Rearrangements of access structures and their realizations in secret sharing schemes

Zhifang Zhang\textsuperscript{a,1}, Mulan Liu\textsuperscript{a,*2}, Liangliang Xiao\textsuperscript{b}

\textsuperscript{a}Academy of Mathematics and Systems Science, Key Laboratory of Mathematics Mechanization, Chinese Academy of Sciences, Beijing 100080, China
\textsuperscript{b}Institute of Software, Chinese Academy of Sciences, Beijing 100080, China

Received 24 April 2005; accepted 5 September 2007
Available online 23 October 2007

Abstract

Firstly, the definitions of the secret sharing schemes (SSS), i.e. perfect SSS, statistical SSS and computational SSS are given in an uniform way, then some new schemes for several familiar rearrangements of access structures with respect to the above three types of SSS are constructed from the old schemes. It proves that the new schemes and the old schemes are of the same security. A method of constructing the SSS which realizes the general access structure by rearranging some basic access structures is developed. The results of this paper can be used to key managements and access controls.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Perfect secret sharing scheme; Statistical secret sharing scheme; Computational secret sharing scheme; Rearrangement of access structures

1. Introduction

Secret sharing schemes (SSSs) which were independently proposed by Shamir \cite{10} and Blakley \cite{4} in 1979 play an important role in cryptography and distributed computation. Informally, an SSS is a protocol between a dealer and a set of participants who share a secret such that only participants in an authorized set can recover the secret. So far, only the perfect case of SSS has been extensively investigated \cite{11}, requiring that the participants in any unauthorized set get absolutely no information about the secret. Recently, two other cases of SSS with statistical security and computational security were studied by Beimel and Ishai \cite{2} in 2001 and Krawczyk \cite{9} in 1993, respectively. Although fruitful results for the perfect SSS have been obtained, it is still necessary to study the statistical SSS and the computational SSS. Since all perfect SSS are trivially statistical ones, the statistical SSS are expected more powerful than the perfect SSS. The computational SSS are closer to real life because the computational security requirement is good enough to resist an adversary in real life. Beimel and Ishai \cite{2} gave a statistical SSS with respect to an access structure for which no perfect SSS is known to efficiently realize it. Some computational SSS which can efficiently realize more general access

\* Corresponding author.
E-mail address: mliu@amss.ac.cn (M. Liu).

1 Supported by the Knowledge Innovation Program of the Chinese Academy of Sciences.

2 Supported by the National Natural Science Foundation of China (No. 90304012) and 973 project (No. 2004CB318000).

0012-365X/S - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.disc.2007.09.007
structures were considered in [1,12]. In this paper, we will give definitions for the three types of SSS in an uniform way which is more helpful to understand the difference between them.

In SSSs, the access structures sometimes change. This happens when new participants join in, some participants leave out, and so on. Hence, how to build a new scheme from old ones to realize the changed access structure is always an important topic in SSS. Benaloh and Leichter [3] realized the union and intersection of two access structures in the perfect case, and papers [5,7,8] discussed some other changes. In this paper we mainly discuss several familiar rearrangements which contain most of the changes discussed so far. This problem was considered by Xiao and Liu [13] in the perfect case. We further extend their results to the statistical case and the computational case which are much closer to real implementation.

Finally, we construct the SSS for the general access structure by rearranging some basic access structures. The construction is similar to Benaloh and Leichter’s [3] which was proposed from a traditional point in view of that the access structures are equivalent to the underlying monotone formulae. Our construction is more direct and more efficient in some cases.

2. Preliminaries

Let \( \mathbb{N} \) be the set of non-negative integers and \( \mathbb{R} \) the set of real numbers. We call a function \( \mu : \mathbb{N} \to \mathbb{R} \) negligible if for every positive polynomial \( p(\cdot) \), there exists a positive integer \( N \) such that \( \mu(n) < 1/p(n) \) for all \( n \in \mathbb{N} \) and \( n > N \).

Suppose that \( P = \{ P_1, \ldots, P_n \} \) is a set of participants and AS is a non-empty subset of \( 2^P \). Then the closure of AS, denoted by \( \text{cl}(AS) \), is the set \( \text{cl}(AS) = \{ C \subseteq P \mid \exists B \in AS, \text{ such that } B 
subseteq C \} \). We call AS an access structure over \( P \) if it satisfies the monotone ascending property: for any \( A \in \text{AS} \) and any \( A' \in 2^P \), \( A \subseteq A' \) implies \( A' \in \text{AS} \). Obviously, if AS is an access structure, then \( \text{AS} = \text{cl}(\text{AS}) \) holds. The elements in AS are usually called the authorized sets, and the elements in \( 2^P \setminus \text{AS} \) are called the unauthorized sets. Furthermore, \( B \in \text{AS} \) is a minimum authorized subset of AS if \( A \notin \text{AS} \) whenever \( A \nsubseteq B \). Let \( \text{AS}_m \) be the collection of all minimum authorized subsets of AS. It is obvious that \( \text{AS} = \text{cl}(\text{AS}_m) \). Actually, \( \text{AS} \) and \( \text{AS}_m \) can be uniquely determined by each other.

An SSS is a protocol between a dealer and a group of participants, where the dealer can be viewed as an authority who distributes shares of a secret to participants. More precisely, there is a distribution function \( H(\cdot) \), which is the secret-domain, \( \mathbb{S}_i \) the share-domain of \( P_i \), \( 1 \leq i \leq n \), and \( R \) is the set of random inputs. Without loss of generality, we may assume that the sets \( S, R, S_i \) are finite. A secret \( s \in S \) is shared as follows: the dealer randomly selects \( r \in R \) and computes \( \Pi(s, r) = (s_1, \ldots, s_n) \). Then he secretly sends the share \( s_i \) to \( P_i \). For the sake of simplicity, we denote by capital letters the random variable which ranges over the corresponding set. For instance, \( S \) denotes the random variable ranging over the secret-domain \( S \) according to some specified distribution. For any \( A = \{ P_1, \ldots, P_{\mid A \mid} \} \subseteq P \) and \( s \in S, \Pi(s, R)|_A \) is a random variable ranging over \( (S_1 \times \cdots \times S_{\mid A \mid}) \) induced by the random variable \( R \).

An SSS is said to (perfectly) realize an access structure AS if the following conditions hold:

\[ (1') \ \text{Correctness: Participants of any authorized set can recover the secret by putting their shares together. Formally, for all } A \in \text{AS}, H(S \mid \Pi(S, R)|_A) = 0 \text{ holds, where } H(\cdot) \text{ is the entropy function.} \]

\[ (2') \ \text{Security: Participants of any unauthorized set get absolutely no information on the secret. Formally, for all } B \notin \text{AS}, H(S \mid \Pi(S, R)|_B) = H(S) \text{ holds.} \]

The perfect SSS was defined in [10,6]. We now give an equivalent definition by comparing the probability distribution of two random variables. The equivalence can be easily proved and we omit the details.

**Definition 1.** An SSS is called perfect if the following conditions hold:

\[ (1'') \ \text{Perfect correctness: For any } A \in \text{AS}, \text{ there exists a reconstruction function } \text{Re}_A : (S_1 \times \cdots \times S_{\mid A \mid}) \to S, \text{ such that for any } s \in S, \Pr[\text{Re}_A(\Pi(s, R)|_A) = s] = 1. \text{ Since the subscript for the reconstruction function is explicit from the context, we often omit it and denote an SSS by } \text{SSS}(\Pi, \text{Re}). \]

\[ (2'') \ \text{Perfect security: For any } B \notin \text{AS} \text{ and any } s, s' \in S, \Pi(s, R)|_B \text{ and } \Pi(s', R)|_B \text{ are identically distributed, i.e. for any } z \in (S_1 \times \cdots \times S_{\mid B \mid}), \Pr[\Pi(s, R)|_B = z] = \Pr[\Pi(s', R)|_B = z]. \]
Although perfect SSSs have been widely studied, in some practical cases, the security requirements can be relaxed so that the distributions of the random variables \( II(s, R)_B \) and \( II(s', R)_B \) are somewhat different. In fact, Krawczyk [9] slightly relaxed the security requirement and studied the SSS with computational security, but he only considered the extension of the security requirement. We now further extend the correctness requirement to the computational case and get the complete definition for computational SSS. The concept of statistical SSS was proposed by Beimel and Ishai [2] in which they focused on non-linear SSS. Inspired by their points, we define three types of SSS in an uniform way so that the security requirement can be determined by the difference between \( II(s, R)_B \) and \( II(s', R)_B \). By our uniform definition, the difference among these three types of SSS is clear.

**Definition 2.** An SSS is called statistical if the following conditions hold:

1. Statistical correctness: For any \( A \in AS \), there exists a reconstruction function \( Re : (S_1 \times \cdots \times S_n)_A \rightarrow S \) such that for any \( s \in S \), \( Pr[Re(II(s, R)_A) = s] > 1 - \mu(n) \) for some negligible function \( \mu \).
2. Statistical security: For any \( B \not\in AS \) and any \( s, s' \in S \), the statistical difference between \( II(s, R)_B \) and \( II(s', R)_B \) is negligible, i.e., \( \frac{1}{2} \sum_x |Pr[II(s, R)_B = x] - Pr[II(s', R)_B = x]| < \mu(n) \), where \( \mu \) is a negligible function and \( x \) runs over \( (S_1 \times \cdots \times S_n)_B \).

**Definition 3.** An SSS is called computational, if the following conditions hold:

1. Computational correctness: For any \( A \in AS \), there exists a probabilistic polynomial-time computable function \( Re : (S_1 \times \cdots \times S_n)_A \rightarrow S \) such that for any \( s \in S \), \( Pr[Re(II(s, R)_A) = s] > 1 - \mu(n) \), for some negligible function \( \mu \).
2. Computational security: For any \( B \not\in AS \) and any \( s, s' \in S \), no probabilistic polynomial-time algorithm can distinguish between \( II(s, R)_B \) and \( II(s', R)_B \), i.e., for any probabilistic polynomial-time algorithm \( D \), \( |Pr[D(II(s, R)_B, 1^n) = 1] - Pr[D(II(s', R)_B, 1^n) = 1]| < \mu(n) \), for some negligible function \( \mu \), where \( 1^n \) is the unary expression of \( n \) which is a auxiliary input to \( D \) as the security parameter.

3. Union and intersection of two access structures

In this section, we consider the realization of the union and intersection of two access structures. We construct the SSS which realizes the rearranged access structures based on the original schemes and prove that the constructed scheme is still perfect (resp. statistical, computational) if the original schemes are perfect (resp. statistical, computational). Before giving formal descriptions, we use an example to explain the real life background for rearrangements of access structures.

Suppose Alice and Bob each has an bank account with a corresponding password. For some reasons they want to manage their accounts together, so they have to unite the two accounts and set a new password for the united account. The former can enter the united account alone; If they do not trust each other, then each of them holds a share which cannot deduce other’s manage their accounts together, so they have to unite the two accounts and set a new password for the united account.

Before giving formal descriptions, we use an example to explain the real life background for rearrangements of access structures.

Let \( AS_1, AS_2 \) be two access structures over the set of participants \( P = \{ P_1, \ldots, P_n \} \), and \( SSS_1(II_1, Re_1) \), \( SSS_2(II_2, Re_2) \) the SSSs which realize \( AS_1 \) and \( AS_2 \), respectively. For simplicity, we assume that \( SSS_1 \) and \( SSS_2 \) have the same secret-domain and \( II_i : S \times R_i \rightarrow S_1^{(i)} \times \cdots \times S_n^{(i)} \) for \( i = 1, 2 \). First, we construct an secret sharing scheme \( SSS(II, Re) \) to realize \( AS = AS_1 \cup AS_2 \) as follows.

**Construction 1.** Given a secret \( s \in S \), by using the distribution functions \( II_1 \) and \( II_2 \), the dealer computes \( II_1(s, r_1) = (s_{11}, \ldots, s_{1n}) \) and \( II_2(s, r_2) = (s_{21}, \ldots, s_{2n}) \), where \( r_1 \) and \( r_2 \) are independently and uniformly selected in \( R_1 \) and \( R_2 \), respectively. Then the dealer secretly sends the share \( (s_{1i}, s_{2i}) \) to \( Pi \). For the sake of convenience, we denote \( II : S \times R_1 \times R_2 \rightarrow S_1^{(1)} \times \cdots \times S_n^{(1)} \times S_1^{(2)} \times \cdots \times S_n^{(2)} \),

\[
(s, r_1, r_2) \mapsto (II_1(s, r_1)|_{\{P_1\}}, \ldots, II_1(s, r_1)|_{\{P_n\}}, II_2(s, r_2)|_{\{P_1\}}, \ldots, II_2(s, r_2)|_{\{P_n\}}),
\]

where \( Pi \)'s share-domain is \( S_i^{(1)} \times S_i^{(2)} \).
Proposition 1 (Union of access structures). Suppose that SSS_1 and SSS_2 are perfect (resp. statistical, computational) SSSs realizing AS_1 and AS_2, respectively. Then the scheme SSS given in Construction 1 is a perfect (resp. statistical, computational) SSS which realizes AS = AS_1 ∪ AS_2.

Proof. The reader is referred to [13] for the perfect case. We only consider the statistical and the computational cases.

From the construction, SSS(II, Re) evidently satisfies the statistical (resp. computational) correctness requirement. We only need to consider the security requirement. For any s ∈ S and k ∈ AS, that is, B ∈ AS_1 and B /∈ AS_2, we observe that II(s, R)|_B = (II_1(s, R_1)|_B, II_2(s, R_2)|_B), where R = (R_1, R_2) is a couple of independent random variables. Then for any α ∈ (S_1^{(1)} × · · · × S_n^{(1)})|_B, β ∈ (S_1^{(2)} × · · · × S_n^{(2)})|_B, and for any s, s’ ∈ S, we have

\[
\frac{1}{2} \sum_{\alpha, \beta} |\Pr[II(s, R)|_B = (\alpha, \beta)] - \Pr[II(s’, R)|_B = (\alpha, \beta)]|
\]

\[
= \frac{1}{2} \sum_{\alpha, \beta} |\Pr[II_1(s, R_1)|_B = \alpha] \Pr[II_2(s, R_2)|_B = \beta] - \Pr[II_1(s’, R_1)|_B = \alpha] \Pr[II_2(s’, R_2)|_B = \beta]|
\]

\[
\leq \frac{1}{2} \sum_{\alpha, \beta} |\Pr[II_1(s, R_1)|_B = \alpha] \Pr[II_2(s, R_2)|_B = \beta] - \Pr[II_1(s, R_1)|_B = \alpha] |
\]

\[
\times \Pr[II_2(s’, R_2)|_B = \beta]| + \frac{1}{2} \sum_{\alpha, \beta} |\Pr[II_1(s, R_1)|_B = \alpha] \Pr[II_2(s, R_2)|_B = \beta]|
\]

\[
- \Pr[II_1(s’, R_1)|_B = \alpha] \Pr[II_2(s’, R_2)|_B = \beta]|
\]

\[
= \frac{1}{2} \sum_{\beta} |\Pr[II_2(s, R_2)|_B = \beta] - \Pr[II_2(s’, R_2)|_B = \beta]|
\]

\[
+ \frac{1}{2} \sum_{\alpha} |\Pr[II_1(s, R_1)|_B = \alpha] - \Pr[II_1(s’, R_1)|_B = \alpha]|.
\]

The second equality follows from the fact

\[
\sum_{\alpha} \Pr[II_1(s, R_1)|_B = \alpha] = \sum_{\beta} \Pr[II_2(s’, R_2)|_B = \beta] = 1.
\]

Because SSS_1, SSS_2 are statistical and B /∈ AS = AS_1 ∪ AS_2, we see that SSS also has statistical security.

Finally, we consider the computational case. For any B /∈ AS, any probabilistic polynomial-time algorithm T and any s, s’ ∈ S, we claim |Pr[T(II_1(s, R_1)|_B, II_2(s, R_2)|_B, 1^n) = 1] - Pr[T(II_1(s’, R_1)|_B, II_2(s’, R_2)|_B, 1^n) = 1]| < μ(n), for some negligible function μ. Otherwise there exists B /∈ AS, a probabilistic polynomial-time algorithm D, s, s’ ∈ S and positive polynomial p(·), such that |Pr[D(II_1(s, R_1)|_B, II_2(s, R_2)|_B, 1^n) = 1] - Pr[D(II_1(s’, R_1)|_B, II_2(s, R_2)|_B, 1^n) = 1]| > 1/p(n), for infinitely many n’s. Thus we can construct a polynomial-time algorithm D’ to distinguish II_1(s, R_1)|_B and II_1(s’, R_1)|_B by using D as a subroutine. More precisely, on the input α ∈ (S_1^{(1)} × · · · × S_n^{(1)})|_B, D’ randomly selects β ∈ (S_1^{(2)} × · · · × S_n^{(2)})|_B according to the distribution of II_2(s, R_2)|_B and initiates D with the input (α, β, 1^n). Finally, D’ outputs what D returns. Then we have |Pr[D’(II_1(s, R_1)|_B, 1^n) = 1] - Pr[D’(II_1(s’, R_1)|_B, 1^n) = 1]| = |Pr[D(II_1(s, R_1)|_B, II_2(s, R_2)|_B, 1^n) = 1] - Pr[D(II_1(s’, R_1)|_B, II_2(s’, R_2)|_B, 1^n) = 1]| > 1/p(n), for infinitely many n’s, this contradicts to the hypothesis that SSS_1 is a computational SSS and B /∈ AS_1. Similarly for any probabilistic polynomial-time algorithm T and any s, s’ ∈ S, we have

|Pr[T(II_1(s’, R_1)|_B, II_2(s, R_2)|_B, 1^n) = 1] - Pr[T(II_1(s’, R_1)|_B, II_2(s’, R_2)|_B, 1^n) = 1]| < μ(n).
Thus,
\[
\begin{align*}
|\Pr[T(II(s, R)|_B, 1^n) = 1] - \Pr[T(II(s', R)|_B, 1^n) = 1]| \\
= |\Pr[T(II_1(s, R)|_B, II_2(s, R)|_B, 1^n) = 1] - \Pr[T(II_1(s', R)|_B, II_2(s', R)|_B, 1^n) = 1]| \\
\leq & |\Pr[T(II_1(s, R)|_B, II_2(s, R)|_B, 1^n) = 1] - \Pr[T(II_1(s', R)|_B, II_2(s, R)|_B, 1^n) = 1]| \\
& + |\Pr[T(II_1(s', R)|_B, II_2(s, R)|_B, 1^n) = 1] - \Pr[T(II_1(s', R)|_B, II_2(s', R)|_B, 1^n) = 1]| \\
< & 2\mu(n).
\end{align*}
\]

Hence the computational security requirement is satisfied. □

Note that Construction 1 needs two independent random variables \(R_1\) and \(R_2\). To realize \(AS\), it is possible but sometimes difficult to design a new SSS independent of \(SSS_1\) and \(SSS_2\) by using less randomness. Furthermore, some improvements can be made to reduce the size of shares. For example, the dealer can send \(s_{1i}\) to \(P_i\) for \(1 \leq i \leq n\), and send \(s_{2j}\) to \(P_j\) only when \(P_j\) belongs to some \(A' \in AS_2\setminus AS_1\). However, our construction is natural and convenient in both design and implementation. Furthermore, it preserves the perfect (resp. statistical, computational) security of original schemes, so it is more practical.

Let \(AS_1, AS_2, SSS_1(II_1, Re_1)\) and \(SSS_2(II_1, Re_2)\) be as before. We now construct an SSS which realizes \(AS = AS_1 \cap AS_2\) as follows.

**Construction 2.** Given a secret \(s \in S\), the dealer first uniformly selects \(s_1 \in S\). Then by using distribution functions \(II_1\) and \(II_2\), the dealer computes \(II_1(s_1, r_1) = (s_11, \ldots, s_1n)\) and \(II_2(s - s_1, r_2) = (s_21, \ldots, s_2n)\), where \(r_1, r_2\) are selected independently and uniformly in \(R_1\) and \(R_2\), respectively. After that, \(P_i\) gets his share \((s_{1i}, s_{2i})\) for \(1 \leq i \leq n\).

For the sake of convenience, we denote
\[
II : S \times R_1 \times R_2 \rightarrow S_1^{(1)} \times \cdots \times S_1^{(n)} \times S_2^{(1)} \times \cdots \times S_2^{(n)},
\]
\[
(s, r_1, r_2) \mapsto (II_1(s_1, r_1)|_{\{p_1\}}, \ldots, II_1(s_1, r_1)|_{\{p_n\}}, II_2(s - s_1, r_2)|_{\{p_1\}}, \ldots, II_2(s - s_1, r_2)|_{\{p_n\}}).
\]

Note that \(s_1\) is a randomly chosen element in \(S\) and should be included in the variables of \(II\). Here we omit it from the variables for the coherence of expressions.

**Proposition 2 (Intersection of access structures).** Suppose that \(SSS_1\) and \(SSS_2\) are perfect (resp. statistical, computational) SSSs which realize \(AS_1\) and \(AS_2\), respectively. Then the scheme SSS given in Construction 2 is a perfect (resp. statistical, computational) SSS which realizes \(AS = AS_1 \cap AS_2\).

**Proof.** The reader is referred to [13] for the perfect case. For the other two cases, it is easy to show that SSS satisfies the statistical (resp. computational) correctness requirement by the hypothesis that \(SSS_1\) and \(SSS_2\) are statistical (resp. computational).

As for the security requirement, we first observe that for any \(B \notin AS\) and \(s \in S\), \(II(s, R)|_B = (II_1(s_1, R_1)|_B, II_2(s - s_1, R_2)|_B)\), where \(s_1\) is uniformly chosen in \(S\) and \(R = (R_1, R_2)\) is a couple of independent random variables. Without loss of generality, we may assume that \(B \notin AS_2\). Then for any \(s, s' \in S\), and any \(\alpha \in \langle S_1^{(1)} \times \cdots \times S_1^{(n)} \rangle|_B, \beta \in \langle S_2^{(1)} \times \cdots \times S_2^{(n)} \rangle|_B\), we have
\[
\frac{1}{2} \sum_{\alpha, \beta} |\Pr[II(s, R)|_B = (\alpha, \beta)] - \Pr[II(s', R)|_B = (\alpha, \beta)]| \\
= \frac{1}{2} \sum_{\alpha, \beta} \frac{1}{|S|} \sum_{s_1 \in S} |\Pr[(II_1(s_1, R_1)|_B, II_2(s - s_1, R_2)|_B) = (\alpha, \beta)]| \\
- |\Pr[(II_1(s_1, R_1)|_B, II_2(s' - s_1, R_2)|_B) = (\alpha, \beta)]| \\
\leq \frac{1}{|S|} \sum_{s_1 \in S} \left( \frac{1}{2} \sum_{\alpha, \beta} |\Pr[(II_1(s_1, R_1)|_B, II_2(s - s_1, R_2)|_B) = (\alpha, \beta)]| \\
- |\Pr[(II_1(s_1, R_1)|_B, II_2(s' - s_1, R_2)|_B) = (\alpha, \beta)]| \right).
\]
Similar to the proof of Proposition 1, we can prove that (1) is negligible. Hence, the security requirement is satisfied in the statistical case.

Similarly, for any probabilistic polynomial-time algorithm $T$, we have that $\Pr[T(\Pi(s, R)|B, 1^n) = 1] = (1/|S|)\sum_{s_1 \in S} \Pr [T(\Pi_1(s_1, R)|B, \Pi_2(s - s_1, R_2)|B, 1^n) = 1]$. Thus for any $s, s' \in S$, we have

$$
\left| \Pr[T(\Pi(s, R)|B, 1^n) = 1] - \Pr[T(\Pi(s', R)|B, 1^n) = 1] \right| \\
\leq \frac{1}{|S|} \sum_{s_1 \in S} \left| \Pr[T(\Pi_1(s_1, R)|B, \Pi_2(s - s_1, R_2)|B, 1^n) = 1] \\
- \Pr[T(\Pi_1(s_1, R)|B, \Pi_2(s' - s_1, R_2)|B, 1^n) = 1] \right|.
$$

(2)

Again similar to the proof of Proposition 1, we can prove that (2) is negligible. Hence, SSS satisfies the security requirement in the computational case. □

Although Constructions 1 and 2 seem trivial, they can be used to construct an SSS which realizes the general access structure, so they are important in both theory and practice. We will show this point in Section 5.

4. Substitution of participants and other rearrangements

In real life, a participant can sometimes be replaced by several other participants. Suppose that $P' = \{P_1', \ldots, P_n'\}$ is a set of participants and $A S'$ is an access structure over $P'$. Then $SSS'(\Pi', Re')$ is an SSS which realizes $A S'$ with the secret-domain $S'$ and the share-domain $S_i'$ for $1 \leq i \leq n$, that is, $\Pi' : S' \times R' \rightarrow S'_1 \times \cdots \times S'_n$. Suppose that $P'' = \{P_1'', \ldots, P_m''\}$ is another set of participants with an access structure $A S''$ and $SSS''(\Pi'', Re'')$ is an SSS which realizes $A S''$ with the secret-domain $S_i''$, that is, $\Pi'' : S''_1 \times R'' \rightarrow S''_1 \times \cdots \times S''_m$. Replacing $P_1'$ by $P''$, we get the new set of participants $P = \{P_1'', \ldots, P_m, P_2', \ldots, P_n'\}$ and define an access structure over $P$ by

$$
A S = \{A \subseteq P|A \cap P' \subseteq A S'\} \cup \{A \subseteq P|A \cap P'' \subseteq A S''\} \cup (A \cap P') \cup \{P'_1\} \subseteq A S'.
$$

We now construct an secret sharing scheme $SSS(\Pi, Re)$ for the realization of $A S$ with the secret-domain $S'$.

Construction 3. For any given secret $s' \in S'$, the dealer firstly computes $\Pi'(s', r') = (s'_1, \ldots, s'_n)$. Then with the first share $s'_1$, he computes $\Pi''(s'_1, r'') = (s''_1, \ldots, s''_m)$, where $r'$ and $r''$ are independently and uniformly selected in $R'$ and $R''$, respectively. Finally, he secretly sends $s'_1$ to $P'_1$, $2 \leq i \leq n$, and sends $s''_j$ to $P''_j$, $1 \leq j \leq m$. For the sake of convenience, we denote

$$
\Pi : S' \times R' \times R'' \rightarrow S'_2 \times \cdots \times S'_n \times S''_1 \times \cdots \times S''_m,
$$

$$
(s', r', r'') \rightarrow (\Pi'(s', r')|_{P'_1}, \ldots, \Pi'(s', r')|_{P'_n}, \Pi''(\Pi'(s', r')|_{P'_1}, r'')|_{P''_1}, \ldots, \Pi''(\Pi'(s', r')|_{P'_n}, r'')|_{P''_m}).
$$

Then for $2 \leq i \leq n, P'_i$ has share-domain $S'_i$ and for $1 \leq j \leq m, P''_j$ has share-domain $S''_j$.

Proposition 3 (Substitution of participants). Suppose that $SSS'$ and $SSS''$ are perfect (resp. statistical, computational) SSSs which realize $A S'$ and $A S''$, respectively. Then the scheme $SSS(\Pi, Re)$ given in Construction 3 is a perfect (resp. statistical, computational) SSS which realizes $A S$.

Proof. The reader is referred to [13] for the perfect case. Similar to the proof of Proposition 2, the correctness of the SSS can be easily proved and the proof is omitted. We next argue the security requirement. For any $B \notin A S$, there are two cases to be considered:

1. $(B \cap P') \cup \{P'_1\} \notin A S'$ and $B \cap P'' \in A S''$. For any $s, s' \in S'$ and any possible value of $\Pi(s, R)|B$, denoted by $x$, we have

$$
\Pr[\Pi(s, R)|B = x] = \sum_{a \in S'_i} \Pr[\Pi'(s, R')|B \cap P', \Pi''(u, R'')|B \cap P'' = (x', x'')] \cdot \Pr[\Pi'(s, R')|P'_1 = u],
$$

(3)
where \( \mathcal{Z}' = x|_{B \cap P'} \), \( \mathcal{Z}'' = x|_{B \cap P''} \) and \( R = (R', R'') \) is a couple of independent random variables. So it holds that

\[
\frac{1}{2} \sum_{x} |\Pr[II(s, R)|B = x] - \Pr[II(s', R)|B = x]| \\
\leq \frac{1}{2} \sum_{x'} \sum_{u \in \mathcal{S}_1'} |\Pr[II'(s', R')|B \cap P' = x'] \Pr[II'(s', R')|\{P'_i\} = u] \\
- \Pr[II'(s', R')|B \cap P' = x'] \Pr[II'(s', R')|\{P'_i\} = u]| \sum_{x''} \Pr[II''(u, R'')|B \cap P'' = x''] \\
= \frac{1}{2} \sum_{x, u} |\Pr[II'(s', R')|B \cap P' = x'] \Pr[II'(s', R')|\{P'_i\} = u] \\
- \Pr[II'(s', R')|B \cap P' = x'] \Pr[II'(s', R')|\{P'_i\} = u]|,
\]

(4)

where the last equality follows the fact that \( \sum_{x''} \Pr[II''(u, R'')|B \cap P'' = x''] = 1 \) for any given \( u \in \mathcal{S}_1' \). Similar to the proof of Proposition 1, we can show that (4) is negligible.

Now we consider the computational security. Suppose on the contrary, that there exist \( s', s'' \in \mathcal{S}' \), a probabilistic polynomial-time algorithm \( T \) and a positive polynomial \( p(\cdot) \) such that |\Pr[T(I(s', R)|B, 1^n) = 1| - \Pr[T(I(s'', R)|B, 1^n) = 1|/p(n), for infinitely many \( n \)'s. By using \( T \) as a subroutine, we can construct a polynomial-time algorithm \( T' \) to distinguish \( II'(s', R')|\{B \cap P', \{P'_i\}\} \) and \( II'(s'', R')|\{B \cap P', \{P'_i\}\} \). But the \( \mathcal{S}' \) is a computational \( \mathcal{S} \) and \( (B \cap P') \cup \{P'_i\} \notin \mathcal{S}' \), we get a contradiction.

(2) \( B \cap P' \notin \mathcal{S}' \) and \( B \cap P'' \notin \mathcal{S}' \). In this case, we still first consider the statistical case. As in case (1), we have Eq. (3). For any given \( z \) and \( u \), where \( z \) and \( u \) are as before, define \( \xi_{z,u} = \Pr[II'(s, R')|B \cap P', II'(u, R'')|B \cap P'' = (z', x'')] \). Hence, we get a multi-set \( D_{z} = \{\xi_{z,u} | u \in \mathcal{S}_1'\} \). Define a random variable \( \xi_z \) over \( D_z \) with the probability distribution \( \Pr[\xi_z = \xi_{z,u}] = \Pr[II'(s, R')|\{P'_i\} = u] \). This random variable is well-defined because \( \sum_u \Pr[II'(s, R')|\{P'_i\} = u] = 1 \) for any given \( s \in \mathcal{S}' \). Thus Eq. (3) can be transformed into \( \Pr[II(s, R)|B = z] = E_{\xi_z} \). Similarly, we can define a random variable \( \eta_z \) with \( \Pr[\eta_z = \Pr[II'(s', R')|B \cap P', II'(u', R'')|B \cap P'' = (z', x'')] = \Pr[II'(s', R')|\{P'_i\} = u] \). Also, we get \( \Pr[II(s, R)|B = z] = E_{\eta_z} \). Thus

\[
\frac{1}{2} \sum_{z} |\Pr[II(s, R)|B = z] - \Pr[II(s', R)|B = z]| \leq \frac{1}{2} \sum_{z} |E_{\xi_z} - E_{\eta_z}| \\
\leq \frac{1}{2} \sum_{z} \|\xi_{z} - \eta_{z}\|.
\]

Let \( \xi_{z,u} = \Pr[II'(s, R')|B \cap P', II'(u, R'')|B \cap P'' = (z', x'')] \) and \( \eta_{z,u} = \Pr[II'(s', R')|B \cap P', II'(u', R'')|B \cap P'' = (z', x'')] \) be any given samples of \( \xi_z \) and \( \eta_z \), respectively. Because \( \mathcal{S}' \) and \( \mathcal{S}'' \) are statistical \( \mathcal{S} \)'s and \( B \cap P' \notin \mathcal{S}' \), \( B \cap P'' \notin \mathcal{S}' \), similar to the proof of Proposition 1, we can show that \( \frac{1}{2} \sum_{z} \|\xi_{z,u} - \eta_{z,u}\| \) is negligible. Since there are only finite samples, \( E_{\frac{1}{2}} \sum_{z} \|\xi_{z} - \eta_{z}\| \) is negligible.

As to the computational case, we observe that for any probabilistic polynomial-time algorithm \( T \), \( \Pr[T(II(s, R)|B, 1^n) = 1] = \sum_{u \in \mathcal{S}_1'} \Pr[T(II(s, R')|B \cap P', II'(u, R'')|B \cap P'' | \{P'_i\} = u] \cdot \Pr[II'(s, R')|\{P'_i\} = u] \). By using the method similar to that in the statistical case above, i.e. by regarding the \( \Pr[T(II(s, R)|B, 1^n) = 1] \) as an expected value of a random variable, we can prove the computational security in this case. \( \square \)

In the end, we just mention some other rearrangements that are sometimes encountered in real life. Because the proofs and constructions are either simple or similar to those of the previous ones, we omit the details.

**Proposition 4 (Contraction of access structures).** Let \( P = \{P_1, \ldots, P_n\} \) be the set of participants, \( AS \) an access structure over \( P \) and \( SSS(II, Re) \) the \( SSS \) which realizes \( AS \). Let \( P' = \{P_{i_1}, \ldots, P_{i_m}\} \subseteq P \) and \( AS' = \{A \subseteq P' | A \in AS\} \). Assume that \( II' = II|P' \), i.e. \( II' : S \times R \rightarrow S_{i_1} \times \cdots \times S_{i_m}, II'(s, r) = (s_{i_1}, \ldots, s_{i_m}) \). Then \( II' \) with the corresponding reconstruction algorithm \( Re \) is a perfect (resp. statistical, computational) \( SSS \) which realizes \( AS' \) over \( P' \), when \( SSS(II, Re) \) is perfect (resp. statistical, computational).
Proposition 5 (Reducing authorized sets). Let \( P = \{P_1, \ldots, P_n\} \) be the set of participants and \( AS \) an access structure over \( P \). Let \( SSS(\Pi, Re) \) be the SSS which realizes \( AS \). Then \( AS' = AS - 2^{\{P_1, \ldots, P_i\}} \) is another access structure over \( P \), and we have a perfect (resp. statistical, computational) SSS which realizes \( AS' \) based on the original perfect (resp. statistical, computational) scheme \( SSS(\Pi, Re) \).

Certainly there are many other rearrangements of access structures and some of which can be considered as combinations of the cases in this paper. Hence, we can also give some constructions for them in the perfect, statistical and computational cases.

5. Decomposition and realization of general access structures

Based on the rearrangements and constructions in Section 3, we now design the SSS for the general access structure by rearranging some basic access structures through union and intersection only. We first recall that any access structure \( AS \) over \( P = \{P_1, \ldots, P_n\} \) can be written as \( AS = cl(\{A_1, \ldots, A_r\}) \), where \( r \in \mathbb{N} \) and \( A_i \subseteq P \) for \( 1 \leq i \leq r \). Then the following lemma is an obvious but important result concerning the union and intersection of access structures. We here omit the details.

Lemma 1. Let \( \mathcal{A} \) and \( \mathcal{B} \) be two access structures over \( P = \{P_1, \ldots, P_n\} \), where \( \mathcal{A} = cl(\{A_1, \ldots, A_r\}) \) and \( \mathcal{B} = cl(\{B_1, \ldots, B_s\}) \) for some \( r, s \in \mathbb{N} \), \( A_i \subseteq P, 1 \leq i \leq r, 1 \leq j \leq s \). Then \( \mathcal{A} \cup \mathcal{B} = cl(\{A_1, \ldots, A_r, B_1, \ldots, B_s\}) \), and \( \mathcal{A} \cap \mathcal{B} = cl(\{A_1 \cap B_1, \ldots, A_1 \cap B_s, \ldots, A_r \cap B_1, \ldots, A_r \cap B_s\}) \).

The result of Lemma 1 can be easily extended to union and intersection of more than two access structures. In order to generate all access structures over \( P = \{P_1, \ldots, P_n\} \), we define the basic access structures as \( AS_i = cl(\{P_i\}) \) for \( 1 \leq i \leq n \), that is, \( AS_i \) consists of all subsets of \( i \)-th participant \( P_i \). The following proposition claims that \( AS_1, \ldots, AS_n \) actually generate all access structures through union “∪” and intersection “∩”.

Proposition 6. Any access structure over \( P \) can be decomposed into the basic access structures \( AS_1, \ldots, AS_n \) connected through union “∪” and intersection “∩”.

Proof. Suppose that \( AS \) is an arbitrary access structure over \( P \). Then \( AS \) can be written as \( AS = cl(\{A_1, \ldots, A_m\}) \), where \( m \in \mathbb{N} \) and \( A_i \subseteq P \) for \( 1 \leq i \leq m \). By Lemma 1, it follows that

\[
AS = cl(\{A_1\}) \cup \cdots \cup cl(\{A_m\}).
\] (5)

It suffices to show for every \( i \), \( cl(\{A_i\}) \) has the required form of decomposition.

Since \( A_i \subseteq P \), we can assume that \( A_i \) has the form \( \{P_{i_1}, \ldots, P_{i_k}\} \), where \( k \in \mathbb{N} \) and \( 1 \leq i_1 < i_2 < \cdots < i_k \leq n \). Clearly, \( cl(\{A_i\}) = cl(\{P_{i_1}, \ldots, P_{i_k}\}) = cl(\{P_{i_1}, \ldots, P_{i_{k-1}}\}) \cap AS_{i_k} = AS_{i_1} \cap \cdots \cap AS_{i_k} \cap AS_{i_{k-1}} \), where \( AS_{i_1} \cap \cdots \cap AS_{i_k} \cap \cdots \cap AS_{i_{k-1}} \). Hence for \( 1 \leq i \leq m \), \( cl(\{A_i\}) \) can be decomposed as a combination of \( AS_1, \ldots, AS_n \) through union and intersection. Finally, by Eq. (5) and Lemma 1, the proposition is proved. \( \Box \)

For simplicity, we say that \( AS \) can be expressed by \( AS_1, \ldots, AS_n \), if it can be decomposed into \( AS_1, \ldots, AS_n \) connected by union “∪” and intersection “∩”. The following proposition indicates that the basic access structures can be uniquely decided.

Proposition 7. Suppose that \( \mathcal{A}_1, \ldots, \mathcal{A}_m \) are \( m \) access structures over \( P \) such that any access structure over \( P \) can be expressed by \( \mathcal{A}_1, \ldots, \mathcal{A}_m \), while \( \mathcal{A}_j \) cannot be expressed by \( \mathcal{A}_1, \ldots, \mathcal{A}_{j-1}, \mathcal{A}_{j+1}, \ldots, \mathcal{A}_m \) for \( 1 \leq i \leq m \). Then \( m = n \) and \( \mathcal{A}_i = AS_i \) under the proper labels, where \( AS_i = cl(\{P_i\}) \).

Proof. First, we argue that for \( 1 \leq i \leq n \), \( AS_i \) cannot be decomposed any more, that is, if \( AS_i = \mathcal{B}_1 \cup \mathcal{B}_2 \) or \( AS_i = \mathcal{B}_1 \cap \mathcal{B}_2 \), then either \( \mathcal{B}_1 = AS_i \) or \( \mathcal{B}_2 = AS_i \). (In this sense, we say that the decomposition is minimal.) Suppose on the contrary, that (1) \( AS_i = \mathcal{B}_1 \cup \mathcal{B}_2 \) with \( \mathcal{B}_1 \neq AS_i \) and \( \mathcal{B}_2 \neq AS_i \). It follows that \( \mathcal{B}_1 \not\subseteq AS_i \) and \( \mathcal{B}_2 \not\subseteq AS_i \). Because \( \{P_i\} \in AS_i \), we have \( \{P_i\} \in \mathcal{B}_1 \) or \( \{P_i\} \in \mathcal{B}_2 \). Without loss of generality, we may assume that \( \{P_i\} \in \mathcal{B}_1 \). Then by the monotone ascending property, \( AS_i \subseteq \mathcal{B}_1 \) which contradicts to \( \mathcal{B}_1 \not\subseteq AS_i \); (2) \( AS_i = \mathcal{B}_1 \cap \mathcal{B}_2 \) with \( \mathcal{B}_1 \neq AS_i \) and \( \mathcal{B}_2 \neq AS_i \).
Therefore, AS₁ ⊆ G₁ and AS₂ ⊆ G₂. Take A ∈ G₁ − AS₁ and B ∈ G₂ − AS₂, i.e., A ∈ G₁, B ∈ G₂ and P₁ /∈ A ∪ B. Then A ∪ B ∈ (G₁ ∩ G₂) − AS₁ contradicts to AS₁ = G₁ ∩ G₂.

Since AS₁ cannot be decomposed any more and AS₂ can be expressed by A₁, . . . , Aₘ, AS₁ = Aᵢ holds for some i ∈ {1, . . . , m}. Under proper labels, we have Aᵢ = AS₁ for 1 ≤ i ≤ n and n ≤ m. Now, by Proposition 6 and the hypothesis that Aᵢ cannot be expressed by A₁, . . . , Aᵢ−₁, Aᵢ+₁, . . . , Aₘ, we get m = n. □

Before constructing an SSS with respect to the general access structure, we give an (perfect) SSS which realizes the basic access structure ASᵢ for 1 ≤ i ≤ n.

**Construction 4.** Suppose that P = {P₁, . . . , Pₙ} and AS = cl({Pᵢ}) for some i ∈ N and 1 ≤ i ≤ n. An SSS which realizes AS with the secret-domain S is constructed as follows: For any given s ∈ S, the dealer selects r₁, . . . , rᵢ−₁, rᵢ+₁, . . . , rₙ independently and uniformly from S. Then he sends s to P₁ and rᵢ to Pⱼ for 1 ≤ j ≤ n and j ≠ i.

Since for any access structure over P, there are only union and intersection operations in the final decomposition, an SSS can be constructed based on Constructions 1, 2 and 4. Actually, the proof of Proposition 6 provides a general way to decompose an arbitrary access structure. When dealing with specific access structures, some simplifications can help to shorten the size of shares, i.e., to reduce data expansion. See the example below.

**Example 1.** Suppose that P = {P₁, P₂, P₃, P₄} and the access structure AS = cl({{P₁, P₂}, {P₁, P₄}, {P₂, P₃, P₄}}). We are to design an SSS to realize AS with secret-domain S which is a finite field. Define AS₁ = cl({{Pᵢ}}) for 1 ≤ i ≤ 4. Then according to the proof of Proposition 6, we have AS = (AS₁ ∩ AS₂) ∪ (AS₁ ∩ AS₃) ∪ (AS₂ ∩ AS₃ ∩ AS₄). By Simplifying the decomposition, we get

\[
AS = [AS₁ ∩ (AS₂ ∪ AS₄)] ∪ (AS₂ ∩ AS₃ ∪ AS₄). \tag{6}
\]

Note that AS₂ ∪ AS₄ = cl({{P₂}, {P₄}}), AS₂ ∩ AS₃ ∪ AS₄ = cl({{P₂, P₃, P₄}}) and they both can be easily realized. Thus we define the basic access structures: A₁ = cl({{P₁}}), A₂ = cl({{P₂}, {P₄}}), and A₃ = cl({{P₂, P₃, P₄}}). Hence AS = (A₁ ∩ A₂) ∪ A₃ and the procedure of constructing an SSS for the secret s ∈ S is displayed below, where r₁, r₂ and r₃ are selected independently and uniformly in S.

<table>
<thead>
<tr>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₁</td>
<td>r₂</td>
<td>r₃</td>
<td>r₄</td>
</tr>
<tr>
<td>s − r₁</td>
<td>r₂</td>
<td>r₃</td>
<td>s − r₂ − r₁</td>
</tr>
<tr>
<td>s − r₃</td>
<td>r₁</td>
<td>r₃</td>
<td>s − r₂ − r₁</td>
</tr>
<tr>
<td>s − r₃</td>
<td>r₁</td>
<td>s − r₂</td>
<td>s − r₃</td>
</tr>
</tbody>
</table>

That is, P₁ has share s − r₁, P₂ has r₂ and r₃, P₃ has r₁ and P₄ has s − r₂ − r₁ and r₃.

There are some other decompositions and the corresponding realizations for AS. For instance, AS can be decomposed as

\[
AS = cl({{P₁, P₂}}) \cup cl({{P₂, P₃, P₄}}) \cup cl({{P₁, P₄}}),
\]

and the corresponding scheme can be described in the following table, where r₁, r₂, r₃ and r₄ are selected independently and uniformly in S:

<table>
<thead>
<tr>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₁</td>
<td>r₂</td>
<td>r₃</td>
<td>r₄</td>
</tr>
<tr>
<td>s − r₁</td>
<td>s − r₂</td>
<td>s − r₂ − r₁</td>
<td>s − r₄</td>
</tr>
<tr>
<td>r₁</td>
<td>s − r₁</td>
<td>s − r₂</td>
<td>s − r₄</td>
</tr>
<tr>
<td>s − r₁</td>
<td>r₂</td>
<td>s − r₂ − r₁</td>
<td>s − r₄</td>
</tr>
<tr>
<td>s − r₁</td>
<td>r₂</td>
<td>r₃</td>
<td>s − r₄</td>
</tr>
</tbody>
</table>

However, we observe that the former scheme uses less randomness, that is, it needs r₁, r₂ and r₃, while the latter scheme needs r₄ in addition, and the total size of shares in the former scheme is 6 log |S| while the latter is 7 log |S|, hence the former scheme is better. This is because that in the former scheme, we use the simplified decomposition (6).
It is obvious that in the final form of decomposition, the less minimum authorized sets $P_i$ is contained in, the shorter share he will get in our construction. Hence proper simplification for decompositions of access structures in some cases is desirable.

References