On Efficiently Transferring the Linear Secret-Sharing Scheme Matrix in Ciphertext-Policy Attribute-Based Encryption

Zhen Liu and Zhenfu Cao [⋆]

Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
`liuzhen@sjtu.edu.cn, zfcao@cs.sjtu.edu.cn`

Abstract. Ciphertext-Policy Attribute-Based Encryption(CP-ABE) is a system for realizing complex access control on encrypted data, in which attributes are used to describe a user's credentials and a party encrypting data determines a policy over attributes for who can decrypt. In CP-ABE schemes, access policy is attached to the ciphertext to be the input of the decryption algorithm.

An access policy can be expressed in terms of monotone boolean formula or monotone access structure, and can be realized by a linear secret-sharing scheme(LSSS). In recent provably secure and efficient CP-ABE schemes, the LSSS induced from monotone span program(MSP) is used, where the LSSS is a matrix whose rows are labeled by attributes. And a general algorithm for converting a boolean formula into corresponding LSSS matrix is described recently. However, when there are threshold gates in the access structure, the number of rows of the LSSS matrix generated by the algorithm will be unnecessary large, and consequently the ciphertext size is unnecessary large.

In this paper, we give a more general and efficient algorithm that the number of rows of the LSSS matrix is as small as possible. And by some tricks, the boolean formula acts as the label function, so that only the boolean formula needs to be attached to the ciphertext, which decreases the communication cost drastically.

Keywords: attribute based encryption, ciphertext policy, access structure, linear secret sharing scheme, monotone span program

1 Introduction

Attribute-Based Encryption(ABE), introduced by Sahai and Waters [8], provides a new way for access control of encrypted data. Goyal, Pandey, Sahai, and Waters [3] clarified the concept of ABE into Key-Policy ABE

⋆ Corresponding author.

(KP-ABE) and Ciphertext-Policy (CP-ABE). In KP-ABE, attributes are used to annotate the ciphertexts and formula over attributes are ascribed to user's secret keys. CP-ABE is complementary in that attributes are used to describe user's credentials and the formulas over these credentials are attached to ciphertext by encrypting party. As the CP-ABE is conceptually closer to the traditional access control methods, since the first CP-ABE scheme is presented by Bethencourt, Sahai, and Waters [2], CP-ABE is regarded as a promising concept for next-generation access control. And recently several elegant CP-ABE schemes, which achieve satisfying security, efficiency, access policy expressibility and other virtues such as multi-authority, have been presented in [10, 6, 5].

Access policy can be described in terms of monotone boolean formula or monotone access structure, and can be realized by linear secret-sharing schems (LSSS). In these schemes in [10, 6, 5], the access policy attached to the ciphertext is realized by an LSSS induced from monotone span program(MSP) [4], i.e., a labeled matrix $\widehat{M} = (M, \rho)$ over $\mathbb{Z}_p$ where p is a large prime[1]. While p is related with the security parameter and must be very large, efficiently transferring the LSSS matrix to the decryptor is an important task for using these schemes in practice. In addition, as the ciphertext size is linear in the size of the LSSS matrix (the number of rows), it is desired to construct LSSS matrices with smaller size.

Our contribution In this paper, we present a general and efficient algorithm that converts a boolean formula into corresponding LSSS matrix. Taking the boolean formula as an access tree where interior nodes are threshold gates and leaf nodes correspond to attributes, our algorithm generates the corresponding LSSS matrix whose size is as small as possible. And the i^{th} row of the LSSS matrix is labeled by the i^{th} attribute in the boolean formula. Embedding the algorithm into the Encrypt and Decrypt algorithm, only the boolean formula needs to be attached to the ciphetext. The total communication cost is drastically decreased.

Related work In [5] a general algorithm for converting boolean formulas into LSSS matrices is described. The algorithm considers the formula as an access tree, where interior nodes are $AND(\wedge)$ or $OR(\vee)$ gates and leaf nodes correspond to attributes. The number of rows in the corresponding LSSS matrix is same with the number of leaf nodes in the access tree.

[1] $\mathbb{Z}_p$ is used in [10] and $\mathbb{Z}_N$ is used in [6, 5] where N is a product of three large primes. The use of composite N is critical to the schemes in [6, 5]. But for the access policy, $\mathbb{Z}_N$ plays the same role as $\mathbb{Z}_p$. For simplicity, we state only $\mathbb{Z}_p$.

However, when there are threshold gates in the access structure, the number of rows of the LSSS matrix will be unnecessary large. For example, for a $(3, 2)$ threshold access structure $(A \wedge B) \vee (B \wedge C) \vee (A \wedge C)$, the matrix generated by the algorithm has 5 rows at least. In fact, there exists an LSSS matrix of 3 rows for the access structure.

Organization The remainder of our paper is structured as follows. In Section 2 we first give background on access structure and the recent CP-ABE schemes, then we review the definitions and some previous work on the monotone span program. We present our algorithm and give some examples in Section 3. Efficiency improvements are discussed in Section 4, and finally we conclude in Section 5.

2 Background

We first give formal definitions for access structures and relevant background on Linear Secret Sharing Schemes(LSSS).

2.1 Access Structures

Definition 1 (Access Structure [1]). *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

In these recent CP-ABE schemes [10, 6, 5], the attributes will play the role of parties and only the monotone access structures are dealt with. From now on, unless stated otherwise, by an access structure we mean a monotone access structure.

The recent CP-ABE schemes [10, 6, 5] employ linear secret-sharing schemes and use the definition adapted from [1].

Definition 2 (Linear Secret-Sharing Schemes(LSSS)[1, 10]). *A secret scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix A called the share-generating matrix for Π. The Matrix A has l rows and n columns. For all $i = 1, \ldots, l$, the i^{th} row of A is labeled by a party $\rho(i)$ (ρ is a function from $\{i, \ldots, l\}$ to $\mathcal{P}$).*

When we consider the column vector $\boldsymbol{v} = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $A\boldsymbol{v}$ is the vector of l shares of the secret s according to Π. The share $(A\boldsymbol{v})_i$ belongs to party $\rho(i)$.

It is shown in [1] that every linear secret sharing-scheme according to the above definition also enjoys the linear reconstruction property: Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be an authorized set, and define $I \subset \{1, 2, \ldots, l\}$ as $I = \{i : \rho(i) \in S\}$. There exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that for any valid shares $\{\lambda_i\}$ of a secret s according to Π, $\sum_{i \in I} \omega_i \lambda_i = s$. And these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix A. For the security property of LSSS, no such constants $\{\omega_i\}$ exist for unauthorized sets.

Boolean formulas Access policy can also be described in terms of monotone boolean formulas. Every monotone access structure or monotone boolean formula can be realized by a linear secret-sharing scheme. The LSSS in definition 2 is a scheme induced from monotone span program (MSP) [4]. Naturally, boolean formulas are used to describe the access policy, and equivalent LSSS are used to encrypt the message and decrypt the ciphertext.

2.2 Access Structure in CP-ABE Schemes

[10, 6] are on CP-ABE and [5] is on Multi-Authority CP-ABE. The definitions and constructions in these two kinds of schemes are different, but the uses of the access structure in the Encrypt and Decrypt algorithms are same. We use the definition and construction in [6] as the example.

A CP-ABE scheme consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt.

Setup $(\lambda, U) \rightarrow (PK, MSK)$ The setup algorithm takes in the security parameter λ and the attribute universe description U. It outputs the public parameters PK and a master key MSK.

Encrypt $(PK, M, \mathbb{A}) \rightarrow CT$ The encryption algorithm takes in the public parameters PK, the message M, and an access structure $\mathbb{A}$ over the universe of attributes. It will output a ciphertext CT such that only users whose attributes satisfy the access structure $\mathbb{A}$ should be able to decrypt the message. *It is assumed that $\mathbb{A}$ is implicitly included in CT.*

KeyGen $(MSK, PK, S) \to SK$ The key generation algorithm takes in the master secret key MSK, the public parameters PK, and a set of attributes S. It outputs a private key SK.

Decrypt $(PK, CT, SK) \to M$ The decryption algorithm takes in the public parameters PK, *a ciphertext* CT*, which contains an access policy* $\mathbb{A}$, and a private key SK. If the set S of attributes of the private key satisfies the access structure $\mathbb{A}$, it outputs the message M.

Observations on the Access Policy in the CP-ABE Schemes Observing the constructions in [10, 6, 5], we draw the following conclusions.

1. In these schemes, LSSS is used to realize the access policy. And the LSSS matrix (A, ρ) is attached to the ciphertext.
2. (A, ρ) is a labeled matrix over $\mathbb{Z}_p$. It is very large and increases the communication cost drastically.
3. The size of the ciphertext is linear in the size of A (the number of rows of A).
4. In order to use these schemes in practice, we should design a general algorithm for converting any boolean formula into corresponding LSSS matrix.
5. For a boolean formula, the size of the corresponding LSSS matrix should be as small as possible.

2.3 Monotone Span Program and LSSS

The label matrix (A, ρ) in definition 2 is also called a monotone span program [4]. Karchmer and Widgerson [4] introduced the model of monotone span program, and proved that if there is a monotone span program for some boolean function then there exists a linear secret sharing scheme for the corresponding access structure in which the sum of the sizes of the shares of the parties is the number of rows in the span program, i.e., the size of the span program. We give the formal definitions and conclusions as follows.

Definition 3 (Monotone Span Program(MSP)[4, 7]). *A Monotone Span Program (MSP)* $\mathcal{M}$ *is a quadruple* $(\mathbb{F}, M, \varepsilon, \rho)$*, where* $\mathbb{F}$ *is a field,* M *is a matrix (with* m *rows and* $d \leq m$ *columns) over* $\mathbb{F}$*,* $\rho : \{1, \ldots, m\} \to \{1, \ldots, n\}$ *is a surjective function and the row vector* $\varepsilon = (1, 0, \ldots, 0) \in \mathbb{F}^d$ *is called target vector. The size of* $\mathcal{M}$ *is the number* m *of rows and is denoted as size*$(\mathcal{M})$*.*

As ρ labels each row i of M to a player $P_{\rho(i)}$, each player can be regarded as the "owner" of one or more rows. For any set of players $G \subseteq \mathcal{P}$, the sub-matrix consisting of rows owned by players in G is denoted by M_G.

The span of a matrix M, denoted $span(M)$ is the subspace generated by the rows of M, i.e., all vectors of the form sM.

An MSP is said to compute an access structure $\mathcal{A}$ if

$$G \in \mathcal{A} \quad \text{if and only if} \quad \varepsilon \in span(M_G).$$

Theorem 1 (LSSS induced from MSP[4, 1]). *Assume that there exists a monotone span program $\mathcal{M} = (\mathbb{F}, M, \varepsilon, \rho)$, of size m, computing the access structure $\mathcal{A}$. Then there is a linear secret sharing scheme over $\mathbb{F}$ realizing the access structure in which the total size of the shares is m.*

Using the labeled matrix (M, ρ) in definition 3 as the share-generating matrix in definition 2, we get the LSSS Π. The LSSS Π is said to be induced from MSP $\mathcal{M}$. The complete proof is presented in [4], and we only show the reconstruction property of the LSSS here.

For any secret $s \in \mathbb{Z}_p, \boldsymbol{v} = (s, r_2, \dots, r_d)$ is a random vector in $\mathbb{Z}_p^d$. For an authorized set $G \in \mathcal{A}$, let $I = \{i : P_{\rho(i)} \in G\}$ and M_i denote the i^{th} row of M, the shares $\{\lambda_i = (M\boldsymbol{v})_i = M_i \cdot \boldsymbol{v} | i \in I\}$ are held by G, and G can compute the reconstruction constants $\{\omega_i : i \in I\}$ such that $\sum_{i\in I} \omega_i M_i = \varepsilon$. Then G can compute

$$\sum_{i\in I} \omega_i \lambda_i = \sum_{i\in I} \omega_i (M_i \cdot \boldsymbol{v}) = \varepsilon \cdot \boldsymbol{v} = s$$

2.4 New Monotone Span Programs from Old

Definition 4 (Insertion of MSP [7]). *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two monotone access structures defined on participant sets $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively, and let $P_z \in \mathcal{P}_1$. Define the insertion of $\mathcal{A}_2$ at player P_z in $\mathcal{A}_1$, $\mathcal{A}_1(P_z \to \mathcal{A}_2)$, to be the monotone access structure defined on the set $(\mathcal{P}_1 \setminus \{P_z\}) \cup \mathcal{P}_2$ such that for $G \subseteq (\mathcal{P}_1 \setminus \{P_z\}) \cup \mathcal{P}_2$ we have*

$$G \in \mathcal{A}_1(P_z \to \mathcal{A}_2) \iff \begin{cases} (G \cap \mathcal{P}_1 \in \mathcal{A}_1), \quad or \\ ((G \cap \mathcal{P}_1) \cup \{P_z\} \in \mathcal{A}_1 \text{ and } G \cap \mathcal{P}_2 \in \mathcal{A}_2). \end{cases}$$

In other words, $\mathcal{A}_1(P_z \to \mathcal{A}_2)$ is the monotone access structure $\mathcal{A}_1$ with participant P_z "replaced" by the sets of $\mathcal{A}_2$.

Theorem 2. *[7] Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be monotone access structures defined on the set of participants $\mathcal{P}_1$ and $\mathcal{P}_2$ and with MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively, and let $P_z \in \mathcal{P}_1$. Let the size of $\mathcal{M}_1$ be m_1 and the size of $\mathcal{M}_2$ be m_2. Then there exists an MSP $\mathcal{M}$ computing the access structure $\mathcal{A}_1(P_z \to \mathcal{A}_2)$ of size equal to $m_1 + (m_2 - 1)q$, where q is the number of rows owned by P_z in $\mathcal{M}_1$.*

Nikov and Nikova [7] give the construction of the MSP $\mathcal{M}$ and prove that $\mathcal{M}$ computes $\mathcal{A}_1(P_z \to \mathcal{A}_2)$. We only present the construction here. Let $m_1 \times d_1$ matrix $M^{(1)} = \begin{pmatrix} M^{(1)}_{P_z} \\ \overline{M}^{(1)} \end{pmatrix}$ and $m_2 \times d_2$ matrix $M^{(2)} = \begin{pmatrix} \boldsymbol{u}^{(2)} & \widetilde{M}^{(2)} \end{pmatrix}$ be the corresponding matrices of MSPs $\mathcal{M}_1$ and $\mathcal{M}_2$, where $M^{(1)}_{P_z}$ is the q rows owned by P_z and $\boldsymbol{u}^{(2)}$ is the first column, assuming that the rows of P_z are the first rows in $M^{(1)}$. Let $M^{(1)}_{P_z} = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_q \end{pmatrix}$, $\boldsymbol{u}^{(2)} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{m_2} \end{pmatrix}$, where $\boldsymbol{v}_i (i = 1, \ldots, q)$ are row vectors in $\mathbb{Z}_p^{d_1}$ and $u_j \in \mathbb{Z}_p (j = 1, \ldots, m_2)$. For $i = 1$ to q define $\boldsymbol{v}_i \otimes \boldsymbol{u}^{(2)} = \begin{pmatrix} u_1 \boldsymbol{v}_i \\ u_2 \boldsymbol{v}_i \\ \vdots \\ u_{m_2} \boldsymbol{v}_i \end{pmatrix}$, note that $\boldsymbol{v}_i \otimes \boldsymbol{u}^{(2)}$ be a $m_2 \times d_1$ matrix. Let $M = \begin{pmatrix} \boldsymbol{v}_1 \otimes \boldsymbol{u}^{(2)} & \widetilde{M}^{(2)} & 0 & 0 & 0 \\ \boldsymbol{v}_2 \otimes \boldsymbol{u}^{(2)} & 0 & \widetilde{M}^{(2)} & 0 & 0 \\ \vdots & 0 & 0 & \ddots & 0 \\ \boldsymbol{v}_q \otimes \boldsymbol{u}^{(2)} & 0 & 0 & 0 & \widetilde{M}^{(2)} \\ \overline{M}^{(1)} & 0 & 0 & 0 & 0 \end{pmatrix}$. The rows of $\begin{pmatrix} \overline{M}^{(1)} & 0 \end{pmatrix}$ are labeled as the rows of $\overline{M}^{(1)}$ in $\mathcal{M}_1$, and the rows of $\begin{pmatrix} \boldsymbol{v}_i \otimes \boldsymbol{u}^{(2)} & 0 & \widetilde{M}^{(2)} & 0 \end{pmatrix}$ are labeled as the corresponding rows in $\mathcal{M}_2$. Then M computes $\mathcal{A}_1(P_z \to \mathcal{A}_2)$, and it is a $(m_1 + (m_2 - 1)q) \times (d_1 + (d_2 - 1)q)$ matrix.

3 Our Technique to Transfer the LSSS Matrix

Intuition

1. Design a general and efficient algorithm for converting boolean formulas into corresponding LSSS matrices, and embed the converting algorithm into the Encrypt and Decrypt algorithm.
2. The converting algorithm takes as input a boolean formula, and outputs a labeled LSSS matrix (A, ρ).
3. Use the boolean formula as the label function (for example, the i^{th} row is labeled by the i^{th} attribute in the boolean formula).
4. The size of the LSSS matrix should be as small as possible, and using the threshold MSP as the underlying MSP is an attractive idea.

With the above ideas, the boolean formula rather than the LSSS matrix (A, ρ) needs to be attached to the ciphertext, and the communication cost is decreased drastically. This is at the cost of running the converting algorithm in Decrypt algorithm. And we will show that the running cost is small.

The Expression of the Boolean Formula We consider the boolean formula as an access tree, where interior nodes are threshold gates and the leaf nodes correspond to attributes. The access tree can be expressed by a formatted formula.

A boolean formula F can be expressed as $(F_1, F_2, \ldots, F_n, t)$. The root node of the tree is a (n, t)-gate, and its children are $F_1, F_2, \ldots, F_n$, where F_i is a leaf node corresponding to an attribute or is a threshold gate node with similar form. Note that the value n is implicitly decided by the number of the children, and only the threshold value t is explicit in the formula.

For example,

$$\begin{aligned}
&((A \wedge B) \vee (B \wedge C) \vee (A \wedge C)) \wedge (A \vee D) \wedge E \\
=&((A, B, C, 2), (A, D, 1), E, 3) \\
=&(F_1, F_2, F_3, t)
\end{aligned}$$

where $F_1 = (A, B, C, 2)$, $F_2 = (A, D, 1)$, $F_3 = E$, and $t = 3$.

It is easy to format any boolean formula to this form, and for simplicity we will use "Formatted Boolean Formula" of this form as the input of our converting algorithm.

In addition, when we say "the i^{th} attribute of the formatted formula", we mean the index of the attribute in the formatted formula from left to right. For example, in the formatted formula $((A, B, C, 2), (A, D, 1), E, 3)$, A, B, C, A, D are the $1^{th}, 2^{th}, 3^{th}, 4^{th}, 5^{th}$ attributes respectively.

MSP for Threshold Access Structure Inspired by [4], for a (n, t) threshold access structure, we can construct a special (n, t)-MSP over field $\mathbb{Z}_p (p > n + 1)$ as $M = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 2^2 & \dots & 2^{t-1} \\ 1 & 3 & 3^2 & \dots & 3^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & n & n^2 & \dots & n^{t-1} \end{pmatrix}$. It is easy to see that for any subset $S \subseteq \{1, 2, \dots, n\}$, $(1, 0, \dots, 0) \in span(M_S)$ iff $|S| \geq t$.

Actually, it is based on the same idea behind Shamir's sceret sharing scheme for threshold functions [9].

Note that this (n, t)-MSP is completely decided by the values of n and t, and we will use this special (n, t)-MSP for threshold gates in our converting algorithm.

Modified Construction of Theorem 2 In theorem 2, it is assumed that all the rows of P_z are first rows in $M^{(1)}$. To use the construction in our algorithm and make the formatted boolean formula act as the label function, we remove the assumption. We "replace" one row each time as follows.

Without loss of generality, we let $q = 2$ and $M^{(1)} = \begin{pmatrix} \overline{M}^{(11)} \\ \boldsymbol{v}_1 \\ \overline{M}^{(12)} \\ \boldsymbol{v}_2 \\ \overline{M}^{(13)} \end{pmatrix}$. We find the first row labeled by P_z, the row $\boldsymbol{v}_1$, "replace" $\boldsymbol{v}_1$ with $M^{(2)}$ and get the new $M^{(1)} = \begin{pmatrix} \overline{M}^{(11)} & 0 \\ \boldsymbol{v}_1 \otimes \boldsymbol{u}^{(2)} & \widetilde{M}^{(2)} \\ \overline{M}^{(12)} & 0 \\ \boldsymbol{v}_2 & \mathbf{0} \\ \overline{M}^{(13)} & 0 \end{pmatrix}$. Repeatedly, we find the first row labeled by P_z in the new $M^{(1)}$, the row $\begin{pmatrix} \boldsymbol{v}_2 \, \mathbf{0} \end{pmatrix}$, "replace" this row with $M^{(2)}$ and get the new $M^{(1)} = \begin{pmatrix} \overline{M}^{(11)} & 0 & 0 \\ \boldsymbol{v}_1 \otimes \boldsymbol{u}^{(2)} & \widetilde{M}^{(2)} & 0 \\ \overline{M}^{(12)} & 0 & 0 \\ \boldsymbol{v}_2 \otimes \boldsymbol{u}^{(2)} & \mathbf{0} \otimes \boldsymbol{u}^{(2)} & \widetilde{M}^{(2)} \\ \overline{M}^{(13)} & 0 & 0 \end{pmatrix} =$

$\begin{pmatrix} \overline{M}^{(11)} & 0 & 0 \\ \boldsymbol{v}_1 \otimes \boldsymbol{u}^{(2)} & \widetilde{M}^{(2)} & 0 \\ \overline{M}^{(12)} & 0 & 0 \\ \boldsymbol{v}_2 \otimes \boldsymbol{u}^{(2)} & 0 & \widetilde{M}^{(2)} \\ \overline{M}^{(13)} & 0 & 0 \end{pmatrix}$. This is an equivalent MSP matrix that computes the access structure $\mathcal{A}_1(P_z \to \mathcal{A}_2)$.

3.1 Our algorithm

Taking formatted boolean formula as input, repeatedly using the special (n,t)-MSP and the above modified construction, we describe our algorithm and give the detail pseudocode.

The Description of Our Algorithm

Input : A formatted boolean formula F.
Output : An LSSS matrix M that realizes the access structure of the boolean formula. The i^{th} row of M is labeled by the i^{th} attribute in the formatted formula.
Convert(F) : In the following, M is a $m \times d$ matrix over $\mathbb{Z}_p$, and L is a vector with m coordinates, where each coordinate is an attribute or a formatted boolean formula. The i^{th} coordinate of L labels the i^{th} row of M.

```
1. Let matrix $M = (1)$, vector $L = (F)$ , and let $m = 1, d = 1$.
2. Repeat until all coordinates of $L$ are attributes
   (a) $M$ is a $m \times d$ matrix over $\mathbb{Z}_p$, and $L = (L_1, L_2, \ldots, L_m)$.
   (b) Scan the coordinates of $L$ to find the first coordinate that is
       a formatted formula rather than an attribute. Assume the index is $z$ and
       $L_z = F_z = (F_{z1}, F_{z2}, \ldots, F_{zm_2}, t_2)$ is a formatted boolean formula.
   (c) Resolve $F_z$ to get its $m_2$ children $F_{z1}, F_{z2}, \ldots, F_{zm_2}$ and its
       threshold value $t_2$.
   (d) Execute "Insertion" of the special $(m_2, t_2)$-MSP matrix on the
       $z^{th}$ row of $M$, getting the new $M$ with $m - 1 + m_2$ rows and
       $d - 1 + t_2$ columns.
       Let $L = (L_1, L_2, \ldots, L_{z-1}, F_{z1}, F_{z2}, \ldots, F_{zm_2}, L_{z+1}, \ldots, L_m)$.
       Let $m = m - 1 + m_2, d = d - 1 + t_2$.
3. Return the matrix $M$
```

Note that *"2.(c) Resolve F_z to get its m_2 children $F_{z1}, F_{z2}, \ldots, F_{zm_2}$ and its threshold value t_2"* is easy and we do not discuss the detail here.

Algorithm 1.1 Convert Boolean Formula F to LSSS Matrix M

```
Require: $F$ is a formatted boolean formula.
Ensure: $M$ is the LSSS matrix of $F$, and the $i^{th}$ row of $M$ is labeled by the $i^{th}$ attribute in $F$.
$(M[1,1], L[1], m, d) \leftarrow (1, F, 1, 1)$
$z \leftarrow 1$
while $z \neq 0$ do
    $z \leftarrow 0$
    $i \leftarrow 1$
    while $i \leq m$ $AND$ $z = 0$ do
        if L[i] is a Formula then
            $z \leftarrow i$
        end if
        $i \leftarrow i + 1$
    end while
    if $z \neq 0$ then
        $F_z \leftarrow L[z]$
        $m_2 \leftarrow$ the number of children of $F_z$
        $L_2[i] \leftarrow$ the $i^{th}$ children of $F_z$ $(i = 1, 2, \ldots, m_2)$
        $d_2 \leftarrow$ the threshold value of $F_z$
        $(M_1, L_1, m_1, d_1) \leftarrow (M, L, m, d)$
        for $i = 1$ to $z - 1$ step 1 do
            $L[i] \leftarrow L_1[i]$
            for $j = 1$ to $d_1$ step 1 do
                $M[i,j] \leftarrow M_1[i,j]$
            end for
            for $j = d_1 + 1$ to $d_1 + d_2 - 1$ step 1 do
                $M[i,j] \leftarrow 0$
            end for
        end for
        for $i = z$ to $z + m_2 - 1$ step 1 do
            $L[i] \leftarrow L_2[i - z + 1]$
            for $j = 1$ to $d_1$ step 1 do
                $M[i,j] \leftarrow M_1[z,j]$
            end for
            $a \leftarrow i - (z - 1)$
            $x \leftarrow a$
            for $j = d_1 + 1$ to $d_1 + d_2 - 1$ step 1 do
                $M[i,j] \leftarrow x$
                $x \leftarrow x * a \mod p$
            end for
        end for
        for $i = z + m_2$ to $m_1 + m_2 - 1$ step 1 do
            $L[i] \leftarrow L_1[i - m_2 + 1]$
            for $j = 1$ to $d_1$ step 1 do
                $M[i,j] \leftarrow M_1[i - m_2 + 1, j]$
            end for
            for $j = d_1 + 1$ to $d_1 + d_2 - 1$ step 1 do
                $M[i,j] \leftarrow 0$
            end for
        end for
        $(m, d) \leftarrow (m_1 + m_2 - 1, d_1 + d_2 - 1)$
    end if
end while
```

3.2 Examples of Our algorithm

We show some examples here. For simplicity, we use the filed $\mathbb{Z}_p = \mathbb{Z}_{47}$.

Example 1. The access structure is

$$((A \wedge B) \vee (B \wedge C) \vee (C, D, E, 2)) \wedge (E, F, G, H, 3).$$

Format the boolean formula as

$$\begin{aligned} &((A \wedge B) \vee (B \wedge C) \vee (C, D, E, 2)) \wedge (E, F, G, H, 3) \\ =&((B \wedge (A \vee C)) \vee (C, D, E, 2)) \wedge (E, F, G, H, 3) \\ =&(((B, (A, C, 1), 2), (C, D, E, 2), 1), (E, F, G, H, 3), 2) \end{aligned}$$

Taking the formula $(((B, (A, C, 1), 2), (C, D, E, 2), 1), (E, F, G, H, 3), 2)$ as input, the algorithm works as follows.

1. $M = \begin{pmatrix} 1 \end{pmatrix}$, $L = \begin{pmatrix} (((B, (A, C, 1), 2), (C, D, E, 2), 1), (E, F, G, H, 3), 2) \end{pmatrix}$
2. $M = \begin{pmatrix} 1\,1 \\ 1\,2 \end{pmatrix}$, $L = \begin{pmatrix} ((B, (A, C, 1), 2), (C, D, E, 2), 1) \\ (E, F, G, H, 3) \end{pmatrix}$
3. $M = \begin{pmatrix} 1\,1 \\ 1\,1 \\ 1\,2 \end{pmatrix}$, $L = \begin{pmatrix} (B, (A, C, 1), 2) \\ (C, D, E, 2) \\ (E, F, G, H, 3) \end{pmatrix}$
4. $M = \begin{pmatrix} 1\,1\,1 \\ 1\,1\,2 \\ 1\,1\,0 \\ 1\,2\,0 \end{pmatrix}$, $L = \begin{pmatrix} B \\ (A, C, 1) \\ (C, D, E, 2) \\ (E, F, G, H, 3) \end{pmatrix}$
5. $M = \begin{pmatrix} 1\,1\,1 \\ 1\,1\,2 \\ 1\,1\,2 \\ 1\,1\,0 \\ 1\,2\,0 \end{pmatrix}$, $L = \begin{pmatrix} B \\ A \\ C \\ (C, D, E, 2) \\ (E, F, G, H, 3) \end{pmatrix}$
6. $M = \begin{pmatrix} 1\,1\,1\,0 \\ 1\,1\,2\,0 \\ 1\,1\,2\,0 \\ 1\,1\,0\,1 \\ 1\,1\,0\,2 \\ 1\,1\,0\,3 \\ 1\,2\,0\,0 \end{pmatrix}$, $L = \begin{pmatrix} B \\ A \\ C \\ C \\ D \\ E \\ (E, F, G, H, 3) \end{pmatrix}$

7. $M = \begin{pmatrix} 1&1&1&0&0&0 \\ 1&1&2&0&0&0 \\ 1&1&2&0&0&0 \\ 1&1&0&1&0&0 \\ 1&1&0&2&0&0 \\ 1&1&0&3&0&0 \\ 1&2&0&0&1&1 \\ 1&2&0&0&2&4 \\ 1&2&0&0&3&9 \\ 1&2&0&0&4&16 \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \\ \boldsymbol{v}_4 \\ \boldsymbol{v}_5 \\ \boldsymbol{v}_6 \\ \boldsymbol{v}_7 \\ \boldsymbol{v}_8 \\ \boldsymbol{v}_9 \\ \boldsymbol{v}_{10} \end{pmatrix}, L = \begin{pmatrix} B \\ A \\ C \\ C \\ D \\ E \\ E \\ F \\ G \\ H \end{pmatrix}$

$S_1 = \{A, B, F, G, H\}$ is an authorized set, and the rows owned by S_1 are $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_8, \boldsymbol{v}_9, \boldsymbol{v}_{10}\}$, the reconstruction constants are $\{\omega_1 \equiv 4, \omega_2 \equiv -2, \omega_8 \equiv -6, \omega_9 \equiv 8, \omega_{10} \equiv -3\} \mod 47$.
$S_2 = \{D, E, F, G\}$ is an authorized set, and the rows owned by S_2 are $\{\boldsymbol{v}_5, \boldsymbol{v}_6, \boldsymbol{v}_7, \boldsymbol{v}_8, \boldsymbol{v}_9\}$, the reconstruction constants are $\{\omega_5 \equiv 6, \omega_6 \equiv -4, \omega_7 \equiv -3, \omega_8 \equiv 3, \omega_9 \equiv -1\} \mod 47$.

If there are only AND and OR gates in the access structure, the size of the LSSS matrix generated by our algorithm is same with that of [5]. If there are threshold gates in the access structure, our algorithm generates matrix of smaller size than that of [5].

Example 2. The access structure is $A \wedge (D \vee (B \wedge C))$.

The matrix generated by algorithm [5] is $\begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix}$ labeled by $\begin{pmatrix} A \\ D \\ B \\ C \end{pmatrix}$

The matrix generated by our algorithm is $\begin{pmatrix} 1&1&0 \\ 1&2&0 \\ 1&2&1 \\ 1&2&2 \end{pmatrix}$ labeled by $\begin{pmatrix} A \\ D \\ B \\ C \end{pmatrix}$.

Example 3. The access structure is

$$(A \wedge B) \vee (B \wedge C) \vee (A \wedge C) = (B \wedge (A \vee C)) \vee (A \wedge C) = (A, B, C, 2).$$

Taking $(B \wedge (A \vee C)) \vee (A \wedge C)$ as input, the algorithm [5] generates the matrix $\begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & -1 \end{pmatrix}$ labeled by $\begin{pmatrix} B \\ A \\ C \\ A \\ C \end{pmatrix}$ with size $= 5$.

Taking the formatted formula $(A, B, C, 2)$ as input, our algorithm generates the matrix $\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}$ labeled by $\begin{pmatrix} A \\ B \\ C \end{pmatrix}$ with size = 3.

3.3 Using Our Algorithm in the CP-ABE schemes

Our algorithm can be used in the recent CP-ABE schemes[10, 6, 5] by

1. Embedding our converting algorithm in the Encrypt and Decrypt algorithm.
2. Attaching the formatted boolean formula to the ciphertext.

In this way, at the cost of running the converting algorithm in the Decrypt algorithm, we attach the formatted boolean formula, instead of the LSSS matrix (A, ρ), to the ciphertext. Note that (A, ρ) over $\mathbb{Z}_p$ is very large, we decrease the communication cost drastically.

Observing **Algorithm 1.1**, we know that the main computation cost of the algorithm is on "$x \leftarrow x * a \mod p$", i.e., computing

$$a^j \mod p \quad (a = 1, 2, \ldots, n; j = 1, 2, \ldots, t - 1)$$

for each (n, t) threshold gate. While $1 \leq t \leq n \ll p$ and most values can be used multiple times, the computation cost of the algorithm is small.

4 Efficiency Improvement of the Converting Algorithm in Decryption

While the LSSS matrix must be completely generated in the Encrypt algorithm, we can only generate the rows labeled by the attributes that the decryptor possesses.

4.1 Modified Converting Algorithm

Input : A formatted boolean formula F, a set S of attributes.
Output : An LSSS sub-matrix M_S. The i^{th} row of M_S is labeled by the i^{th} attribute in the formatted formula F and possessed by S.
Convert(F) : In the following, M is a $m \times d$ matrix over $\mathbb{Z}_p$, and L is a vector with m coordinates, where each coordinate is an attribute or a formatted boolean formula. The i^{th} coordinate of L labels the i^{th} row of M.

1. Let matrix $M = (1)$, vector $L = (F)$, and let $m = 1, d = 1$.

2. Repeat until all coordinates of L are attributes in S
   (a) M is a $m \times d$ matrix over $\mathbb{Z}_p$, and $L = (L_1, L_2, \ldots, L_m)$.
   (b) Scan the coordinates of L to find the first coordinate that is a formatted formula or an attribute not in S. Assume the index is z and $L_z = F_z = (F_{z1}, F_{z2}, \ldots, F_{zm_2}, t_2)$ is a formatted boolean formula or $L_z = att_z$ is an attribute not in S. Let S_{F_z} denote the attributes set on the leaf nodes of F_z.
   (c) If $(L_z = att_z \notin S)$ OR $((L_z = F_z)\ AND\ (S_{F_z} \cap S = \emptyset))$,
      i. Remove the z^{th} of M
      ii. Let $L = (L_1, L_2, \ldots, L_{z-1}, L_{z+1}, \ldots, L_m)$
      iii. Let $m = m - 1$.

      Otherwise,
      i. Resolve F_z to get its m_2 children $F_{z1}, F_{z2}, \ldots, F_{zm_2}$ and its threshold value t_2.
      ii. Execute "Insertion" of the special (m_2, t_2)-MSP matrix on the z^{th} row of M, getting the new M with $m-1+m_2$ rows and $d - 1 + t_2$ columns.
      iii. Let $L = (L_1, L_2, \ldots, L_{z-1}, F_{z1}, F_{z2}, \ldots, F_{zm_2}, L_{z+1}, \ldots, L_m)$.
      iv. Let $m = m - 1 + m_2, d = d - 1 + t_2$.
3. Return the matrix M as M_S

4.2 Examples

For the access structure in Example 1, the modified converting algorithm will work as follows.

Example 4. The access structure is

$$((A \wedge B) \vee (B \wedge C) \vee (C, D, E, 2)) \wedge (E, F, G, H, 3),$$

and the formatted boolean formula is

$$(((B, (A, C, 1), 2), (C, D, E, 2), 1), (E, F, G, H, 3), 2).$$

Considering the authorized set $S_2 = \{D, E, F, G\}$,

1. $M'_{S_2} = \begin{pmatrix} 1 \end{pmatrix}, L'_{S_2} = \begin{pmatrix} (((B, (A, C, 1), 2), (C, D, E, 2), 1), (E, F, G, H, 3), 2) \end{pmatrix}$
2. $M'_{S_2} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, L'_{S_2} = \begin{pmatrix} ((B, (A, C, 1), 2), (C, D, E, 2), 1) \\ (E, F, G, H, 3) \end{pmatrix}$

3. $M'_{S_2} = \begin{pmatrix} 1\,1 \\ 1\,1 \\ 1\,2 \end{pmatrix}$, $L'_{S_2} = \begin{pmatrix} (B,(A,C,1),2) \\ (C,D,E,2) \\ (E,F,G,H,3) \end{pmatrix}$

4. $M'_{S_2} = \begin{pmatrix} 1\,1 \\ 1\,2 \end{pmatrix}$, $L'_{S_2} = \begin{pmatrix} (C,D,E,2) \\ (E,F,G,H,3) \end{pmatrix}$

5. $M'_{S_2} = \begin{pmatrix} 1\,1\,1 \\ 1\,1\,2 \\ 1\,1\,3 \\ 1\,2\,0 \end{pmatrix}$, $L'_{S_2} = \begin{pmatrix} C \\ D \\ E \\ (E,F,G,H,3) \end{pmatrix}$

6. $M'_{S_2} = \begin{pmatrix} 1\,1\,2 \\ 1\,1\,3 \\ 1\,2\,0 \end{pmatrix}$, $L'_{S_2} = \begin{pmatrix} D \\ E \\ (E,F,G,H,3) \end{pmatrix}$

7. $M'_{S_2} = \begin{pmatrix} 1\,1\,2\,0 & 0 \\ 1\,1\,3\,0 & 0 \\ 1\,2\,0\,1 & 1 \\ 1\,2\,0\,2 & 4 \\ 1\,2\,0\,3 & 9 \\ 1\,2\,0\,4 & 16 \end{pmatrix}$, $L'_{S_2} = \begin{pmatrix} D \\ E \\ E \\ F \\ G \\ H \end{pmatrix}$

8. $M'_{S_2} = \begin{pmatrix} 1\,1\,2\,0\,0 \\ 1\,1\,3\,0\,0 \\ 1\,2\,0\,1\,1 \\ 1\,2\,0\,2\,4 \\ 1\,2\,0\,3\,9 \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}'_5 \\ \boldsymbol{v}'_6 \\ \boldsymbol{v}'_7 \\ \boldsymbol{v}'_8 \\ \boldsymbol{v}'_9 \end{pmatrix}$, $L'_{S_2} = \begin{pmatrix} D \\ E \\ E \\ F \\ G \end{pmatrix}$.

The reconstruction constants of S_2 are $\{\omega'_5 \equiv 6, \omega'_6 \equiv -4, \omega'_7 \equiv -3, \omega'_8 \equiv 3, \omega'_9 \equiv -1\} \mod 47$.

Note that in Example 1, $M_{S_2} = \begin{pmatrix} 1\,1\,0\,2\,0\,0 \\ 1\,1\,0\,3\,0\,0 \\ 1\,2\,0\,0\,1\,1 \\ 1\,2\,0\,0\,2\,4 \\ 1\,2\,0\,0\,3\,9 \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}_5 \\ \boldsymbol{v}_6 \\ \boldsymbol{v}_7 \\ \boldsymbol{v}_8 \\ \boldsymbol{v}_9 \end{pmatrix}$, $L_{S_2} = \begin{pmatrix} D \\ E \\ E \\ F \\ G \end{pmatrix}$,
and the reconstruction constants of S_2 are also $\{\omega_5 \equiv 6, \omega_6 \equiv -4, \omega_7 \equiv -3, \omega_8 \equiv 3, \omega_9 \equiv -1\} \mod 47$.

The Decrypt algorithm works well with the modified converting algorithm. Essentially, some all-zero columns are "missing" in the modified converting algorithm, but the reconstruction constants are not affected. Thus we can use the modified converting algorithm in Decrypt algorithm to spare computation and space cost.

5 Conclusion

In this paper, we present a general and efficient algorithm for converting boolean formula into corresponding LSSS matrix whose size is as small as possible. And by some tricks, the boolean formula acts as the label function simultaneously. Thus, embedding this converting algorithm in the Encrypt and Decrypt algorithm of the CP-ABE schemes, only formatted boolean formula needs to be attached to the ciphertext so that the communication cost is drastically decreased.

References

1. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Ph.D. thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. pp. 321–334. IEEE Computer Society (2007)
3. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security. pp. 89–98. ACM (2006)
4. Karchmer, M., Wigderson, A.: On span programs. In: Structure in Complexity Theory Conference. pp. 102–111 (1993)
5. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. Cryptology ePrint Archive, Report 2010/351 (2010), http://eprint.iacr.org/
6. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 6110, pp. 62–91. Springer (2010)
7. Nikov, V., Nikova, S.: New monotone span programs from old. Cryptology ePrint Archive, Report 2004/282 (2004), http://eprint.iacr.org/
8. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 3494, pp. 457–473. Springer (2005)
9. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
10. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290 (2008), http://eprint.iacr.org/