersection) can be obtained.

For example, in a $(2,n)$-threshold Blakley’s scheme each shadow is a line.

Simmons [11] does not consider Blakley’s scheme completely equivalent to Shamir’s because it’s not perfect (knowledge of some shadows whose number is less than $k$ narrows the range of possible values that the secret could take). Although, Blakley’s scheme can be modified to be perfect.

Blakley’s scheme has not achieved an acceptance level comparable to Shamir’s. One factor is the intuition support provided by Shamir’s polynomial model.

### 2.4 Generalized secret sharing schemes and access structures

In the outline of threshold schemes, we wanted $t$ out of $n$ participants to be able to determine the secret. A more general situation is to specify exactly which subsets of participants should be able to determine the secret and those that should not [14].

The generalized secret sharing scheme (GSS scheme for short) is a method that divides a secret into a set of participants such that any qualified subset of participants can reconstruct the secret, while any unqualified subsets cannot [1]. The family of qualified subsets of GSS scheme are denoted as positive access structure (access structure for short), and each element in the access structure is called access instance.

For example, let $P$ be the set of participants and denote by $\Gamma$ a set of subsets of $P$, that should be able to compute the secret. Then $\Gamma$ is the access structure, and the subsets in $\Gamma$ are called authorized subsets. Furthermore, if we let $X$ be the secret, and $S$ is the share set, we use the dealer $D$ to share $X$ by giving each player a share $s_i \in S$. Some time later a subset of players might attempt to determine $X$ from the shares they collectively hold.

**Definition 2.5** (Stinson, [14]). A perfect secret sharing scheme for the general access structure $\Gamma$, is a method of sharing a secret $S$ among a set of $n$ participants such that $P$ is the set of all participants, in such a way that the following two conditions are met:

- If an authorized subset of participants $B \subseteq P$ pool their shares, so that they can determine the value of $S$.
- If an unauthorized subset of participants $C \subseteq P$ pool their shares, then they can determine nothing about the value of $S$.

Notice that a $(t,n)$-threshold scheme describes the access structure $\{B \subseteq P \mid |B| \geq t\}$. This structure is referred to by Stinson [14] as the threshold access structure. In other words, the access structure of $(t,n)$-threshold schemes is
where $2^P$ is the class of all subsets of $P$ (or $2^P$ consists of all possible groups which can be created from $P$). The access structure in this case consists of all groups whose cardinality is at least $t$. Although, in general, secret sharing may have a lot more sophisticated access structure than a $(t,n)$-threshold one. Let again denote the set of all participants by $P = \{P_1, \ldots, P_n\}$. The class $2^P$ of all subsets of $P$ is split into two disjoint subclasses: the class $\Gamma$ of all authorized subsets of $P$, and the class $2^P \backslash \Gamma$ of all unauthorized subsets of $P$. Any authorized subset of participants is able to reconstruct the secret value $S$, while any unauthorized subset is not.

Benaloh and Leichter [1] observed that it is possible to create a SSS for any access structure as long as this access structure satisfies monotone property: if $B$ is an authorized subset $B \in \Gamma$ and $B \subseteq C \subseteq P$ then $C \in \Gamma$. In other words a superset of an authorized set is again an authorized set.

Notice, that among the subsets of $\Gamma$ it is easy to identify the minimal subsets. A subset $A \in \Gamma$ is minimal if for all $B \subset A$ the subset $B$ does not belong to the access structure $\Gamma$. The collection of all minimal subset of $\Gamma$ is called the access structure basis $\Gamma_0$ and

$$\Gamma_0 = \{A \in \Gamma \mid \forall B \subset A \ B \not\in \Gamma\}.$$  

$\Gamma$ is the closure of $\Gamma_0 : \Gamma = cl(\Gamma_0)$, i.e. $\Gamma$ is the intersection of all sets $F$ that contain $\Gamma_0$. For example, the basis $\Gamma_0$ of $(t,n)$-threshold scheme is

$$\Gamma_0 = \{ A \in 2^P : |A| = t \}.$$  

Ito et al. proposed the first GSS scheme in 1987 [6]. Their scheme assigned different numbers of shares to each participant. They also proposed a method to extend the scheme from access structure $\Gamma_1$ to $\Gamma_2$ for $\Gamma_1 \subset \Gamma_2$ such that each participant can use only one share during a reconstruction. However, Ghodosi proved that the Ito's scheme is not perfect, and their extended secret sharing scheme is not secure [4]. Benaloh and Leichter proved that not all GSS schemes could be realized if there is only one share assigned to each participant [2]. Lin and Harn proposed a GSS scheme where the honest participants can detect cheaters even if all of the other participants are corrupt. They also proposed a method to extend the lifetime of shares. The shares can be repeatedly used for $n$ times to obtain $n$ different secrets [5].
2. Overview of area

2.5 Cheater detection

The cheater problem is a serious obstacle for secret sharing schemes. A cheater is a qualified participant who possesses a true share, but releases a fake share or withholds a share during a reconstruction of the secret. If a cheater releases a fake share or withholds a share on secret reconstruction, then he/she can obtain the secret and exclude others. Thus, the cheater has an advantage over the other shareholders. Rabin et al. used an information checking protocol [9] to verify the validity of each share and thereby detect cheaters.

2.6 Secret sharing schemes with extended capabilities

Secret sharing schemes with a variety of extended capabilities exist, including:

1. pre-positioned secret sharing schemes. All necessary secret information is put in place excepting a single (constant) share which must later be communicated, e.g., by broadcast, to activate the scheme. The last share is expected to be unique, which results in a unique secret.

2. dynamic secret sharing schemes. These are pre-positioned schemes wherein the secrets reconstructed by various authorized subsets vary with the value of communicated activating shares. Thus, the resulting secret might be different depending on the value of the last share.

3. multi-secret threshold schemes. In these secret sharing schemes different secrets are associated with different authorized subsets.

4. detection of cheaters, and verifiable secret sharing. These schemes respectively address cheating by one or more shareholders, and the distributor of the shares (or the dealer).

5. proactive schemes. These schemes are helpful when the nature of a secret does not allow changing it – they allow renewing existing shares, recovering lost shares, enrolling and disenrolling participants.
3. Proposed secret sharing scheme

For the rest of this paper, we assume that the set of participants is \( P = \{ P_1, P_2, \ldots, P_n \} \), the access structure is \( \Gamma = \{ \Gamma_1, \Gamma_2, \ldots, \Gamma_m \} \), \( S \) is a secret integer of arbitrary precision, \( p \) is a prime larger than \( S \), and all arithmetic operations are performed \( \text{mod } p \). Assume that there are \( t \) shareholders in the access instance \( \Gamma_j \subseteq \Gamma \). That is if all \( t \) qualified participants in \( \Gamma_j \) release shares honestly then the secret can be reconstructed. Also assume the presence of a trusted party (dealer) or a piece of trusted software that can construct and reconstruct the secret.

3.1 Secret sharing and reconstruction

For access instance \( \Gamma_j \subset \{ 1, 2, \ldots, n \} \), the secret sharing and reconstruction phases are as follows.

3.1.1 Secret sharing phase

Assume that access structure is \((t,n)\)-threshold. The dealer first randomly chooses two vectors \( \overline{S} \) and \( \overline{X} \) of length \( t \) each, such that \( \overline{S} \cdot \overline{X} = s_1x_1 + s_2x_2 + \ldots + s_tx_t = S \). Vector \( \overline{S} \) is secret and will be destroyed after the construction phase is complete. Vector \( \overline{X} \) is public and will be known to every shareholder. Now the dealer generates \( n \) vectors \( \overline{Y}_i \), \( 1 \leq i \leq n \). These vectors are also public and each \( t \) of them are linear independent. We assume that we can come up with that many independent vectors \( \overline{Y}_i \) (one good way of doing this is by using Vandermonde determinant, but there are other techniques as well). The actual \( n \) private shares are computed as \( s_i = \overline{S} \cdot \overline{Y}_i \), \( 1 \leq i \leq n \). Finally, the dealer distributes the shares \( s_i \) to \( P_i \) through a secure channel. Note that, for security reason, it is better to destroy \( \overline{S} \) and \( S \) after this phase is finished.

3.1.2 Secret reconstruction phase

At least \( t \) out of \( n \) shares must be collected together before reconstruction process can begin. Assuming that \( a_1, a_2, \ldots, a_t \) are elements of the secret vector \( \overline{S} \) and \( y_{ij} \) is the \( j \)th element of the \( i \)th vector \( \overline{Y}_i \), the following system of \( t \) equations with \( t \) unknowns can be constructed

\[
\begin{align*}
    s_1 &= a_1y_{11} + a_2y_{12} + \ldots + a_ty_{1t} \\
    s_2 &= a_1y_{21} + a_2y_{22} + \ldots + a_ty_{2t} \\
    &\vdots \\
    s_t &= a_1y_{t1} + a_2y_{t2} + \ldots + a_ty_{tt}
\end{align*}
\] (3.1)
3. Proposed secret sharing scheme

From (3.1) vector \( \overrightarrow{S} = (a_1, a_2, \ldots, a_t) \) can be found. Recall that the actual secret is \( S = \overrightarrow{S} \cdot \overrightarrow{X} \), where \( \overrightarrow{X} \) is publicly known.

3.1.3 Discussion

The security of the secret is guaranteed by the dot product problem. The scheme is perfect (see Definition 2.2), because knowledge of \((t - 1)\) or fewer shares does not narrow the range of possible values that the secret could take, i.e. the range remains \(1 \leq S \leq n\). \((t - 1)\) shares can obtain the secret vector \( \overrightarrow{S} \) with only one degree of freedom, but the actual secret number \( S \) remains in the range \([1, n]\).

The scheme is ideal (see Definition 2.4), because the information rate (size of the shared secret) / (size of that user’s share) = 1.

3.2 Renewal of shares

Often the nature of a secret does not allow changing it (for example secret keys, documents, or data files). Renewal allows updating shares in the way such that the renewed shares combine to the same secret as original ones. Any information learned by the adversary about individual shares becomes obsolete after they are renewed. This feature is especially important for sensitive or long-lived secrets.

If more than \((n - t)\) shares are lost, then the secret cannot be recovered. If we assume that shares are being compromised (revealed or lost) gradually, then it is possible to divide the lifetime of the system into relatively short periods of time. At the beginning of each consecutive period, a share renewal protocol is run.

The share renewal protocol is run concurrently by all participants \( P_i; 1 \leq i \leq n \) at the beginning of each time period. First, dealer randomly generates a vector \( \overrightarrow{R} \) such that \( \overrightarrow{R} \cdot \overrightarrow{X} = 0 \). Recall that vector \( \overrightarrow{X} \) is public. Then vector \( \overrightarrow{R} \) is distributed to shareholders. Each participant \( P_i \) executes the following:

\[
s_i = s_i + \overrightarrow{R} \cdot \overrightarrow{Y}_i
\]

where \( s_i \) is \( i^{th} \) share and \( \overrightarrow{Y}_i \) is its corresponding vector. The system of equation in the reconstruction phase is now

\[
\begin{align*}
s_1 + \overrightarrow{R} \cdot \overrightarrow{Y}_1 &= a_1y_{11} + a_2y_{12} + \ldots + a_iy_{1t} \\
s_2 + \overrightarrow{R} \cdot \overrightarrow{Y}_2 &= a_1y_{21} + a_2y_{22} + \ldots + a_iy_{2t} \\
&\vdots \\
s_i + \overrightarrow{R} \cdot \overrightarrow{Y}_i &= a_1y_{i1} + a_2y_{i2} + \ldots + a_iy_{it}
\end{align*}
\]
This change in equations does not affect integer secret $S$, which is computed as $S = \overline{S} \cdot \overline{X}$, because $\overline{R} \cdot \overline{X} = 0$. Thus, the renewed shares combine to the same secret as original ones.

### 3.2.1 Discussion

The renewal feature does not require any modifications to the original scheme. In the same time it provides a great boost to the security of shares. It takes $O(t)$ time to construct vector $\overline{R}$ and $O(n)$ to distribute it to every shareholder. Then it takes $O(t)$ time to update a share, but, most likely, that will be done in parallel.

### 3.3 Enrollment of new shares

Sometimes access structure of a secret sharing scheme has to be updated to either include new shareholders, or to remove some existing shareholders. This section describes how to add enrollment capability to the proposed scheme. Disenrollment feature is discussed in section 3.5.

The difficulty of adding new shareholders to an access structure depends on the complexity of that structure. The proposed scheme has a $(t,n)$-threshold structure, although it can be extended to a general one. This paper will not describe how to add shareholders to a general access structure; the discussion is limited to a threshold scheme.

When constructed, the proposed scheme uses linear combination of $t$ initial shares to generate other $(n - t)$ shares. It is possible to use the same mechanism to generate any number of additional shares after a scheme is constructed. An important and hard to satisfy condition for that is cooperation of at least $t$ shareholders, whose shares are going to be used to compute linear combination of $t$ independent shares. We have to deal with the fact that no one wish to disclose secret shares to other people, even to the dealer. The other problem is cheating: some shareholders may submit modified shares which will lead to an incorrect new share.

The proposed scheme uses the following algorithm to produce a new share:

1. The dealer generates a random share. Recall, that every share is represented by a public vector of length $t$ and a private integer number. The public vector of this share is not necessarily independent of existing shares’ vectors.
2. The dealer generates $t$ constants (constants must be carefully chosen to ensure that the result of the linear combination obtained in step 6 of this algorithm is independent of already existing vectors $\overline{Y}_i$).
3. The dealer finds $t$ shareholders willing to participate in enrollment.
4. The dealer passes results of step 1 and 2 to the first shareholder. Shareholder should multiply her share by the $1^{st}$ constant and add multiple to the share she received. The sum is passed to the next shareholder.
5. Repeat step 4 for other \((t - 1)\) shareholders: the \(i^{th}\) shareholder multiplies his share by the \(i^{th}\) constant and adds multiple to the share he received. The \(i^{th}\) shareholder passes his sum to the dealer.

6. The dealer knows all constants and initial random share passed to the first shareholder. The dealer subtracts the private integer part of the initial random share from the result she obtained from the \(i^{th}\) shareholder – this is the private part of the new share. To get the public part (a vector of length \(t\)) the dealer computes the linear combination of public vector parts of shares that participated in the enrollment process.

This concludes the generation process. The private part of the new share is delivered to the new shareholder, and the public part is added to the public data file. The updated public data file is distributed to all shareholders.

### 3.3.1 Discussion

The enrollment feature does not require any modifications to the original scheme. In the same time, it provides a way to add new shareholders without reconstructing the secret and recreating scheme. It takes \(O(t)\) time to construct the private part of the new share, and \(O(1)\) to construct the public vector.

The proposed algorithm does not prevent cheating. Two cooperating shareholders can easily obtain private share of another participant. Denote cheats by C1 and C2. Denote attacked shareholder by S. To get S’s share C1 has to send his result to S, and C2 has to receive result from S. Then C1 and C2 communicate and subtract the result received by C2 from the result sent by C1. Then they divide the difference by the constant term used by S. The result of this division is the private part of S’s share.

This algorithm does not give the dealer a chance to reconstruct the secret, because the dealer only gets the final product of the linear combination. Although, the dealer can still cheat by corrupting the new share.

### 3.4 Recovery of lost shares

Shares can be lost. And no kind of protection can guarantee their safety. A loss of a share does not imply that its shareholder loses her part of the secret, but it prevents this shareholder to participate in any kind of secret-related activity. In non-proactive schemes the only way to deal with loss is to reconstruct the secret and create a new scheme, by distributing new shares to shareholders. But it is usually unsafe to reconstruct a secret.

The proposed scheme simply generates a new share to replace the lost one. Thus, the recovery algorithm is basically the same as the enrollment algorithm described in section 3.3. The only difference is that the generated share is sent to an existing shareholder.
3. Proposed secret sharing scheme

3.5 Disenrollment of shares

Changes of the access structure do not consist solely of new shareholders’ enrollment and lost shares’ recovery. Sometimes it is required to disenroll a participant, either because he is cheating or because he does not wish to participate any more. There are many other reasons as well. The simplest, but not the best, solution is to reconstruct the secret and split it into fewer shares.

In the proposed secret sharing scheme a participant is disenrolled by removing the public part of her share from the public data file. This prevents the removed shareholder from participating in a reconstruction of the secret. Of course, if \((t - 1)\) shareholders decide to cooperate with the removed one, they will be able to reconstruct the secret. Although, if public data file is obtained from a trustworthy place and all participating shareholders agree on the content of this file, then removed shares are not able to participate.

3.6 General access structure

The implemented version of the proposed secret sharing scheme supports only threshold access structures. In this section we are going to describe how to realize general access structure.

Ito, Saito, and Nishizeki showed that any monotone access structure can be realized as a perfect secret sharing scheme [6]. We are going to illustrate their method here.

For every access structure \(\Gamma\), there is a unique collection of maximal unauthorized subsets. The collection of all maximal unauthorized subsets is defined as

\[
M = \{B_1, \ldots, B_m\}
\]

such that each subset \(B_i \not\in \Gamma\), but

\[
B_i \cup P_j \in \Gamma
\]

for any \(P_j \not\in B_i\), where \(P_j\) is the \(j^{th}\) shareholder.

As an example consider the following access structure

\[
\Gamma = cl(\{P_1, P_2\}; \{P_3, P_4\})
\]

where the set of shareholders \(P = \{P_1, P_2, P_3, P_4\}\). The collection of maximal unauthorized subsets \(M = \{B_1, B_2, B_3, B_4\} = \{\{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\}\}\). For example, if any extra shareholder \(P_2\) or \(P_4\) is added to \(\{P_1, P_3\}\), then \(\{P_1, P_3\}\) becomes an authorized subset.

A Boolean function \(\Gamma(P_1, \ldots, P_n)\) can be associated with every access structure \(\Gamma\)

\[
\Gamma(P_1 = p_1, \ldots, P_n = p_n) = \begin{cases} 
1 & \text{if } \{P_i \mid p_i = 1; i = 1, \ldots, n\} \in \Gamma \\
0 & \text{otherwise}
\end{cases}
\]

In our example \(\Gamma = cl(\{P_1, P_2\}; \{P_3, P_4\})\) and the corresponding Boolean function is \(\Gamma(P_1, P_2, P_3, P_4) = P_1P_2 + P_3P_4\).
3. Proposed secret sharing scheme

A dual Boolean function \( \Gamma^*(P_1, \ldots, P_n) \) is generated from Boolean function \( \Gamma(P_1, \ldots, P_n) \)

\[
\Gamma^*(P_1, \ldots, P_n) = \neg \Gamma(\neg P_1, \ldots, \neg P_n)
\]

The dual Boolean function is associated with the corresponding dual access structure \( \Gamma^* \).

In our example the dual function generated from the function \( \Gamma(P_1, P_2, P_3, P_4) = P_1P_2 + P_3P_4 \) is

\[
\Gamma^*(P_1, P_2, P_3, P_4) = (P_1 + P_2) (P_3 + P_4) = P_1P_3 + P_1P_4 + P_2P_3 + P_2P_4.
\]

And the dual access structure is \( \Gamma^* = cl(\{ \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\} \}) \).

**Definition 3.1** (Simmons, [12]). Given \( \Gamma = cl(\Gamma_0) \) for \( n \) shareholders with the corresponding collection \( M = \{B_1, \ldots, B_m\} \) of maximal unauthorized subsets. Let \( T \) be a set of integers from which share tokens are chosen. Then the cumulative array \( C_\Gamma \) is the assignment of \( m \) share tokens \( s_i \in T \) to each shareholder \( P_i \in P \) according to the following table:

<table>
<thead>
<tr>
<th>( M )</th>
<th>( B_1 )</th>
<th>( B_2 )</th>
<th>\ldots</th>
<th>( B_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Gamma^* )</td>
<td>( A_1 )</td>
<td>( A_2 )</td>
<td>\ldots</td>
<td>( A_m )</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>( s_2 )</td>
<td>\ldots</td>
<td>( s_m )</td>
<td></td>
</tr>
<tr>
<td>( P_1 )</td>
<td>( b_{1,1} )</td>
<td>( b_{1,2} )</td>
<td>\ldots</td>
<td>( b_{1,m} )</td>
</tr>
<tr>
<td>( P_2 )</td>
<td>( b_{2,1} )</td>
<td>( b_{2,2} )</td>
<td>\ldots</td>
<td>( b_{2,m} )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ddots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( P_n )</td>
<td>( b_{n,1} )</td>
<td>( b_{n,2} )</td>
<td>\ldots</td>
<td>( b_{n,m} )</td>
</tr>
</tbody>
</table>

The entries \( b_{i,j} \) are binary and

\[
b_{i,j} = \begin{cases} 1 & \text{if } P_i \in A_j \\ 0 & \text{otherwise} \end{cases}
\]

The participant \( P_i \) gets a share \( S_i = \{ s_j \mid b_{i,j} = 1 \} \) for \( i = 1, \ldots, n \).

In our example \( \Gamma(P_1, P_2, P_3, P_4) = cl(P_1P_2 + P_3P_4) \) and its dual access structure is \( \Gamma^* = cl(\{ \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_3\}, \{P_2, P_4\} \}) \). A Boolean function associated with \( \Gamma^* \) is \( \Gamma^*(P_1, P_2, P_3, P_4) = P_1P_3 + P_1P_4 + P_2P_3 + P_2P_4 \). The cumulative array \( C_\Gamma \) has the following content:
3. Proposed secret sharing scheme

<table>
<thead>
<tr>
<th>$\Gamma^*$</th>
<th>$P_1P_3$</th>
<th>$P_1P_4$</th>
<th>$P_2P_3$</th>
<th>$P_2P_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$s_1$</td>
<td>$s_2$</td>
<td>$s_3$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$P_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$P_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Hence, the shareholder $P_1$ gets shares $s_1$ and $s_2$, $P_2$ – shares $s_3$ and $s_4$, $P_3$ – shares $s_1$ and $s_3$, and $P_4$ – shares $s_2$ and $s_4$.

The cumulative array algorithm does not guarantee that the amount of shares kept by a shareholder is the smallest possible. For example, consider a (2,3) threshold scheme. $\Gamma = cl(\{\{P_1, P_2\}, \{P_1, P_3\}, \{P_2, P_3\}, \{P_1, P_2, P_3\}\})$. The Boolean function is

$$\Gamma(P_1, P_2, P_3) = P_1P_2 + P_1P_3 + P_2P_3 + P_1P_2P_3 = P_1P_2 + P_1P_3 + P_2P_3.$$ 

And the dual Boolean function is

$$\Gamma^*(P_1, P_2, P_3) = (P_1 + P_2)(P_1 + P_3)(P_2 + P_3) = P_1P_2 + P_1P_3 + P_2P_3.$$ 

The cumulative array $C_\Gamma$ is:

<table>
<thead>
<tr>
<th>$\Gamma^*$</th>
<th>$P_1P_2$</th>
<th>$P_1P_3$</th>
<th>$P_2P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$s_1$</td>
<td>$s_2$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$P_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Hence, any two shareholders can reconstruct the secret and each shareholder gets 2 shares. Obviously, this is not the least possible amount of shares, because Shamir’s scheme requires only one share per shareholder in the same scenario.
3.7 Comparison of the proposed scheme with Shamir’s and Blackley’s

In this section we are going to compare the underlying structure of the three schemes.

3.7.1 Proposed scheme

In the \((t,n)\)-threshold scheme proposed in this chapter a secret is an integer number \(S\). For better security the actual secret is not shared right away. First, a dot product equal to the secret number is generated,

\[
\overline{S} \cdot \overline{X} = a_1x_1 + a_2x_2 + \ldots + a_nx_n = S
\]

Then one of the dot product vectors, \(\overline{S}\), is shared and the second vector, \(\overline{X}\), becomes public and is known to every shareholder.

The \(i^{th}\) share is a dot product of the secret vector \(\overline{S}\) with a public vector \(\overline{Y}_i = (y_{i1}, y_{i2}, \ldots, y_{in})\): \(share_i = \overline{S} \cdot \overline{Y}_i\). Every \(t\) of the \(n\) vectors \(\overline{Y}_i\) must be linear independent, so that any \(t\) shares are enough to reconstruct the secret vector \(\overline{S}\).

\[
\Delta = \begin{vmatrix}
y_{11} & y_{12} & \cdots & y_{1t} \\
y_{21} & y_{22} & \cdots & y_{2t} \\
\vdots & \vdots & \ddots & \vdots \\
y_{t1} & y_{t2} & \cdots & y_{nt}
\end{vmatrix}
\]

The determinant \(\Delta \neq 0\) because every \(t\) vectors \(\overline{Y}_i\) are linear independent. Thus the system (3.4) has a unique solution for \((a_1,\ldots,a_n)\)

\[
\begin{align*}
share_1 &= a_1y_{11} + a_2y_{12} + \ldots + a_ny_{1t} \\
share_2 &= a_1y_{21} + a_2y_{22} + \ldots + a_ny_{2t} \\
&\vdots
\end{align*}
\]

To reconstruct the actual secret \(S\), which is a scalar, the dot product \(\overline{S} \cdot \overline{X} = S\) is computed.

3.7.2 Shamir’s scheme

In Shamir’s \((t,n)\)-threshold scheme a secret is an integer number \(S\), which is equal to first coefficient of the polynomial \(f(x) = \sum_{j=0}^{t-1} a_jx^j\) (refer to the section 2.2 for a complete description
of the Shamir’s scheme). The vector of coefficients \( \overline{S} = (a_1, a_2, \ldots, a_r) \) is shared among \( n \) participants. The \( i^{th} \) share \( \text{share}_i = f(x_i) \), where \( x_i, 1 \leq i \leq n \), are \( n \) distinct public points.

To reconstruct the secret \( S = a_1 \), the following system of equations has to be solved:

\[
\begin{align*}
\text{share}_1 &= a_1 + a_2 x_1 + \ldots + a_r x_1^{r-1} \\
\text{share}_2 &= a_1 + a_2 x_2 + \ldots + a_r x_2^{r-1} \\
&\vdots \quad (3.5) \\
\text{share}_r &= a_1 + a_2 x_r + \ldots + a_r x_r^{r-1}
\end{align*}
\]

Since the Vandermonde determinant \( \Delta \neq 0 \), the system (3.5) has a unique solution for \((a_1, \ldots, a_r)\)

\[
\Delta = \begin{vmatrix}
1 & x_1 & \cdots & x_1^{r-1} \\
1 & x_2 & \cdots & x_2^{r-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_r & \cdots & x_r^{r-1}
\end{vmatrix}
\]

Observe that the Shamir’s scheme is a special case of the proposed scheme where the shares are the power series of \( x_i \)'s and the secret is just the first element of the secret vector \( \overline{S} \).

### 3.7.3 Blakley’s scheme

In the Blakley’s scheme the secret is a point in a \( t \)-dimensional space and \( n \) shares are constructed, with each share defining an affine hyperplane in this space. In other words the \( i^{th} \) share is a vector \( \overline{Y}_i = (y_{i1}, \ldots, y_{ir}) \), which is a solution to an equation \( a_1 y_{i1} + \ldots + a_r y_{ir} = b_i \), where \( b_i \)'s are publicly known constants. Every \( t \) of the \( n \) vectors \( \overline{Y}_i \) must be linear independent. The secret is a point with coordinates \((a_1, a_2, \ldots, a_r)\). In other words the secret is a vector \( \overline{S} = (a_1, a_2, \ldots, a_r) \). By finding the intersection of any \( t \) of the planes-shares, the secret (point of the intersection) is obtained. The system of equation that has to be solved is

\[
\begin{align*}
b_1 &= a_1 y_{i1} + a_2 y_{i2} + \ldots + a_r y_{ir} \\
b_2 &= a_1 y_{i2} + a_2 y_{i2} + \ldots + a_r y_{2i} \\
&\vdots \quad (3.6) \\
b_t &= a_1 y_{it} + a_2 y_{i2} + \ldots + a_r y_{rit}
\end{align*}
\]
3. Proposed secret sharing scheme

\[
\Delta = \begin{vmatrix}
  y_{11} & y_{12} & \cdots & y_{1r} \\
  y_{21} & y_{22} & \cdots & y_{2r} \\
  \vdots & \vdots & \ddots & \vdots \\
  y_{t1} & y_{t2} & \cdots & y_{tr}
\end{vmatrix}
\]

The determinant \( \Delta \neq 0 \) because every \( t \) vectors \( \overrightarrow{y_i} \) are linear independent. Thus the system (3.6) has a unique solution for \((a_1, \ldots, a_t)\).

The Blackley’s scheme is quite similar to the proposed and Shamir’s schemes, but it has some differences. Its secret and shares are vectors of length \( t \), not scalars. Basically, shares in Blackley’s correspond to public vectors in the proposed and Shamir’s schemes. And the public constants in Blackley’s correspond to shares in the proposed and Shamir’s schemes. The secret in Blackley’s is the same as the intermediate secret vector in the proposed scheme. These observations lead to a simple way to make Blackley’s perfect (recall that we showed its imperfectness in section 2.3). To make it perfect the secret vector \( \overrightarrow{S} \) has to be treated as an intermediate secret vector. And the actual secret should be a dot product of the secret vector \( \overrightarrow{S} \) with some public vector \( \overrightarrow{X} \). Although, now the scheme is not ideal any more. To fix that we need to switch roles of constants \( b_i \) and vectors \( \overrightarrow{Y_i} \). The former should be shares and the latter – public vectors. These changes retain all existing properties of the Blackley’s scheme and add perfectness to it. The interesting fact is that the underlying structure of the new Blackley’s scheme is exactly the same as that of the proposed scheme.

3.7.4 Conclusion

All three schemes share a similar structure and can be represented in a very similar way using determinants and dot product notation. The proposed secret sharing scheme is the most secure of the three reviewed in this section, because its secret is a dot product of an intermediate secret vector shared among participants.

It is fairly easy to make the Blackley’s scheme perfect. Although, the resulting scheme is absolutely equivalent to the proposed one.
Appendix: User Manual

Secret Sharing Scheme consists of two applications – SSSDealer and SSSClient. The former one is used by a dealer to construct and reconstruct secrets, renew, enroll and disenroll shares. The later one is used by shareholders to renew shares and to participate in enrollment processes. Section 1 of this manual describes how to use SSSDealer, section 2 covers SSSClient.

Java Runtime Environment has to be installed in order to run the applications. Java 2 Runtime Environment, build 1.4 or better is recommended.

1. SSSDealer

To run SSSDealer change the current directory to the one containing SSSDealer.jar and execute the following command:

```shell
> java –jar SSSDealer.jar
```

Optionally, path to the directory containing SSSDealer.jar can be added to CLASSPATH variable. This will allow running the application from any directory.

Use “Exit” button in the right bottom corner of the window to terminate application.

1.1 Secret construction

![Secret Sharing Scheme - Dealer](image)

Secret must be an integer of arbitrary precision. A special utility is provided to convert a text string of arbitrary length into an integer number ready to be shared. A reverse utility allows converting an integer into a string. Learn more about these utilities in section 1.6 of this manual.

The initial parameters of the secret sharing scheme are defined by 3 options: threshold, number of shares, and precision. We will discuss each of them in order.

**Threshold** determines how many shares are required to be put together in order to reconstruct the secret (reconstruction is discussed in section 1.2 of this manual). Threshold must be
- greater than or equal to 1,
- less than or equal to 255,
- greater than or equal to the total number of shares.

SSSDealer automatically adjusts the threshold to satisfy these requirements. Note that once the scheme is constructed the threshold value cannot be changed.
**Shares** parameter defines the total number of shares initially generated. Additional shares can be generated anytime later (enrollment is discussed in section 1.4 of this manual). At any time the total number of shares must be
- greater than or equal to 1,
- less than or equal to 255,
- greater than or equal to the threshold value.

SSSDealer automatically adjusts the “Shares” field to satisfy these requirements.

**Precision** parameter defines the maximal size in bits of each share. This size cannot be less than the size of the secret. You cannot manually set precision to be greater than 512 bits, but SSSDealer will make sure that precision is not less than the size of the secret. In other words there is no internal limit for precision. It is advised to use precision parameter to better protect small secrets of size less than 512 bits.

SSSDealer has a simple encryption engine which provides encryption for shares. Check “Encrypt shares” box to enable it. Encryption parameters are defined by 2 options: algorithm and strength. We will discuss each of them in order.

**Algorithm** defines the algorithm used to encrypt shares and can be one of TripleDES (DESede), Blowfish, and DES.

**Strength** defines the size of the encryption key in bits. Each algorithm has its own possible key sizes. The following table describes possible key sizes for each algorithm:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key sizes (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TripleDES</td>
<td>112, 168</td>
</tr>
<tr>
<td>Blowfish</td>
<td>32, 64, 128, 256, 448</td>
</tr>
<tr>
<td>DES</td>
<td>56</td>
</tr>
</tbody>
</table>

The greater the size of the key, the better is the security of encrypted shares. A key size of at least 128 bits is recommended.

By default all generated files (shares and public data file) are in binary format. If you wish to have them in human-readable plain text format as well then check the “**Generate text files**” box. Note that plain text files are not encrypted. The encryption key is only outputted in binary format.

“**Construct**” button gathers entered parameters and generates shares (.shr files) and public data file (.pd file). If encryption is enabled, then a .key file is produced. Text files have .txt extension; their names correspond to the names of binary files.
1.2 Secret reconstruction

To reconstruct a secret in a $(t,n)$-threshold scheme (i.e. threshold is $t$, total number of shares is $n$) at least $t$ shares must be present in the same directory. All shares must have `.shr` extension. If some shares are encrypted then the decryption key must be in the same folder in a file with `.key` extension. The path to this directory should be entered in the “Path to directory with private shares” field. Use the top “Browse...” button to enter the path easily and without errors.

The file with public data is allowed to be in another directory. Path to this directory should be entered in the “Path to public data file” field. Public data file must have `.pd` extension.

“Reconstruct” button gathers entered parameters and reconstructs the secret. In case of successful reconstruction secret integer number is displayed in the “Secret” field under the ”Reconstruct” button. “Secret” field is copy-enabled.
Appendix: User Manual

1.3 Renewal of shares

Often the nature of the secret does not allow changing it (for example secret keys, documents, or data files). Renewal allows updating shares in the way such that the renewed shares combine to the same secret as original ones.

The only information required to generate an update is the public data file. Path to it should be entered in the “Path to public data file” field. The output directory for the generated update file is specified in the “Output Folder” field. Checked “Generate Text File” box triggers the text version of the update to be created and placed in the “Output Folder”.

“Generate Update” button generates update.ren file in the “Output Folder”. This file has to be distributed to every shareholder. Consult SSSClient manual on how to renew a share using SSSClient application (section 2.2).

If the public data file cannot be found or is read-protected then an error message window will pop up and the update file will not be created.
1.4 Enrollment of new shareholders

If at least threshold number of shareholders agree to create a new share, either because a new participant is admitted into the scheme, or because some share was lost, then the “Enroll / Recover” tab should be used.

Enrollment process is quite complicated and lengthy. It consists of two steps and SSSDealer application must not be closed until all steps are completed. If SSSDealer is closed before enrollment is completed all intermediate data is lost and the process has to be restarted.

In the first step the enroll.enr file is generated. To create it you have to specify the path to the public data file in the “Path to public data file” field and the path to the directory where generated file will be placed in the “Output Folder for .enr file” field. If you wish to have text version of the .enr file, check “Generate Text File” box. Once all fields are filled press “Initiate Enroll” button to generate an .enr file. If public data file cannot be found or is read-protected then an error message window will pop up and the update file will not be created.

The generated .enr file should be sent to the first of t shareholders willing to participate in the enrollment process (recall that t = threshold). Each of these t shareholders should modify .enr file using SSSClient application and send the modified .enr file to the next shareholder. The t\textsuperscript{th} shareholder returns its modified .enr file to the dealer. To learn more about how to use SSSClient application to participate in the enrollment consult section 2.3 of this manual.

Once you get .enr file back from the last participating shareholder, the second step begins. Indicate the path to the returned .enr file in the “Path to enroll data file” text field. Specify the directory where the new share should be stored in the “Output Folder for new share” text field. Checked “Generate Text File” box triggers the text version of the new share to be created and placed in the same directory as the new share. Note that there is no option to encrypt the new share.

In case of a recovery process you might wish to give the new share the index of the lost share. To do this check “Replace share number” box and choose the index number in the corresponding field. SSSDealer will make sure that the entered number is within the current indices. Do not use this option if you are not sure what index the lost share had. You must not use this option if you are performing an enrollment.
Once all fields are filled, press the “Generate and Enroll share” button to generate the new share in the specified directory. The new .shr file should be sent to the new shareholder. The public data file has been modified and should be distributed to all shareholders. Note that you can use an .enr file only once to generate a new share. If it is used more than once, the scheme will be broken.

1.5 Disenrollment of shareholders

Use the “DisEnroll” tab to disenroll a share with a particular index.

First, enter path to the public data file in the “Path to public data file” text field. Then press “Load” button to load the public data file. If the public data file cannot be found or is read-protected then error message window will pop up. Now choose the index of the share you wish to disenroll. Indices of shares can be found either in the text versions of shares and public data file, or by loading shares into SSSClient application. SSSDealer will make sure that the entered index is within the current indices bounds.

Check the “Generate Text File” if you wish to create a text version of the new public data file.

Press “Disenroll Share” button to remove the share with the specified index from the public data file. The new version of the public data file replaces the old one and should be distributed to every shareholder.

Note that you cannot disenroll a share more than once. Also note that once share is disenrolled you may assign its index to a new share (see section 1.4 to learn how to enroll new shares).
1.6 Utilities for TEXT⇔NUMBER conversions

Two easy-to-use utilities are provided to convert a text string of arbitrary length into an integer number ready to be shared and vice versa.

To convert a text string into an integer, enter the string in the “Text String” field and press “Text to Number” button.

To convert an integer number into a text string, enter the number in the “Integer number” field and press “Number to Text” button. Be sure to enter a valid decimal integer.

(Figure 1.11)
2. SSSClient

To run SSSClient change the current directory to the one containing SSSClient.jar and execute the following command:

> java -jar SSSClient.jar

Optionally, path to the directory containing SSSClient.jar can be added to CLASSPATH variable. This will allow running the application from any directory.

Use the “Exit” button in the right bottom corner of the application window to terminate the program (see Figure 2.1).

2.1 Loading shares

To load a share into the SSSClient application indicate path to the share in the “Path to share file” text field and press the “Load” button. The “Browse” button brings up a browse window with a directory tree where you can easily identify the target directory. The share must have .shr extension. If the share is encrypted, the decryption key must be in the same directory as the share. The key must have .key extension.

If share is successfully loaded, its information will be displayed underneath the “Share Information”. The information consists of a serial number and a decimal integer number. You can copy integer number from the “Share” field.

Once share is loaded, the other tabs become activated.

(Figure 2.1)

2.2 Renewal of shares

When you receive an update file for your share(s) from the dealer you have to apply this update to maintain validity of your share(s). Perform the following steps for each of your shares.

Load a share into the SSSClient application as described in section 2.1 of this manual. Go to the “Renew” tab (see Figure 2.2). Specify the path to the public data file in the “Path to public data file” field. The public data file must have .pd extension. Indicate path to the update file in the “Path to update file” field. The update file must have .ren extension.
The “Browse…” buttons may help you to enter paths faster and without errors. Check “Generate Text File” box if you want to have plain text version of the updated share. Note that old version of the share is going to be overwritten. Remove write protection from your share file.

Press “Renew” button to apply the update. If all files are successfully found and are not read-protected the information about the updated share will be displayed underneath the “Updated share information”.

2.3 Participating in enrollment

When you choose to participate in an enrollment and receive .enr file from your dealer, you need to do the following actions.

Load your share into the SSSClient application as described in section 2.1 of this manual. Go to the “Participate in Enrolment” tab (see Figure 2.3). Indicate the path to the enroll data file sent to you by the dealer in the “Path to enroll data file” field. Note that the enroll data file must have .enr extension. Once you press “Participate in Enrolment” button, the .enr file is updated with the loaded share information. Send .enr file to the next participant or to the dealer.

3. About

SSSDealer and SSSClient applications were written by Ivan Minevskiy as a part of an Undergraduate Honours Project under the supervision of Dr. Joel Friedman. University of British Columbia, 2003-2004.
Bibliography


