KP-HABE: A Fully Secure Key-Policy Hierarchical Attribute-Based Encryption

Yu-qiao DENG¹, Ge SONG²*, and Ya-min WEN¹

¹School of Mathematics And Statistics, Guangdong University of Finance and Economics, Guangzhou, China
²College of Mathematics And Informatics, South China Agricultural University, Guangzhou, China

*Corresponding author

Keywords: Attribute-based encryption, KP-HABE, Fully security, Dual system encryption.

Abstract. One drawback of the traditional KP-ABE is the limitation for expressing the hierarchical structure of attributes. In this paper, we firstly propose a fully secure Key-Policy Hierarchical Attribute-based encryption (KP-HABE). In KP-HABE, attribute is expressed by a vector in order to facilitate the indication of the hierarchical structure of attribute. The technique of composite order bilinear pairing and linear secret sharing scheme is used to build the KP-HABE scheme.

Introduction

Attribute Based Encryption (ABE), which has been presented by Sahai and Waters[1], is an influential paradigm for embedding complex access policy into the encrypted data. Key - Policy Attribute Based Encryption (KP - ABE) and Cipher - Policy Attribute Based Encryption (CP - ABE) are two typical kinds of ABE schemes[2]. In KP - ABE, ciphertexts are associated with the attribute set and private keys are associated with the access policy, in CP - ABE, ciphertexts are associated with the access policy and private keys are associated with the attribute set.

Our contribution in this paper is multifold, we describe it in detail as follows. In traditional KP - ABE, all the distributions of private keys rely on the TA which results in the low performance of the system. To address this problem, we present the first fully secure Key-Policy Hierarchical Attribute-Based Encryption. Technically speaking, we import the skill proposed by Boneh et al. [3] to express the hierarchical property of private keys, then, we utilize the Linear Secret Sharing Scheme (LSSS) to split a master secret into multiple shares. Finally, we adopt the technique of Lewko and Waters [4] to "bind" the secret shares to the hierarchical private keys.

Composite Order Bilinear Groups

Let \( G \) be a group generator and \( \ell \) be the security parameter. Composite order bilinear groups [5] can be defined as: \( (N = p_1p_2p_3, G, G_T, e) \leftarrow \mathcal{R} G(1^\ell) \), for which \( p_1, p_2, p_3 \) are three distinct primes, \( G, G_T \) are cyclic groups with order \( N \) and the group operation in \( G, G_T \) is computable in polynomial time with respect to the security parameter \( \ell \). A bilinear map \( e : G \times G \rightarrow G_T \) is an efficiently computable map with the following properties:

- **Bilinearity:** for all \( a, b \in \mathbb{Z}_N \) and \( g_1, g_2 \in G \), \( e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} \).
- **Non-degeneracy:** \( \exists g \in G \) such that \( e(g, g) \) has order \( N \) in \( G_T \).

Suppose that \( G_i \) denotes the subgroup of order \( p_i \) and \( G_{i(j)} \) denotes the subgroup of order \( p_i p_j \). The orthogonality property of \( G_1, G_2, G_3 \) is defined as follows: for all \( h_i \in G_i, h_j \in G_j \) it holds that \( e(h_i, h_j) = 1 \), for which \( i \neq j \in \{1, 2, 3\} \). We demonstrate why the above property holds: let \( h_1 \in G_1, h_2 \in G_2 \) and \( g \) be a generator of \( G \). Then \( g^{p_1 p_2} \) generates \( G_3 \), \( g^{p_1 p_3} \) generates \( G_2 \), and \( g^{p_2 p_3} \) generates \( G_1 \). Therefore, for some \( a, b \), we can express \( h_1, h_2 \) as:

\[
h_1 = (g^{p_2 p_3})^a, h_2 = (g^{p_1 p_3})^b.
\]
Then we have:

\[ e(h_1, h_2) = e((g^{p_2 p_3})^a, (g^{p_1 p_3})^b) = 1. \]

**Key-Policy Hierarchical Attribute-Based Encryption**

A Key-Policy Hierarchical Attribute-Based Encryption (KP-HABE) includes five algorithms: Setup, Encrypt, KeyGen, Decrypt and Delegate.

- **Setup(\(1^\lambda\))**: The algorithm takes the security parameter \(\lambda\) as input and it takes the public key PK and a master secret key MSK as output.

- **Encrypt(m, PK, S)**: The algorithm takes a message \(m\), the public key PK and an attribute set S as inputs. It takes the ciphertext CT as output.

- **KeyGen(PK, MSK, A)**: The algorithm takes an access structure \(A\), the master secret key MSK and public key PK as inputs. It outputs a secret key \(SK_A\) for the access structure \(A\).

- **Delegate(PK, SK_A, A_1, A_2)**: The algorithm takes public key PK, a secret key \(SK_A\) associated with the access structure \(A_1\), an attribute set \(A_1\) of depth \(k\) corresponding to the access structure \(A_1\), an access structure \(A_2\), an attribute set \(A_2\) of depth \(k+1\) corresponding to the access structure \(A_2\) as inputs. It takes the secret key \(SK_{A_2}\) for the access structure \(A_2\) as output.

- **Decrypt(CT, SK_A)**: The algorithm takes a ciphertext CT associated with the attribute set S, secret keys \(SK_A\) for the access structure \(A\) as input. If \(S \in A\), it outputs \(m\); otherwise, outputs \(\perp\).

**Our KP-HABE Scheme**

Given a group description \(\langle N, G, G_T, e \rangle \leftarrow G(\lambda)\), where \(N = p_1 p_2 p_3, G, G_T\) are groups of order \(N\), \(e\) is a map: \(G \times G \rightarrow G_T\). The attribute universe is \(U = \mathbb{Z}_n\). Our scheme consists of four algorithms, we demonstrate them as follows.

- **Setup(\(1^\lambda\))**: The algorithm takes the security parameter \(\lambda\) as input. Assume that the maximum depth of the attribute vector in our scheme is \(I\). The algorithm picks random generators \(g \in G_1, g_3 \in G_3\), random elements \(\alpha, a \in \mathbb{Z}_N\) and random group elements \(h_1, ..., h_I\). It outputs \(PK = \langle N, G, G_T, e, g, g_3, g^a, h_1, ..., h_I, e(g, g^a) \rangle\). The master secret key is \(MSK = \alpha\).

- **KeyGen(PK, MSK, A)**: The algorithm takes an access structure \(A = (M, \rho)\) (where \(M\) is the secret sharing matrix with \(l\) rows and \(n\) columns, \(\rho\) is the function mapping one row of \(M\) to an attribute vector of depth \(k\)), the master key MSK and public key PK as inputs. The algorithm chooses secret vector \(\vec{y} = (\alpha, y_2, ..., y_n)\) where \((y_2, ..., y_n) \leftarrow \mathbb{Z}_{p_1}^{l-1}\). Notice that \(\alpha\) is the secret to be shared. The algorithm evaluates the \(i\)th shares of the secret \(\alpha\) as:

\[
\lambda_i = M_i \cdot \vec{y}, i \in \{1, ..., l\},
\]

where \(M_i\) indicates the \(i\)th row of the access matrix \(M\).

Recall that the maximum depth of the attribute vector is \(I\). For each \(j\) from 1 to \(l\), the algorithm picks random elements:

\[
t_j \in \mathbb{Z}_N, R_{j,0}, R_{j,1}, R_{j,2}, R_{j,k+1}, ..., R_{j,l} \in G_3.
\]

Assume that the attribute vector associated with the row \(j\) is \(\vec{u} = (u_1, ..., u_k)\), the algorithm uses a group element \((h_1 u_1, ..., h_k u_k)\) to express the attribute vector \(\vec{u} = (u_1, ..., u_k)\) and computes:
Finally, the algorithm outputs the private keys as: $SK_A = (K_{j,0}, K_{j,1}, K_{j,2}, K_{j,k+1}, ..., K_{j,L})_{j=1}^L$.

Delegate ($PK, SK_{A_1}, A_2$): The algorithm takes public key $PK$, a secret key associated with the access structure $A_1 = (M_1, \rho_1)$, an attribute set $A_1$ of depth $k$ corresponded to the access structure $A_1$, an access structure $A_2 = (M_2, \rho_2)$, an attribute set $A_2$ of depth $k+1$ corresponded to the access structure $A_2$ as inputs.

Recall that, according to the definition 3, if $\rho_1$ holds, then for any attribute $\vec{u}_3$ of depth $k+1$ in $A_2$, it must exist $\vec{u}_1 \in A_1$ and $\vec{u}_2 = (\vec{u}_1, u_{k+1})$ holds. Assume that the matrix $M_1$ has $l_1$ rows and matrix $M_2$ has $l_2$ rows. Let the given secret key be:

$SK_A = (K_{i,0}, K_{i,1}, K_{i,2}, K_{i,k+1}, ..., K_{i,L})_{i=1}^{l_1}$

Then the algorithm can generate the key for $A_2$ through delegating the key of $A_1$. For each $j$ from 1 to $l_2$, the algorithm chooses random elements:

$t_j \in \mathbb{Z}_N, R_{j,0}, R_{j,1}, R_{j,2}, R_{j,k+1}, ..., R_{j,L} \in \mathbb{G}_3$.

For each attribute vector $\vec{u} \in A_2$ associated with the $j$th row in the access matrix $M_2$, it must exist prefix $\vec{u}' \in A_1$ such that $\vec{u} = (\vec{u}', u_{k+1})$. Let the $j$th row is the corresponded row of the attribute vector $\vec{u}'$ in the access structure $A_1$, that is, $\rho_1(j') = \vec{u}'$.

The algorithm generates the keys for the attribute vector $\vec{u}'$ as follows:

$K_{j,0} = K'_{j,0}g^a_j R_{j,0}$,
$K_{j,1} = K'_{j,1}h_1^{u_1} \cdots h_{k+1}^{u_{k+1}} R_{j,1}$,
$K_{j,2} = K'_{j,2}g^b_j R_{j,2}$,
$K_{j,k+2} = K'_{j,k+2}h_{k+2}^{t_j} R_{j,k+2}$,

$K_{j,L} = K'_{j,L}h_L^{t_j} R_{j,L}$.

Finally, the algorithm outputs the secret keys:

$SK_S = (K_{j,0}, K_{j,1}, K_{j,2}, K_{j,k+2}, ..., K_{j,L})_{j=1}^{l_2}$.

Indeed, the algorithm transforms the expression of attribute vector from $(u_1, ..., u_k)$ to $(u_1, ..., u_k, u_{k+1})$ and sets $t_j$ to be $t_j = t_j + t'_j$. Note that the keys generated in the delegate algorithm is identically distributed to keys generated by the way of directly calls the KeyGen algorithm.

Encrypt($m, PK, S$). Given the set $S$ of attribute vectors of depth $k \leq L$, the algorithm encrypts a message $m$ with the following mechanism.

Let $\tau = |S|$ denote the number of attribute vectors in the set $S$, the algorithm picks $\tau + 1$ random exponents $(s, r_1, r_2, ..., r_\tau) \leftarrow \mathbb{Z}_p^{\tau+1}$, then it computes:

$C' = m \cdot e(g, g)^{as}$,
$C_0 = g^s$,
$C_{i,1} = g^{r_i}$,
$C_{i,2} = (h_1^{u_1} \cdots h_k^{u_k})^{r_i} g^{-as}$. ($i = 1, ..., \tau$)

The algorithm outputs $CT = (C, C_0, \{C_{i,1}, C_{i,2}\}_{i=1}^\tau)$.
Decrypt(CT, SK). The algorithm takes a ciphertext CT associated with the attribute vector set S, secret keys $SK_S$ for access structure $A$ as input. If $S \not\in A$, the algorithm outputs $1$. If $S \in A$, let $I = \{i : \mu(i) \in S\}$, the algorithm can efficiently find such constants $\omega_i \in Z_N$ for which $i \in I$ subjected to $\sum_{i \in I} \omega_i M_i = (1, 0, \ldots, 0)$, where $M_i$ is $i$th row of the matrix $M$.

The algorithm computes:

$$B = \prod_{i \in I} e(C_0, K_{i,0})e(C_{i,1}, K_{i,1})e(C_{i,2}, K_{i,2})^{\omega_i}$$
$$= \prod_{i \in I} e(g, g)^{\omega_i \lambda_i}$$
$$= e(g, g)^{\alpha \lambda}.$$

Finally, the algorithm outputs $m = C/B$.

Summary

In this paper, we initially propose a new type of ABE, KP-HABE. We import the hierarchical attribute mechanism to the classical ABE to provide the ability of delegating private keys. We describe the scenario that existed ABE cannot handle. We utilize the technique of LSSS and composite order bilinear map to construct our scheme.

Acknowledgement

This work is supported by the Ministry of education of Humanities and Social Science project (No. 15YJCZH029), the Social science planning project of Guangzhou City (No. 2016gzyb25), the National Natural Science Foundation of China (No. 61300204), the Science and Technology Planning Project of Guangdong Province, China under Grant No. 2016A020210103 and the Natural Science Foundation of Guangdong (No. S2012040006711).

References