Secret Sharing for General Access Structures

İlker Nadi Bozkurt, Kamer Kaya, and Ali Aydın Selçuk

***Abstract*—Secret sharing schemes (SSS) are used to distribute a highly sensitive secret among a group of individuals so that only when an authorized group of them come together can the secret be reconstructed. The set of these authorized groups is called the access structure. If the access structure only contains all groups of size larger than a threshold value, the problem is called threshold secret sharing. There exist several efficient solutions in the literature proposed for threshold secret sharing and some methods adapt threshold SSSs to general secret sharing problems where the access structures can be more complex. In this paper, we are interested in the adaptation of threshold SSSs based on the Chinese Remainder Theorem (CRT) for general access structures. Galibus and Matveev (2007) proposed a solution for the same problem. We first modify their algorithm to work over integers rather than polynomials. Then we show that this modified algorithm is impractical and propose another one based on splitting the secret into multiple parts. Experimental results show that the proposed solution is better than the former in terms of the information rate.**

***Index Terms*—secret sharing, general access structures, Asmuth-Bloom secret sharing.**

I. Introduction

Secret sharing schemes (SSS) deal with the problem of the distribution of a highly sensitive secret, such as the private key of a cryptosystem, among a group of individuals so that only certain authorized groups of individuals can later reconstruct it. This problem and the first solutions to it were given by Shamir [13] and Blakley [3] in 1979. Shamir's solution to the secret sharing problem is based on polynomial interpolation over a finite field, whereas Blakley's SSS is based on hyperplane geometry. A fundamentally different SSS is Asmuth-Bloom SSS [1] which is based on the Chinese Remainder Theorem (CRT). This SSS uses a special sequence of integers (called Asmuth-Bloom sequences) for secret sharing.

In the original schemes of Shamir, Blakley and Asmuth-Bloom, the secret is shared among n users such that t or more users can reconstruct the secret but smaller groups cannot. These schemes are examples of (t, n) threshold secret sharing schemes. In the general secret sharing problem, authorization of groups may require more complex rules. General access structures, which we describe in the next section, are used for this purpose.

For general access structures, the challenge in using Asmuth-Bloom SSS is the generation of the appropriate Asmuth-Bloom sequence for the specific access structure at hand. Galibus and Matveev [5] proved that any access structure can be realized using Mignotte SSS [12], which is very similar to Asmuth-Bloom SSS. Originally, both Asmuth-Bloom sequences and Mignotte sequences are composed of pairwise coprime integers, while this condition is not true for their generalizations. Galibus et al. [5], [6] characterized the access structures which can be realized using non-generalized Asmuth-Bloom or Mignotte sequences and proposed a method to obtain generalized sequences for arbitrary access structures. The method of Galibus et al. works on polynomial rings and constructs generalized Mignotte sequences containing polynomials. In this paper, we show that with slight modifications, it can also be used to generate generalized Asmuth-Bloom sequences of integers.

As an alternative method to use Asmuth-Bloom SSS for general access structures, we propose a scheme based on secret splitting which uses Asmuth-Bloom SSS for the split parts. We also show how function sharing can be realized using this method and propose an RSA signature scheme for general access structures as an example.

The rest of the paper is organized as follows: In Section II, general and multipartite access structures are described. In Section III, Asmuth-Bloom SSS and some of its properties are given in detail. Section IV describes the method of Galibus and Matveev, and our modified version. In Section V, the splitting-based secret sharing scheme is proposed and an example function sharing scheme is described. Section VI concludes the paper.

II. General Access Structures

The *access structure* of a secret sharing scheme is the set of all groups which are allowed to reconstruct the secret. We will denote the access structure of a secret sharing scheme with Γ. The elements of the access structure are referred to as authorized groups (sets) and the rest are called unauthorized groups (sets). The set of all unauthorized groups is called the *adversary structure*. The adversary structure will be denoted by $\overline{\Gamma}$. As an example, for a (t, n) threshold access structure,

$$\Gamma = \{A \in 2^{\mathcal{P}} : |A| \geq t\},$$
$$\overline{\Gamma} = \{A \in 2^{\mathcal{P}} : |A| < t\},$$

where $2^{\mathcal{P}}$ is the power set of the user set $\mathcal{P} = \{1, 2, \ldots, n\}$.

Ito, Saito, and Nishizeki [10] showed that for every access structure, there is a secret sharing scheme. As they mentioned, it is reasonable to assume that, if a group can recover the secret, so can a larger group containing that group. Furthermore, if a group cannot recover the secret, neither can a smaller

İlker Nadi Bozkurt works as a researcher at TÜBİTAK-UEKAE, Ankara, Turkey, e-mail: ilker.nadi@uekae.tubitak.gov.tr.

Kamer Kaya is a postdoctoral researcher at CERFACS, France, e-mail: kamer@cerfacs.fr.

Ali Aydın Selçuk is with the Department of Computer Engineering, Bilkent University, Ankara, 06800, Turkey, e-mail: selcuk@cs.bilkent.edu.tr.

This work is supported in part by the Turkish Scientific and Technological Research Agency (TÜBİTAK), under grant number 108E150.

group contained by that group. Hence, for an access structure Γ,

$$(A \in \Gamma) \wedge (A \subseteq B) \Rightarrow B \in \Gamma, \quad (1)$$
$$(A \in \overline{\Gamma}) \wedge (B \subseteq A) \Rightarrow B \in \overline{\Gamma}. \quad (2)$$

An access structure is called *monotone* if it satisfies (1) and (2). The monotonicity of an access structure helps us to make its representation more compact. From now on, we assume that Γ only contains the *minimal* allowed groups which can recover the secret d. Similarly, $\overline{\Gamma}$ only contains *maximal* adversarial groups which cannot recover d. E.g., for a (t, n) access structure,

$$\Gamma = \{A \in 2^{\mathcal{P}} : |A| = t\},$$
$$\overline{\Gamma} = \{A \in 2^{\mathcal{P}} : |A| = t - 1\}.$$

We refer the reader to [2] for a detailed discussion on monotone access structures.

Let $\mathcal{P}$, the user set, be partitioned into $k \geq 2$ disjoint sets $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$. Each set $\mathcal{P}_i$ has n_i users where $n = \sum_{i=1}^{k} n_i$. An access structure is called *multipartite* if there exists a partition such that the users in each set $\mathcal{P}_i$ play the same role. Let σ be a random permutation of the numbers 1 to n. Formally, we say that an access structure Γ is k-partite if $\sigma(\Gamma) = \Gamma$ for any permutation σ of $\mathcal{P}$ satisfying $\sigma(\mathcal{P}_i) = \mathcal{P}_i$ for $1 \leq i \leq k$. Note that any access structure defined on a set of n users is trivially n-partite, because we can always take $\mathcal{P}_1 = \{1\}, \ldots, \mathcal{P}_n = \{n\}$. But, we usually want to consider the minimum possible number of classes. For a detailed discussion on multipartite access structures, the reader may consult [7].

A k-partite access structure can be represented by a set of k-tuple vectors, each vector denoting an authorized combination, and the ith entry in a vector denoting the required number of participants from $\mathcal{P}_i$ in that authorized combination. As an example, $\Gamma = \{(3, 4), (4, 2)\}$ means that a group is authorized if it contains at least 3 users from $\mathcal{P}_1$ and 4 from $\mathcal{P}_2$, or if it contains at least 4 users from $\mathcal{P}_1$ and 2 from $\mathcal{P}_2$. If we assume that there are 5 users in each part, the corresponding adversary structure for Γ is $\overline{\Gamma} = \{(2, 5), (5, 1), (3, 3)\}$.

Note that due to monotonicity, Γ and $\overline{\Gamma}$ contain only minimal and maximal vectors, respectively. I.e., if $v = (v_1, \ldots, v_k) \in \Gamma$ then a vector $v' = (v'_1, \ldots, v'_k)$ such that $v'_i \geq v_i$ for $1 \leq i \leq k$, is not in Γ. For example, for the above access structure, the vector $v' = (4, 4)$ corresponds to an authorized group but is not in Γ since $v = (3, 4)$, a subset is already in.

III. Asmuth-Bloom Secret Sharing Scheme

The Asmuth-Bloom SSS [1] is based on the CRT and uses a public sequence of coprime integers $m_0 < m_1 < \ldots < m_n$ such that

$$\prod_{i=1}^{t} m_i > m_0 \prod_{i=1}^{t-1} m_{n-i+1}. \quad (3)$$

Such a sequence is called a (t, n) Asmuth-Bloom sequence. Let $M = \prod_{i=1}^{t} m_i$. The secret sharing scheme works as follows:

- The secret d is chosen randomly from $\mathbb{Z}_{m_0}$.
- y is computed as $y = d + Am_0$, where A is a random positive integer such that $y < M$.
- The secret shares y_i are computed as $y_i = y \bmod m_i$ for all $1 \leq i \leq n$.
- Given t distinct shares $y_{i_1}, \ldots, y_{i_t}$, the secret d can be obtained as $d = y \bmod m_0$, where y is the unique solution modulo M of the system

$$\begin{aligned} y &\equiv y_{i_1} \pmod{m_{i_1}} \\ &\vdots \\ y &\equiv y_{i_t} \pmod{m_{i_t}}. \end{aligned}$$

Asmuth and Bloom proposed an iterative process to solve the above system of congruences [1]. Here, we use a non-iterative and direct method given in [4]. This method is more suitable for function sharing in the sense that it does not require interaction between parties and has an additive structure which is convenient for function sharing [11]. Suppose S is a coalition of t users gathered to construct the secret d.

1) Let $M_{S\setminus i}$ denote $\prod_{j \in S, j \neq i} m_j$ and $M'_{S,i}$ be the multiplicative inverse of $M_{S\setminus i}$ in $\mathbb{Z}_{m_i}$, i.e.,
$$M_{S\setminus i} M'_{S,i} \equiv 1 \pmod{m_i}.$$
Then, every user i computes
$$y_i = d_i M_{S\setminus i} M'_{S,i} \bmod M_S.$$
2) y is computed as
$$y = \sum_{i \in S} u_i \bmod M_S.$$
3) The secret d is computed as
$$d = y \bmod m_0.$$

A very similar secret sharing scheme is the Mignotte SSS [12]. This scheme also uses a special sequence of integers called Mignotte sequences. A (t, n) Mignotte sequence is a sequence of pairwise coprime integers $m_1 < \ldots < m_n$ such that

$$\prod_{i=1}^{t} m_i > \prod_{i=1}^{t-1} m_{n-i+1}.$$

Obviously if $m_0, m_1, \ldots, m_n$ is a (t, n) Asmuth-Bloom sequence, then $m_1, \ldots, m_n$ is a (t, n) Mignotte sequence.

Note that the numbers in Asmuth-Bloom and Mignotte sequences are selected as pairwise coprime. However, Iftene [8], [9] showed that this is not necessary and generalized these sequences. Here we describe only the generalized Asmuth-Bloom sequences; the generalization of Mignotte sequences is similar.

Definition 1. Let t and n be integers such that $1 \leq t \leq n$ and $\mathcal{P} = \{1, 2, \ldots, n\}$. Let $\mathcal{S}_j$ be the set of all subsets of $\mathcal{P}$ of cardinality j. The sequence of integers $m_0, m_1, \ldots, m_n$ is a (t, n) generalized Asmuth-Bloom sequence if it satisfies

$$m_0 \left(\max_{S \in \mathcal{S}_{t-1}} (\mathrm{lcm}(m_i, i \in S)) \right) < \min_{S \in \mathcal{S}_t} (\mathrm{lcm}(m_i, i \in S))$$

where lcm denotes the least common multiple of its inputs.

We conclude this section with an important property of $(\lceil n/2 \rceil, n)$ Asmuth-Bloom sequences. We will use this property in Section V while proposing a SSS based on secret splitting.

Lemma 1. An $(\lceil n/2 \rceil, n)$ Asmuth-Bloom sequence is a (k, n) Asmuth-Bloom sequence for all k such that $1 \leq k \leq n$.

Proof: Let $t = \lceil n/2 \rceil$.

Let $1 \leq k < t$. Rewriting (3) as

$$\prod_{i=1}^{k} m_i \prod_{i=k+1}^{t} m_i > m_0 \prod_{i=1}^{k-1} m_{n-i+1} \prod_{i=k+1}^{t} m_{n-i+2}$$

and observing $m_i \leq m_{n-i+2}$ for $k+1 \leq i \leq t$,

$$\prod_{i=1}^{k} m_i > m_0 \prod_{i=1}^{k-1} m_{n-i+1} \tag{4}$$

follows.

Now let $n \geq k > t$. Since (3) is satisfied and $m_i \geq m_{n-i+2}$ for $t+1 \leq i \leq k$,

$$\prod_{i=1}^{t} m_i \prod_{i=t+1}^{k} m_i > m_0 \prod_{i=1}^{t-1} m_{n-i+1} \prod_{i=t+1}^{k} m_{n-i+2}$$

and hence, (4) holds again. ∎

IV. Method of Galibus et al.

Although Asmuth-Bloom and Mignotte SSSs are proposed for integers, they can also be used with polynomial rings. The secret sharing and reconstruction processes are the same as the original ones except the secret d, the moduli m_i, and the shares d_i are polynomials. Existence and generation of Asmuth-Bloom and Mignotte sequences for the polynomial case was studied by Galibus et al. [5], [6]. They proved that for polynomials, any access structure can be realized by using Mignotte SSS and proposed an algorithm to produce a generalized Mignotte sequence for an arbitrary access structure [5]. Let $m_i(x)$ be the polynomial of user i for all $1 \leq i \leq n$ and Γ be the desired access structure. Algorithm 1 describes the algorithm of Galibus and Matveev [5].

Algorithm 1 Method of Galibus and Matveev

1: **for all** maximal unauthorized subsets $A \in \overline{\Gamma}$ **do**
2: $M = \min_{C \in \Gamma}(\deg(\operatorname{lcm}(p_i(x), i \in C)))$
3: **if** $\deg(\operatorname{lcm}(m_i(x), i \in A)) \geq M$ **then**
4: Find an irreducible monic polynomial $p(x)$ such that $\deg(p(x))$ is minimum and

$$\deg(\operatorname{lcm}(m_i(x), i \in A)) < \deg(\operatorname{lcm}(p_i(x) : i \in C))$$

for all $C \in \Gamma$ where

$$p_i(x) = \begin{cases} p(x)m_i(x) & \text{if } i \notin A \\ m_i(x) & \text{if } i \in A. \end{cases}$$

5: **for all** $i \notin A$ **do**
6: $m_i(x) = p_i(x)$

Initially each polynomial $m_i(x)$ is set to 1. Then, at each iteration of the Algorithm 1, a maximal unauthorized subset $A \in \overline{\Gamma}$ is processed. If A is at least as powerful as an authorized subset $C \in \Gamma$, which is not supposed to happen, the polynomials $m_i(x)$ of each user i not belonging to A are multiplied by some irreducible monic $p(x) \in GF(q)[x]$, where $GF(q)[x]$ denotes the polynomial ring and $GF(q)$ is the Galois field of prime order q. The polynomial $p(x)$ is chosen such that after all multiplications

$$\deg(\operatorname{lcm}(m_i(x), i \in A)) < \deg(\operatorname{lcm}(p_i(x), i \in C)), \tag{5}$$

is satisfied for all $C \in \Gamma$. As Galibus et al. show, (5) holds true when some other maximal unauthorized subset A' is chosen and the same operation is performed with another $p'(x) \neq p(x)$ at a subsequent iteration of the algorithm. Thus, after repeating the operation for all maximal unauthorized subsets, a Mignotte sequence for the given access structure Γ is obtained.

Our modifications to Algorithm 1 are as follows: We first choose m_0, a random prime number of specified bit length, at the beginning. Then at each step of the algorithm,

$$m_0 \operatorname{lcm}(m_i, i \in A) < \min_{C \in \Gamma}(\operatorname{lcm}(m_i, i \in C)), \tag{6}$$

is checked for the current unauthorized subset $A \in \overline{\Gamma}$ and for all $C \in \Gamma$. If the condition is satisfied, we do not update the moduli m_i, $1 \leq i \leq n$. But if the condition is not satisfied, first, a prime number p is chosen such that

$$m_0 \operatorname{lcm}(m_i, i \in A)) < \operatorname{lcm}(p_i, i \in C)$$

for all $C \in \Gamma$ where $p_i = pm_i$ if $i \notin A$, and $p_i = m_i$, otherwise. Second, all moduli not belonging to users from A are multiplied with p, i.e., m_i is set to p_i for $i \notin A$. After this operation is repeated for every maximal unauthorized subset, a generalized Asmuth-Bloom sequence $m_0, m_1, \ldots, m_n$ is obtained. The modified version of the algorithm for integers is given in Algorithm 2.

Algorithm 2 Modified GM algorithm for integers

1: Generate a prime m_0 of specified bit length
2: **for all** maximal unauthorized subset $A \in \overline{\Gamma}$ **do**
3: $M = \min_{C \in \Gamma}(\operatorname{lcm}(m_i, i \in C))$
4: **if** $m_0 \operatorname{lcm}(m_i, i \in A) \geq M$ **then**
5: Find a prime p such that, $|p|$, the bit length of p is minimum and

$$m_0 \operatorname{lcm}(m_i, i \in A) < \operatorname{lcm}(p_i, i \in C)$$

for all $C \in \Gamma$ where

$$p_i = \begin{cases} pm_i & \text{if } i \notin A \\ m_i & \text{if } i \in A \end{cases}$$

6: **for all** $i \notin A$ **do**
7: $m_i = p_i$

Algorithm 2 generates a generalized Asmuth-Bloom sequence for an arbitrary access structure. Nevertheless, our experiments indicate that it is not practical since the generated moduli in the sequence are too large. Table I shows the average and maximum bit lengths of the generalized Asmuth-Bloom sequences which are generated by Algorithm 2 for three different access structures and for m_0 of length 32, 64,

Bits	A = {(2 3)} s = (4 4)	A = {(2 3),(3 2)} s = (4 4)	A = {(2 3)} s = (5 5)
32	Max : 652 Avg : 327	Max : 7162 Avg : 5612	Max : 1489 Avg : 726
64	Max : 1324 Avg : 663	Max : 14554 Avg : 11404	Max : 3025 Avg : 1475
128	Max : 2668 Avg : 1336	Max : 29338 Avg : 22988	Max : 6097 Avg : 2973
256	Max : 5356 Avg: 2679	Max : 58906 Avg: 46156	Max : 12241 Avg: 5968
512	Max : 10732 Avg: 5367	Max : 56722 Avg: 92492	Max : 24529 Avg: 11958

TABLE I
MAXIMUM AND AVERAGE BIT LENGTHS OF THE GENERALIZED ASMUTH-BLOOM SEQUENCES GENERATED BY THE MODIFIED GM ALGORITHM.

128, 256, and 512 bits. The first row of the table shows the 2-partite access structures used for the experiments. Note that Algorithm 2 is deterministic and the average and maximum values are obtained by running the algorithm just once, not from multiple runs. As Table I shows, the maximum and average bit lengths of the moduli indicate that Algorithm 2 is not of practical value. The results when the number of parts is set to three are worse as the required number of multiplications increase significantly.

V. A SSS BASED ON SPLITTING

To use Asmuth-Bloom (or Mignotte) SSS for general access structures, here, we propose a different method using the corresponding multipartite access structures. Unlike Algorithm 2, the modified method of Galibus et al. for integer Asmuth-Bloom SSS, the proposed method generates more than one sequence.

The idea of the proposed method is simple: First, we find the smallest k value such that the access structure Γ is k-partite. We then partition the user set into k parts where each part i contains n_i users. Second, to share a secret d, for each vector $v = (v_1, v_2, \ldots, v_k) \in \Gamma$, we generate $d_{v,1}, d_{v,2}, \ldots, d_{v,k} \in \mathbb{Z}_{m_0}$ such that

$$\sum_{i=1}^{k} d_{v,i} \equiv d \bmod m_0. \tag{7}$$

We then share $d_{v,i}$, $i \in \{1, 2, \ldots, k\}$, between the users in part i by using a (v_i, n_i) Asmuth-Bloom SSS. Hence, if there are r vectors in Γ each user ends up with r shares. Note that to share a $d_{v,i}$, we need a (v_i, n_i) Asmuth-Bloom sequence. Although the v_i value is different for each vector v, due to Lemma 1, it suffices to generate one Asmuth-Bloom sequence for each part. Moreover, if there are parts with the same number of users, the same Asmuth-Bloom sequence can be reused. In the extreme case where all parts have the same size, just one Asmuth-Bloom sequence is generated and used in all parts. Algorithm 3 describes the process in detail.

As Algorithm 3 shows, for each part, we generate an Asmuth-Bloom sequence with common m_0. More specifically, for part i we generate an $(\lceil n_i/2 \rceil, n_i)$ Asmuth-Bloom sequence. Normally, for each part i and vector $v = (v_1, v_2, \ldots, v_k) \in \Gamma$, we need to generate a different (v_i, n_i) Asmuth-Bloom sequence. However, as Lemma 1 shows,

Algorithm 3 Splitting-based SSS for general access structures.

for all i, $1 \le i \le k$ **do**
  $t = \lceil n_i/2 \rceil$
  generate an Asmuth-Bloom sequence $m_0 < m_{j_1} < m_{j_2} < \ldots < m_{j_{n_i}}$ such that $\prod_{l=1}^{t} m_{j_l} > m_0 \prod_{l=1}^{t-1} m_{j_{n_i-l+1}}$
for all vectors $v = (v_1, v_2, \ldots, v_k) \in \Gamma$ **do**
  generate $d_{v,1}, d_{v,2}, \ldots, d_{v,k}$ such that $d_{v,1} + d_{v,2} + \ldots + d_{v,k} \equiv d \pmod{m_0}$
  for all i, $1 \le i \le k$ **do**
    $M_{v,i} = \prod_{l=1}^{v_i} m_{j_l}$
    let $A_{v,i}$ be a random positive integer such that $y_{v,i} = d_{v,i} + A_{v,i} m_0 \le M_{v,i}$
    for all l, $1 \le l \le n_i$ **do**
      compute, y_{v,j_l}, share of lth user of part i $y_{v,j_l} = y_{v,i} \bmod m_{j_l}$

an $(\lceil n_i/2 \rceil, n_i)$ Asmuth-Bloom sequence is also a (k, n_i) Asmuth-Bloom sequence for all $1 \le k \le n$. So, for a part i, the $(\lceil n_i/2 \rceil, n_i)$ sequence can be used for each different v_i hence, for each vector v.

On the other hand, for each vector $v \in \Gamma$, we have to find a distinct set of $d_{v,i}$, $1 \le i \le k$, such that $\sum_{i=1}^{k} d_{v,i} \equiv d \pmod{m_0}$. Otherwise, i.e. if we use the same summation more than once, unauthorized groups can reconstruct the secret d by combining their shares given for different vectors in the access structure. As an example consider an access structure $\Gamma = \{(5,3),(3,5)\}$ where the vector $(4,4)$ corresponds to an unauthorized group. Having 4 users from each part, such a group can find $d_{v,2}$ from the shares of given for vector $v = (5,3)$ and can find $d_{v',1}$ from the shares given for vector $v' = (3,5)$. If we use the same summation, i.e., $d_{v,1} = d_{v',1}$, combining $d_{v',1}$ and $d_{v,2}$ would yield the secret. Obviously, this situation has to be avoided. Since we cannot use only one summation for d, the information rate of the scheme is

$$\frac{\text{secret size}}{\text{max share size}} = \frac{1}{r},$$

where r is the number of vectors in Γ. Note that as Table I shows, the information rate is far better than that of Algorithm 2.

The proposed splitting-based SSS naturally suits to be used with function sharing schemes. In the next subsection, we show how RSA signatures can be computed when the secret is shared using the proposed splitting-based scheme.

A. A Threshold RSA scheme with the Proposed SSS

In [11], Kaya and Selcuk investigated how threshold cryptography can be conducted with the Asmuth-Bloom secret sharing scheme and presented function sharing schemes for RSA, ElGamal and Paillier cryptosystems. Here, we will follow their approach for realizing RSA signature computation using the proposed splitting-based SSS as the underlying scheme.

When an authorized group come together to sign a message, users from each part can compute their partial signatures as

in [11]. Then partial signatures coming from each part is multiplied to generate the incomplete signature. This product is incomplete because the method for RSA signature computation in [11] first calculates an incorrect signature which is then corrected by using a procedure that utilizes the public key. Here, the required housekeeping is a little more complicated. The computed partial signatures of each part will be incorrect. It is not possible to correct these partial signatures at this step since there is no information (i.e., no public keys or partial public keys for a part) to be used for correction. However, we can still use the correction procedure after the partial signatures are combined. In the following, we give the steps of RSA signature computation when the RSA private key d is shared using the proposed splitting-based secret sharing method.

- *Setup:* In the RSA setup phase, choose the RSA primes $p = 2p^{'}+1$ and $q = 2q^{'}+1$ where $p^{'}$ and $q^{'}$ are also large random primes. $N = pq$ is computed and the public key e and private key d are chosen from $\mathbb{Z}^*_{\phi(N)}$ where $ed \equiv 1 \pmod{\phi(N)}$. Use the above secret sharing scheme for sharing d with $m_0 = \phi(N) = 4p^{'}q^{'}$. Note that each user will receive r shares, which is the number of vectors in the access structure Γ.
- *Signing:* Let w be the hashed message to be signed and suppose the range of the hash function is $\mathbb{Z}^*_N$. Let Γ, the access structure, be k-partite. Assume a coalition S, containing t_i users from the ith part for $1 \leq i \leq k$, wants to obtain the signature $s = w^d \bmod N$. Let S_i denote the set of users in S from part i. For this coalition to be authorized, there has to be at least one vector $v = (v_1, v_2, \ldots, v_k)$ in Γ such that $v_i \leq t_i$ for all $i \in \{1, 2, \ldots, k\}$. Without loss of generality, we assume $v = (t_1, t_2, \ldots, t_k)$. To sign the message, the users in each S_i compute the partial signature $\overline{s_i}$ by using their shares $y_{v,j}$ for $j \in S_i$, with respect to the same vector v.
  - *Generating partial results*: Each user $j \in S_i$ computes

$$u_j = y_{v,j} M^{'}_{v,S_i,j} M_{v,S_i \backslash j} \bmod M_{v,S_i}$$
$$\sigma_j = w^{u_j} \bmod N,$$

    where M_{v,S_i} is the value used in Algorithm 3 to share $d_{v,i}$, $M_{v,S_i \backslash j} = M_{v,S_i}/m_j$, and $M^{'}_{v,S_i,j}$ is the inverse of it modulo m_j.
  - *Combining partial results:* The incomplete signature $\overline{s}$ is obtained by first computing the partial signature $\overline{s_i}$ for each part i, and combining, i.e., multiplying the results. Hence, for each part i

$$\overline{s_i} = \prod_{j \in S_i} \sigma_j \bmod N.$$
$$\overline{s} = \prod_{i=1}^{k} \overline{s_i} \bmod N.$$

  - *Correction:* Let $\kappa_{v,i} = w^{-M_{v,S_i}}$ be the corrector for part i. The incomplete signature can be corrected by trying

$$\left(\overline{s} \prod_{i=1}^{k} \kappa_{v,i}^{l_i} \right)^e = \overline{s}^e \prod_{i=1}^{k} \left(\kappa_{v,i}^e \right)^{l_i} \stackrel{?}{\equiv} w \pmod N \quad (8)$$

    for

$$\begin{aligned} 0 \leq \quad & l_1 \quad < t_1, \\ & \vdots \\ 0 \leq \quad & l_k \quad < t_k. \end{aligned}$$

    Then the signature is computed by

$$s = \overline{s} \kappa_1^{\delta_1} \kappa_2^{\delta_2} \ldots \kappa_k^{\delta_k}, \quad (9)$$

    where $\delta_1, \ldots, \delta_k$ denote the values of $l_1, \ldots, l_k$ that satisfy (8). For details of why this correction procedure is needed and why it works, the reader may consult [11].
- *Verification:* Verification is the same as the standard RSA signature verification.

VI. Conclusion

In this paper, we investigated ways of using Asmuth-Bloom SSS for general secret sharing. The main problem to devise such a scheme is finding a suitable Asmuth-Bloom sequence for the given access structure.

Galibus et al. [6] show that not all access structures can be realized using sequences consisting of pairwise coprime moduli. They call access structures that can be realized using pairwise coprime moduli as elementary access structures. For the secret sharing problem with general access structures, they proposed an algorithm which works in the ring of polynomials. Although a variant of this algorithm for the ring of integers, which we call the modified GM algorithm, can produce generalized integer Asmuth-Bloom sequences, our experiments showed that it produces very large moduli, seriously limiting the algorithm's practical value.

We proposed a splitting-based SSS for the same problem. The proposed scheme compares favorably to the above mentioned modified GM algorithm in terms of the information rate. Furthermore, function sharing schemes for general access structures can easily be implemented with the proposed scheme. We described a threshold RSA signature computation using the proposed scheme as an example.

Obtaining more compact Asmuth-Bloom sequences and hence improving the information rate of the proposed schemes in the literature are the main problems for future research.

References

[1] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29(2):208–210, 1983.

[2] J. C. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *Proc. of CRYPTO '88*, volume 403 of *LNCS*, pages 27–35. Springer-Verlag, 1990.

[3] G. Blakley. Safeguarding cryptographic keys. In *Proc. of AFIPS National Computer Conference*, 1979.

[4] C. Ding, D. Pei, and A. Salomaa. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. World Scientific, 1996.

[5] T. Galibus and G. Matveev. Generalized Mignotte's sequences over polynomial rings. *Electronic Notes on Theoretical Computer Science*, 186:43–48, 2007.

[6] T. Galibus, G. Matveev, and N. Shenets. Some structural and security properties of the modular secret sharing. In *Proc. of SYNASC'2008*, 2008.

[7] J. Herranz and G. Saez. New results on multipartite access structures. *IEE Proceedings of Information Security*, 153(4):153–162, 2006.

[8] S. Iftene. A generalization of Mignotte's secret sharing scheme. In *Proc. of 6th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 196–201. Mirton Publishing House, 2004.

[9] S. Iftene. *Secret Sharing Schemes with Applications in Security Protocols*. PhD thesis, University Alexandru Ioan Cuza of Iaşi, Faculty of Computer Science, 2007.

[10] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *Proc. of GLOBECOM'87*, pages 99–102. IEEE Press, 1987.

[11] K. Kaya and A. A. Selçuk. Threshold cryptography based on Asmuth-Bloom secret sharing. *Information Sciences*, 177(19):4148–4160, 2007.

[12] M. Mignotte. How to share a secret? In *Proc. of the Workshop on Cryptography*, volume 149 of *LNCS*, pages 371–375. Springer-Verlag, 1983.

[13] A. Shamir. How to share a secret? *Communications of the ACM*, 22(11):612–613, 1979.