Chapter 5

POB and Generalized Secret Sharing

5.1 Introduction

In the previous chapters we have seen the secret sharing schemes having threshold and generalized access structures. This chapter explores the construction of an efficient secret sharing scheme realizing the general access structure. An efficient \((n,n)\) threshold secret sharing scheme is proposed by Sreekumar et al in [202] using a specially designed number system called Permutation Ordered Binary (POB) number system. We are combining this scheme with the concept of cumulative arrays proposed by Ito et al in [107], to build secret sharing scheme with more generalized access structure.

In this chapter we give a brief introduction about general access

Some results of this chapter are included in the following paper.

structure based secret sharing schemes, more specifically cumulative arrays are dealt in detail. The POB construction and \((n, n)\) secret sharing using POB is explained next. The chapter ends with the algorithm of the proposed secret sharing scheme using POB and cumulative arrays.

There exist several monotone access structure for which there is no threshold scheme possible. Benaloh and Leichter had proven in [15] that, there are access structures that cannot be realized using threshold scheme. So secret sharing based on arbitrary monotone increasing access structure was a challenge. Several researchers address this problem and introduced secret sharing schemes realizing the general access structure. The most efficient and easy to implement scheme was Ito, Saito, Nishizeki’s [107] construction. It is based on Shamir’s scheme. The idea is to distribute shares to each authorized set of participants using multiple assignment scheme where more than one share is assigned to a participant if he belongs to more than one minimal authorized subset.

The disadvantage with multiple share assignment scheme is that the share size depends on the number of authorized set that contain \(P_j\). A simple optimization is to share the secret \(S\) only for minimal authorized sets. Still this scheme is inefficient. Benaloh and Leichter [15] developed a secret sharing scheme for an access structure based on monotone formula. This generalizes the multiple assignment scheme of Ito, Saito and Nishizeki [107]. The idea is to translate the monotone access structure into a monotone formula. Each variable in the formula is associated with a trustee in \(P\) and the value of the formula is \textit{true} if and only if the set of variables which are true corresponds to a subset of \(P\) which is in the access structure. This formula is then used as a template to describe how a secret is to be divided into shares.

Brickell [36] developed some ideal schemes for general access structure based secret sharing using vector spaces. Stinson [204] introduced a monotone circuit construction based on monotone formula and also the
5.2. Cumulative Secret Sharing Scheme

construction based on public distribution rules. Benaloh’s scheme was
generalized by Karchmer and Wigderson [116]. They showed that if an
access structure can be described by a small Monotone Span Program
(MSP), then it has an efficient Linear Secret Sharing Scheme (LSSS). The
proposed generalized secret sharing scheme make use of cumulative arrays
for the generalized secret sharing which is given in the next section.

5.2 Cumulative Secret Sharing Scheme

Cumulative secret sharing schemes provide a secret sharing capability using
an arbitrary access structure. Cumulative schemes were first introduced by
Ito et al [107] and then used by several authors to construct a general
scheme for arbitrary access structures. Simmons [199] proposed cumulative
map, Jackson [109] proposed the notion of cumulative array. Ghodosi et al
[80] introduced simple and more efficient scheme. The scheme also having
the capabilities to detect cheaters. Generalized cumulative arrays in secret
sharing is introduced by Long [139].

Definition 5.2.1. Let $\mathcal{A}$ be a monotone authorized access structure
on a set of participants $\mathcal{P}$. A cumulative scheme for the access structure $\mathcal{A}$
is map $\alpha : \mathcal{P} \rightarrow 2^S$, where $S$ is some set such that for any $\mathcal{A} \subseteq \mathcal{P}$,

$$\bigcup_{P_i \in \mathcal{A}} \alpha(P_i) = S$$

The scheme is represented using $|\mathcal{P}| \times |S|$ array $M = [m_{ij}]$, where row
$i$ of the matrix $M$ is indexed by $p_i \in P$ and column $j$ of the matrix $M$ is
indexed by an element $s_j \in S$, such that $m_{ij} = 1$ if and only if $P_i$ is given
$s_j$, otherwise $m_{ij} = 0$.

Definition 5.2.2. Let $\mathcal{A}$ be an access structure over the set of
participants $\mathcal{P} = \{P_1, \ldots, P_n\}$ and $\mathcal{A}_{\min} = \{A_1, \ldots, A_l\}$ is the set of all
minimal set of $\mathcal{A}$. Then the **incident array** of $\mathcal{A}$ is a $l \times n$ Boolean matrix $I_\mathcal{A} = [a_{ij}]$ defined by,

$$a_{ij} = \begin{cases} 1 & \text{if } P_j \in \mathcal{A}_i \\ 0 & \text{if } P_j \notin \mathcal{A}_i \end{cases}$$

for $1 \leq j \leq n$ and $1 \leq i \leq l$

**Definition 5.2.3.** Let $\mathcal{A}_{c_{\max}}^c = \{B_1, \ldots, B_m\}$ be the set of all maximal unauthorized sets. The **cumulative array** $C_\mathcal{A}$ for $\mathcal{A}$ is an $n \times m$ matrix $C_\mathcal{A} = [b_{ij}]$, where each row of the matrix is indexed by a participant $P_i \in \mathcal{P}$ and each column is indexed by a maximal unauthorized set $B_j \in \mathcal{A}_{c_{\max}}^c$, such that the entries $b_{ij}$ satisfy the following:

$$b_{ij} = \begin{cases} 0 & \text{if } P_i \in B_j \\ 1 & \text{if } P_i \notin B_j \end{cases}$$

for $1 \leq i \leq n$ and $1 \leq j \leq m$.

It is noted that following theorem is true and proved in [80].

**Theorem 5.2.1.** If $\alpha_i$ is the $i^{th}$ row of the cumulative array $C_\mathcal{A}$, then $\alpha_1 + \cdots + \alpha_t = \overrightarrow{1}$ if and only if $\{P_1, \ldots, P_t\} \in \mathcal{A}$

cumulative scheme of [107] and [46] uses Shamir’s threshold [190] scheme, where as Blakley’s scheme is used in [200] and [109]. A simple scheme using cumulative array and Karnin-Greene-Hellman threshold scheme [117], proposed by Ghodosi et al [80] is given below.

**The Scheme**

Let $\mathcal{A}_{\min} = \mathcal{A}_1 + \cdots + \mathcal{A}_t$ be a monotone access structure over the set of participants $\mathcal{P} = P_1, \ldots, P_n$. Let $\mathcal{A}_{c_{\max}}^c = B_1 + \cdots + B_m$ be the set of
maximal unauthorized subsets. The share distribution and reconstruction phases are given below.

Share Distribution Phase

1. The dealer $\mathcal{D}$ constructs the $n \times m$ cumulative array $C_A = [b_{ij}]$, where $n$ is the number of participants and $m$ is the cardinality of $A_{\text{max}}^c$.

2. $\mathcal{D}$ used Karnin-Greene-Hellman $(m, m)$ threshold scheme [117] to generate $m$ shares $S_j, 1 \leq j \leq m$.

3. $\mathcal{D}$ gives shares $S_j$ privately to participant $P_i$ if and only if $b_{ij} = 1$.

Secret Reconstruction Phase

1. The secret can be recovered by every set of participants in the access structure using the modular addition over $\mathbb{Z}_q$.

Example 5.2.1. Let $n = 4$ and $A_{\text{min}} = \{\{1, 2\}, \{3, 4\}\}$. In this case, we obtain that $A_{\text{max}}^c = \{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}\}$ and $m = 4$.

The cumulative array for the access structure $\mathcal{A}$ is,

$$C_A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$ 

The Dealer then construct a $(4, 4)$ threshold scheme and generate four shares $s_1, s_2, s_3, s_4$ such that the secret $S = s_1 + s_2 + s_3 + s_4$. The share $s_1$ is then assigned to $P_2$ and $P_4$. The share $s_2$ is then given to $P_2$ and $P_3$. The share $s_3$ is given to $P_1$ and $P_4$. Finally the share $s_4$ is given to $P_1$ and $P_3$. Let $S_1, S_2, S_3$ and $S_4$ be the shares of each participant $P_1, P_2, P_3$.
Chapter 5. POB and Generalized Secret Sharing

and $P_4$ respectively. Then $S_1 = \{s_3, s_4\}$, $S_2 = \{s_1, s_2\}$, $S_3 = \{s_2, s_4\}$ and $S_4 = \{s_1, s_3\}$. it is noted that participant $P_1$ and $P_2$ can together have all the 4 shares to reconstruct the secret $S$. Same is the case for the participant $P_3$ and $P_4$. This shows that only the authorized set of participants can obtain the shared secret $S$. The scheme is also perfect.

5.3 Permutation Ordered Binary (POB) System

The POB system is developed by Sreekumar et al [202] for the efficient distributed storage and retrieval of secret data using secret sharing technique. The share generation and reconstruction involves simple XOR operations. The share generation algorithm is linear and depends on the size of the secret. The shares generated are 1 bit less than the secret, but still provides the same level of security and hence a reduction in storage space can be achieved. The POB system can be used to implement an $(n,n)$ threshold secret sharing scheme very efficiently. In this section we will give a brief introduction about the POB system construction and the $(n,n)$ secret sharing scheme using POB. More details about the POB system can be found in [201].

5.3.1 POB system construction

The POB number system is very special in which all the numbers in the range $0, \ldots, \binom{n}{r} - 1$, are represented by a binary string $B = b_{n-1}b_{n-2}\ldots b_0$, of length $n$ and having exactly $r$ 1s and is represented as $POB(n,r)$, where $n$ and $r$ are positive integers and $n \geq r$.

Each POB number $B$ is associated with a value $V(B)$ and is computed as a sum of the positional values of each bit $b_j$ in the number. The positional values of each bit is computed as
5.3. Permutation Ordered Binary (POB) System

\[ b_j \binom{j}{p_j}, \text{ where, } p_j = \sum_{i=0}^{j} b_i, \]

and the value \( V(B) \) represented by the POB-number \( B \) is,

\[ V(B) = \sum_{j=0}^{n-1} b_j \binom{j}{p_j} \quad (5.1) \]

The POB representation is unique and it can be proved that, since exactly \( \binom{n}{r} \) such binary strings exist, each number will have a distinct representation. Each POB number represented using binary \( POB(n, r) \) will have a unique value \( V(B) \) computed as per equation 5.1. The POB number is denoted by the suffix 'p', in order to distinguish it from a binary number.

**Example 5.3.1.** \( POB(9, 4) = 100101010_p \) be a POB number having 9 bits and 4 ones. Its value \( V(B) \) is computed as follows

\[
V(B) = \binom{8}{4} + \binom{5}{3} + \binom{3}{2} + \binom{1}{1} \\
V(B) = 70 + 10 + 3 + 1 = 84
\]

There exist efficient algorithm to convert POB value into corresponding POB number. The following algorithm will generate the POB number from the given POB value.
Algorithm 5.1: Convert POB value to POB number

| Input: n, r and V(B)-The value of the POB number B |
| Output: The POB(n, r) number \( B = b_{n-1}b_{n-2}...b_0 \) corresponds to the value \( V(B) \) |

1. let \( j = n \) and \( temp = V(B) \)
2. for \( k = r \) down to 1 do
3. repeat
4. \( j = j - 1 \)
5. \( p = \binom{j}{k} \)
   if \( temp \geq p \) then
   \( temp = temp - p \)
   \( b_j = 1 \)
   else
   \( b_j = 0 \)
   end if
6. until \( b_j == 1 \);
7. end

The POB number system developed have great potential for secret sharing. Each POB number is considered as a balanced string which contains same number of ones and zeros. These balanced strings are useful for sharing images where each pixel will have uniform code as share. The POB system can be used to develop an \( (n, n) \) secret sharing scheme very efficiently, which is explored in the next section.
5.3. Permutation Ordered Binary (POB) System

5.3.2 $(n, n)$ secret sharing scheme using POB

It is noted that efficient $(n, n)$ schemes are the building blocks of secret sharing schemes having more generalized monotone access structure. There are several proposals for generalized access structure based secret sharing using $(n, n)$ threshold secret sharing scheme. Karnin et al [117] developed an unanimous consent scheme which is used in the Benaloh’s and Leichter scheme [15]. Ito et al [107] used Shamir’s $(n, n)$ threshold scheme. POB system can be used for developing an efficient $(n, n)$ scheme which is secure and reliable. The scheme is perfect and also the POB numbers are balanced strings. They always contain same number of ones and zeros.

The secret sharing algorithm takes a secret of size 8 bits and expands it to 9 bits by adding an extra bit at random position $r$. Algorithm 5.3 explains this procedure. The secret sharing scheme uses POB$(9,4)$ scheme and produces POB numbers of size 9 bits with 4 ones in it. The value $V(B)$ corresponds to each POB number $B$ is computed and is used as the share value, which is only 7 bits in size. The details of share generation is given in the Algorithm 5.2. The secret reconstruction technique is mentioned in Algorithm 5.4. The input POB values are converted to POB numbers using the Algorithm 5.1. The original secret $K$ can be easily reconstructed using simple XOR operation and also the extra bit at position $r$ can be easily removed. This random location is computed based on one of the POB number selected i.e., $A_2$. The random location $r$ is computed as, $r = \left\lceil \frac{V(A_2)+1}{14} \right\rceil$. The POB value $V(A_2)$ will have value in the range $[1..125]$. So $r$ will take values in the range $[1..9]$. 123
Algorithm 5.2: \((n,n)\) Secret Sharing using POB

**Input:** A single byte string \(K = K_1 K_2 K_3 \ldots K_8\).

**Output:** \(n\) shares \(S_1, S_2, \ldots, S_n\) of length 7 bits each.

1. Choose \(n - 2\), \(POB(9,4)\)-numbers randomly \(A_i, 2 \leq i \leq n - 1\).
2. Let \(r = \left\lceil \frac{V(A_2) + 1}{14} \right\rceil\)

   /* The input string \(K\) is expanded to 9 bits by inserting an extra bit at position \(r\) using expand algorithm. */

3. \(T = \text{expand}(K)\)
4. Let \(W = T \oplus A_2 \oplus A_3 \oplus \ldots \oplus A_{n-1}\)

   /* Compute the bits of \(A_1\) using \(W\) */
5. \(\text{noOfOne} = 0\)
6. for \(i = 1\) to 9 do
   
   if \(W_i == 1\) then
      
      \(\text{noOfOne} = \text{noOfOne} + 1\)
   
   if \(\text{noOfOne}\) is odd then
      
      \(A_1[i] = 1\)
   
   else
      
      \(A_1[i] = 0\)
   
   end if

end if
7. end
8. Randomly assign the remaining null bits of \(A_1\) to 0 or 1

   /* Finally \(A_1\) consists of four 1s and five 0s */
9. \(A_n = W \oplus A_1\)

   /* generate the \(n\) shares */
10. for \(i = 1\) to \(n\) do

11. \(S_i = V(A_i)\).
12. end
5.3. Permutation Ordered Binary (POB) System

**Algorithm 5.3: expand algorithm**

**Input:** binary string of 8 bits \( K \)

**Output:** string of 9 bits \( T \)

1. Compute the binary string \( T = T_1T_2\ldots T_9 \)
2. for \( i = 0 \) to 8 do
   3. \( T_i = \begin{cases} 
   K_i, & \text{if } i < r \\
   K_{i-1}, & \text{if } i > r \\
   0, & \text{if } i = r \text{ and } K \text{ is even parity} \\
   1, & \text{if } i = r \text{ and } K \text{ is odd parity} 
   \end{cases} \)
4. end
5. return \( T \)

**Algorithm 5.4: \((n,n)\) POB secret recovery**

**Input:** \( n \) shares \( S_1, S_2, \ldots, S_n \) of length 7 bits each.

**Output:** The secret \( K = K_1K_2K_3\ldots K_8 \)

1. Let \( A_1, A_2, \ldots, A_n \) be the POB-numbers corresponding to the shares \( S_1, S_2, \ldots, S_n \) respectively.
2. \( r = \lceil \frac{S_2+1}{4} \rceil \)
3. Compute \( T = A_1 \oplus A_2 \oplus A_3 \oplus \ldots \oplus A_n \)
4. Let \( T = T_1T_2\ldots T_9 \)
5. for \( i = 1 \) to 8 do
   6. if \( i \geq r \) then
      7. \( j = i + 1 \)
   8. else
      9. \( j = i \)
   10. end if
   11. \( K_i = T_j \)
6. end
7. The recovered secret is \( K = K_1K_2K_3\ldots K_8 \)
Example 5.3.2. Let us consider a (4, 4) threshold secret sharing scheme. The secret to be shared is $K = 10110110$.

Randomly choose two $POB(9, 4)$ numbers $\{A_2, A_3\}$.

\[ A_2 = 101100010 \quad and \quad A_3 = 010101001 \]

Let the random number $r = \left\lceil \frac{V(A_2) + 1}{14} \right\rceil = \left\lceil \frac{102}{14} \right\rceil = 8$.

The secret $K$ is expanded to 9 bits string $T$ as per the Algorithm 5.3

\[ T = 101101110 \]

\[ W = T \oplus A_2 \oplus A_3 \]

\[ W = 101101110 \oplus 101100010 \oplus 010101001 = 010100101 \]

Now we will compute $A_1$ using the step 6 of the Algorithm 5.2

\[ A_1 = *1 * 0 * *1 * 0 \]

randomly fill rest of the bits so that there will be 4 ones and 5 zeros

\[ A_1 = 110010100 \]

Now we will compute $A_4$

\[ A_4 = W \oplus A_1 \]

\[ A_4 = 010100101 \oplus 110010100 = 100110001 \]
5.4. Proposed Generalized Secret Sharing Scheme

The shares are $V(A_1), V(A_2), V(A_3)$ and $V(A_4)$

\[ S_1 = 113 = 1110001 \]
\[ S_2 = 101 = 1100101 \]
\[ S_3 = 48 = 0110000 \]
\[ S_4 = 86 = 1010110 \]

**Secret Recovery**

The secret can be recovered by using simple XOR operation. From the shares, the POB value can be found which is then converted into POB numbers using the Algorithm 5.1.

Compute

\[ T = A_1 \oplus A_2 \oplus A_3 \oplus A_4 \]

\[ T = 110010100 \oplus 101100010 \oplus 010101001 \oplus 100110001 \]
\[ T = 101101110 \]

Deleting the 8th bit, we get secret as

\[ K = 10110110 \]

### 5.4 Proposed Generalized Secret Sharing Scheme

The proposed scheme make use of $(n,n)$ scheme using POB and cumulative arrays to efficiently share a secret according to a generalized access structure. The detailed algorithm for secret sharing according to the generalized access structure is given in 5.5. The secret reconstruction is mentioned in the Algorithm 5.6.
Algorithm 5.5: Generalized Secret Sharing using POB

**Input:** Access structure corresponds to a secret sharing scheme.

**Output:** Shares for each participants corresponds to the given access structure.

1. Find the maximal unauthorized set $A^c_{max}$ corresponds to the given access structure.
2. The dealer $D$, constructs the $n \times m$ cumulative array $C_A = [b_{ij}]$, where $n$ is the number of participants and $m$ is the cardinality of $A^c_{max}$.
3. $D$ uses $(m, m)$ POB scheme to generate $m$ shares $S_j, 1 \leq j \leq m$.
4. $D$ gives shares $S_j$ privately to participant $P_i$ if and only if $b_{ij} = 1$.

Algorithm 5.6: Secret Reconstruction using POB

**Input:** Shares corresponds to the participants.

**Output:** Shared secret corresponds to the authorized set or error.

1. From the shares generate the POB number.
2. The secret can be reconstructed by XORing the shares corresponds to an authorized set of participant.
3. For an unauthorized set the algorithm gives an error else the shared secret $K$ is returned.

One of the issue with general access structure based secret sharing scheme is the number of shares each participant has to maintain. The storage become a major constraint here. For every eight bytes of secret one byte of storage requirement is saved by using the POB number system based threshold secret sharing. The secret reconstruction needs only simple XOR operation. The operations performed in the POB, that is bit expansion, conversion of POB number to POB value and vice versa
can be performed in time proportional to number of bytes in the secret
values. This makes the scheme more suitable for cumulative array based
generalized secret sharing compared with Benaloh’s and Leichter scheme
[15] and Ito et al [107] scheme.

5.5 Concluding Remarks

In this chapter we have considered a secret sharing scheme realizing the
general access structure. The share size is a major concern in the design
of generalized secret sharing scheme. The share size grows exponentially in
many cases. We have proposed a scheme with cumulative arrays and a \((n, n)\)
threshold scheme using POB. The POB system has a great potential for
secret sharing. The representation is unique and also efficient. In the POB
based threshold scheme the share size is small and also the secret generation
and reconstruction can be easily done by simple XOR operation. An 8 bit
secret can be shared with a share of 7 bit size. The scheme is not ideal
but the probability of guessing the share reduces as the size of the secret
to be shared increases. For sharing a key of size 64 bits only 56 bits are
used. Combining this with the cumulative array scheme is a good choice for
secret sharing based on generalized access structure. There is a possibility
of 126 shares corresponds to a single byte secret. Thus the probability of
guessing the share is 1/126. As the size of the secret grows, this probability
reduces. For a \(k\) byte secret the probability reduces to \((1/126)^k\).