Large Universe Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability

Jianting Ning[1], Zhenfu Cao[1], Xiaolei Dong[1], Lifei Wei[2], and Xiaodong Lin[3]

[1] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{jtning@,zfcao@cs.,dong-xl@cs.}sjtu.edu.cn
[2] College of Information Technology,
Shanghai Ocean University, Shanghai 201306, China
Lfwei@shou.edu.cn
[3] Faculty of Business and Information Technology,
University of Ontario Institute of Technology, Oshawa, Canada
xiaodong.lin@uoit.ca

Abstract. A Ciphertext-Policy Attribute-Based Encryption (CP-ABE) system extracts the decryption keys over attributes shared by multiple users. It brings plenty of advantages in ABE applications. CP-ABE enables fine-grained access control to the encrypted data for commercial applications. There has been significant progress in CP-ABE over the recent years because of two properties called *traceability* and *large universe*, greatly enriching the commercial applications of CP-ABE. Traceability is the ability of ABE to track the malicious users or traitors who intentionally leak the partial or modified decryption keys to others for profits. Nevertheless, due to the nature of CP-ABE, it is difficult to identify the original key owner from an exposed key since the decryption privilege is shared by multiple users who have the same attributes. On the other hand, the property of large universe in ABE proposed by Lewko and Waters enlarges the practical applications by supporting flexible number of attributes. Several systems have been proposed to obtain either of the above properties. However, none of them achieve the two properties simultaneously in practice, which limits the commercial applications of CP-ABE to a certain extent. In this paper, we propose a practical large universe CP-ABE system supporting white-box traceability, which is suitable for commercial applications. Compared to existing systems, our new system has three advantages: (1) The number of attributes is not polynomially bounded; (2) Malicious users who leak their decryption keys could be traced; and, (3) The storage overhead for traitor tracing is constant. We also prove the selective security of our new system in the standard model under "q-type" assumption.

Keywords: Attribute-Based Encryption, Ciphertext-Policy, Large Universe, White-box Traceablity, Commercial Applications.

1 Introduction

In traditional public key encryption, a user is privileged to share his/her data with others in a private manner. The access of a targeted user or device to the shared data is all or nothing. In other words, one can get the entire access capability to the shared data if given the secret key; otherwise, nothing will be revealed. In many cases, however, this may not be true. For example, a user may expect to share his/her data through a more general and expressive way based on the targeted user or device's credentials. To address this issue, Sahai and Waters [34] introduced the notion of Fuzzy Identity-Based Encryption (FIBE). Goyal et al. [12] proposed two complementary forms of Attribute-Based Encryption (ABE) : Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). In the KP-ABE, users' decryption keys are issued according to an access policy and the ciphertexts are annotated by attributes. In the CP-ABE, users' decryption keys are issued according to the attributes they possess and the encrypting party specifies an access policy for the ciphertexts. A series of KP-ABE or CP-ABE schemes have been proposed [3, 29, 10, 16, 36, 27, 20, 33, 32, 14, 24], aiming at better expressiveness, efficiency or security. In particular, large universe and traceability are the two significant progress in ABE, we will discuss following.

Recently, Rouselakis and Waters [32] proposed a new construction and its proof method for Large Universe Attribute-Based Encryption (LU-ABE). In general, an ABE system can be classified to "small universe" and "large universe" constructions. In the "small universe" construction, the attributes are fixed at system setup and the size of the attributes is polynomially bounded, and furthermore the size of public parameters grows linearly with the number of attributes. While in the "large universe" construction, the attributes need not be specified at system setup and the size of the attribute universe is unbounded. The "large universe" construction for ABE system brings an obvious advantage that the designer of the ABE system need not bother to choose a particular bound of the attributes at system setup.

On the other hand, several CP-ABE systems supporting traceability have been proposed [22, 21, 24]. In CP-ABE, each user possesses a set of attributes and can decrypt the ciphertext if his/her attributes satisfy the ciphertext's access policy. This results in an obvious consequence that the encrypter or system does not know who leaks the decryption key to others intentionally. Due to the fact that the attributes are shared by multiple users and different users may have the same subset of attributes, the encrypter or system has no feasible method to trace the suspicious receiver if the the decryption key is leaked. We take Alice (with attributes {Alice, Assistant Professor, Computer Science}) and Bob (with attributes {Bob, Assistant Professor, Computer Science}) as an example. They both have the same decryption key corresponding to attributes {Assistant Professor, Computer Science} and can decrypt such a ciphertext encrypted by the attributes {Assistant Professor, Computer Science}. Suppose no other receiver in the system has both attributes ({Assistant Professor} and {Computer Science}) at the same time. If it happens to exist a user who can decrypt the

ciphertext except Alice and Bob, it is significant to find out who leaks such decryption key to him, Alice or Bob? This drawback should be fixed in practice in case of leaking decryption key. It is necessary to add the property of traceability to the original ABE scheme, to identify who exactly leaks the decryption key. The above traceability is called *white-box traceability* [24], which means that any user who leaks his/her decryption key to the third user or device intentionally or unintentionally will be identified. Also note that there exists a relatively stronger notion named *black-box traceability* [23]: the leakage of the user is the decryption equipment instead of its decryption key.

However, there exists no practical traceable CP-ABE system supporting the property of large universe as the (non-traceable) CP-ABE system in [32]. Large universe CP-ABE system with white-box traceability is not yet achieved in practice: (1) The CP-ABE systems supporting traceability proposed in [22, 21, 24] do not support the property of large universe, the attributes need to be fixed at system setup and the size of the attributes is polynomially bounded. Also, public parameters' size grows linearly with the number of attributes. (2) The large universe CP-ABE system proposed in [32] is the first large universe CP-ABE system secure in the standard model; however, it does not support the property of traceability.

A Motivating Story. Consider a commercial application such as a pay-TV system with huge number of users for example. Each user is labeled with lots of related attributes, which which are defined as TV channels that the user have ordered. As a versatile one-to-many encryption mechanism, CP-ABE system is quite suitable in this scenario. The pay-TV system provides several TV channels for users, and those who have paid for the TV channels could satisfy the access policy to decrypt the ciphertext and enjoy the ordered TV channels. CP-ABE enables fine-grained access control to the encrypted data according to attributes in users' ordered lists. However, there are two problems with this approach. First, if someone (who does not have the privilege to access to those TV channels) buys the decryption key from the Internet at a lower cost, she/he could also get access to the TV channels. Then who is selling the decryption key? Second, as the TV channels of the pay-TV system expand, an increasing number of new attributes need to be added to the system to describe the new channels. If the number of the attributes exceeds the bound set during the initial deployment of the pay-TV system, then the entire system has to be re-deployed and possibly all its data will have to be re-encrypted [32].

The problems, as described above, are the main obstacles when CP-ABE is implemented in commercial applications such as pay-TV systems and social networks. Due to the nature of CP-ABE, if a malicious user leaks its decryption key to others for profits (such as selling the decryption key on the Internet), it is difficult to find out the original key owner from an exposed key since the decryption key is shared by multiple users who have the same attributes. As such, the pay-TV company will suffer severe financial loss. Thus, it is necessary for the pay-TV system to trace the malicious users who intentionally leak the partial or modified decryption keys. Also, as the pay-TV system expands, an

increasing new attributes (which describe new TV channels) have to be added to the system. In previous CP-ABE constructions, the attributes are fixed at system setup and the number of the attributes are bounded. If the bound is not specified large enough, the attributes may exhaust if the number of the users exceeds the threshold and the entire system needs to be completely re-built [32]. On the other hand, if the bound is specified too large, it will increase the storage and communication burden of the entire system due to the corresponding increase of the public parameters' size. Thus, it is necessary for the pay-TV system to support flexible number of attributes. Lastly, since the number of users in a pay-TV system could grow fast, the storage for traceability should not increase linearly with the number of users. Otherwise, the storage for traceability will become relatively huge and exhaust if the users increase dramatically. Thus, the storage for traceability needs to be at a constant level in an ideal case.

1.1 Our Contribution

In this paper, we propose a new large universe CP-ABE system which is white-box[4] traceable on prime order bilinear groups. To the best of our knowledge, this is the first practical CP-ABE system that simultaneously supports the following three properties: white-box traceability, large universe and constant storage for tracing. Compared with other constructions using composite order groups, we build our construction on the efficient prime order bilinear groups. We also prove our new system selectively secure in the standard model.

We solve the obstacles of CP-ABE implementation in the commercial applications such as pay-TV systems and social networks as follows:

1. We achieve the property of white-box traceability in CP-ABE. Our new system can trace the malicious users who may leak the partial or modified decryption keys to others for profits.
2. We obtain the property of large universe in white-box traceable CP-ABE. In our new system attributes need not be fixed at system setup, the attributes' size is not polynomially bounded and the public parameters' size does not grow linearly with the number of attributes.
3. We do not need to maintain an identity table for tracing as used in [24]. Instead, we adopt the Shamir's $(\bar{t}, \bar{n})$ threshold scheme in tracing the malicious users, the storage cost for traceability does not grow linearly with the number of the users, it is constant which only depends on the threshold $\bar{t}$.
4. It yields another result that the stored data for traceability need not be updated when new users are added into the system or malicious users are ejected out of the system, which makes the system more practical for applications.

Table 1 gives the comparison between our work and some other related work.

[4] In this paper, we mainly aim to obtain a large universe CP-ABE system with white-box traceability. The realization of black-box traceability for large universe CP-ABE system will be our future work.

Table 1. Comparison with other related work

	[22]	[21]	[24]	[32]	Ours
Large Universe [1]	×	×	×	√	√
Traceability	√	√	√	×	√
Constant Storage for Tracing [2]	√	×	×	−	√
Supporting Any Monotone Access Structures [3]	×	×	√	√	√
Constructed on Prime Order Groups [4]	√	×	×	√	√
Standard Model	×	√	√	√	√

[1] In [22],[21] and [24], their systems only support small universe.
[2] In [21] and [24], the storage for tracing is not constant. In [32], the proposed system does not support traceability.
[3] In [22] and [21], their systems do not support any monotone access structures.
[4] In [21] and [24], their systems are constructed on the composite order groups.

1.2 Our Technique

In this subsection, we briefly introduce the main idea we utilize to realize the properties of large universe and white-box traceability before giving the full details in Section 4.

To realize large universe construction, we adopt the "individual randomness" and "layer" technique from [19, 32]. We use the "layer" technique to encrypt data securely and to be able to decrypt. We employ two "layers" : the "attribute" layer and the "secret sharing" layer, and use a "binder term" to connect these two layers securely. In the "attribute" layer, we utilize u, h terms to provide a Boneh-Boyen-style [4] hash function ($u^A h$). As for the "secret sharing" layer, during `KeyGen` and `Encrypt` phases we use w term to hold the secret randomness r and the secret randomness s's shares respectively. Finally, we use the v term to "bind" this two layers together.

To realize traceability, we use the Boneh-Boyen-style signature [4]. Compared with the related work [24], we find that the table T with the tuple identity and its randomness used in [24] grows linearly with the number of the users.[5] With the number of the users in a system scaling large, the corresponding identity table T for traceability will expand as a result, which leads to heavy burden of the storage space for T. Besides, the corresponding cost of searching K' in T during the `Trace` phase is relatively huge. In this paper, we utilize the Shamir's $(\bar{t}, \bar{n})$ threshold scheme to optimize the property of traceability. We only need store $\bar{t} - 1$ points on a polynomial $f(x)$ at system setup. Consequentially, our storage for traceability does not grow linearly with the number of the users and is a constant.

The main idea of our traceability is as follows.

[5] Note that in the extension of [24], it gives another signature scheme for the purpose of removing the identify table T, but unfortunately the new signature scheme is not as efficient as the original one. Besides, it brings some other parameters, which will cause additional computation overhead.

Firstly, the `Setup` algorithm initializes an instance of Shamir's $(\bar{t}, \bar{n})$ threshold scheme $\mathcal{INS}_{(\bar{t},\bar{n})}$ and keeps a polynomial $f(x)$ and $\bar{t}-1$ points $\{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$ on $f(x)$ secret. Then we insert c into the decryption key sk during `KeyGen` phase where $c = Enc_{\bar{k_2}}(x||y), x = Enc_{\bar{k_1}}(id), y = f(x)$.[6] During the `Trace` phase, the algorithm extracts $(x^* = x', y^* = y')$ from $x'||y' = Dec_{\bar{k_2}}(K')$ in the decryption key sk, and then it checks whether sk is issued by system. If $(x^* = x', y^* = y') \in \{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$, the algorithm computes $Dec_{\bar{k_1}}(x^*)$ to get id to identify the malicious user directly. Otherwise, the algorithm computes the secret of $\mathcal{INS}_{(\bar{t},\bar{n})}$ by interpolating with $\bar{t}-1$ points $\{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$ and (x^*, y^*). If the recovered secret is equal to $f(0)$, the algorithm computes $Dec_{\bar{k_1}}(x^*)$ to get id to identify the malicious user. If the equation fails, sk is not issued by the system. In this way, we could trace the owner of the decryption key. Meanwhile, it brings the benefit that the system only stores $\bar{t}-1$ points on $f(x)$, and thus the storage for traceability is a constant.

1.3 Related Work

Sahai and Waters introduced the notion of Fuzzy Identity-Based Encryption in [34]. Goyal, Pandey, Sahai and Waters [12] later formalized two notions of ABE: CP-ABE (where user keys are labeled with sets of attributes and ciphertexts are associated with policies) and KP-ABE (where ciphertexts are labeled with sets of attributes and private keys are associated with access structures). Subsequently, many constructions of selectively secure KP-ABE and CP-ABE systems were proposed [3, 29, 7, 29, 10, 16, 27, 36, 1]. Many advances have been made for ABE as the following directions: new proof techniques to obtain fully secure [17, 27, 16, 20], decentralizing trust by setting multiple authorities [5, 6, 18] and outsourcing computation [30, 13]. The first large universe KP-ABE construction was proposed in [19]. It was built on composite order groups and proved selectively secure in the standard model. Then the first large universe KP-ABE construction on prime order groups was proposed in [15] inspired by the dual pairing vector space framework [25, 26, 28]. Recently, the first large universe CP-ABE construction [32] built on prime order bilinear groups was proposed by Rouselakis and Waters. It was proved selectively secure in the standard model under "q-type" assumption. Another branch of ABE research considers the problem of traceability. The notion of accountable CP-ABE was first proposed in [22] to prevent illegal key sharing among colluding users. Then a multi-authority ciphertext-policy (AND gates with wildcard) ABE scheme with accountability was proposed in [21], which allowed tracing the identity of a misbehaving user who leaked the decryption key to others. Liu, Cao and Wong lately proposed a white-box [24] and black-box [23] traceability CP-ABE system which supported policies expressed in any monotone access structures.

[6] Note that the tuple (x, y) is a point on $f(x)$

1.4 Organization

Section 2 gives the formal definition of traceable large universe CP-ABE and its security model. Section 3 introduces the background, including the notation, the access policy, the linear secret sharing scheme, the prime order bilinear groups and the assumptions. Section 4 presents the construction of our T-LU-CP-ABE system as well as the security proof. Some extensions of our work are discussed in Section 5. Finally, Section 6 presents a briefly conclusion and foresees our future work.

2 Traceable Large Universe CP-ABE

2.1 Definition

A Traceable Large Universe CP-ABE (T-LU-CP-ABE) system is a CP-ABE system where attributes need not be fixed at system setup and can trace the user by his decryption key. We enhance the original large universe CP-ABE system by adding users' identities and a `Trace` algorithm to it according to [24]. In particular, following the notation of the large universe CP-ABE system introduced in [32], a T-LU-CP-ABE system consists of five algorithms as follows:

- $\texttt{Setup}(1^\lambda) \rightarrow (pp, msk)$: The algorithm takes as inputs a security parameter $\lambda \in \mathbb{N}$ encoded in unary. It outputs the public parameters pp and the master secret key msk. We assume that the description of the attribute universe U is contained in the public parameters.[7] In addition, it initializes an instance of Shamir's $(\bar{t}, \bar{n})$ threshold scheme denoted by $\mathcal{INS}_{(\bar{t},\bar{n})}$.
- $\texttt{KeyGen}(1^\lambda, pp, msk, id, S) \rightarrow sk_{id,S}$: The key generation algorithm takes as inputs the public parameters pp, the master secret key msk and a set of attributes $S \subseteq U$ for a user with identity id. The security parameter in the inputs ensures that it is polynomial time in λ. The algorithm outputs a secret key $sk_{id,S}$ corresponding to S.
- $\texttt{Encrypt}(1^\lambda, pp, m, \mathbb{A}) \rightarrow ct$: The encryption algorithm takes as inputs the public parameters pp, a plaintext message m, and an access structure $\mathbb{A}$ over U. It outputs the ciphertext ct.
- $\texttt{Decrypt}(1^\lambda, pp, sk_{id,S}, ct) \rightarrow m$ or $\perp$: The decryption algorithm takes as inputs the public parameters pp, a secret key $sk_{id,S}$, and a ciphertext ct. It outputs the plaintext m or $\perp$.
- $\texttt{Trace}(pp, \mathcal{INS}_{(\bar{t},\bar{n})}, msk, sk) \rightarrow id$ or $\intercal$: The tracing algorithm takes as inputs the public parameter pp, an instance of of Shamir's $(\bar{t}, \bar{n})$ threshold scheme $\mathcal{INS}_{(\bar{t},\bar{n})}$, the master secret key msk, and a secret key sk. The algorithm first verifies whether sk is well-formed to determine whether sk needs to be traced. If sk is well-formed and could recover the secret of $\mathcal{INS}_{(\bar{t},\bar{n})}$,

[7] In the previous CP-ABE systems, the attribute universe U was one of the arguments in the `Setup` algorithm. In the large universe case, the attribute universe only depends on the size of the security parameter and the group generation algorithm [32].

the algorithm outputs an identity id implying that sk is linked to id. Otherwise, it outputs a special symbol $\intercal$ implying that sk does not need to be traced. We define a secret key sk is *well-formed* which means that it passes the *key sanity check* algorithm. The key sanity check is a deterministic algorithm [9, 11], which is used to guarantee the secret key in the well-formed decryption process.

2.2 T-LU-CP-ABE Selective Security

The security model of our T-LU-CP-ABE scheme is similar to that of the LU-CP-ABE scheme [32], excepting every key query is companied with an explicit identity. In this subsection we present the definition of selective security for our T-LU-CP-ABE scheme. It is parameterized by the security parameter $\lambda \in \mathbb{N}$ and is described by a game between an attacker and a challenger. The phases of the game are as follows:

- **Initialization** : The attacker claims the challenge access structure $\mathbb{A}^*$ he will attack, and then sends it to the challenger.
- **Setup** : The challenger runs the $\texttt{Setup}(1^\lambda)$ algorithm and sends the public parameters pp to the attacker.
- **Query Phase 1** : In this phase the attacker can adaptively ask for secret keys for the sets of attributes $(id_1, S_1), (id_2, S_2), ..., (id_{Q_1}, S_{Q_1})$. For each (id_i, S_i) the challenger calls $\texttt{KeyGen}(1^\lambda, pp, msk, id, S) \rightarrow sk_{id,S}$ and sends $sk_{id,S}$ to the attacker. The only restriction is that the attacker can not query the sets that satisfies the challenge access structure $\mathbb{A}^*$, i.e. $\forall_i \in [Q_1] : S_i \notin \mathbb{A}^*$.
- **Challenge** : The attacker declares two equal length messages m_0 and m_1 and sends them to the challenger. The challenge flips a random coin $\beta \in \{0, 1\}$ and calls $\texttt{Encrypt}(1^\lambda, pp, m_\beta, \mathbb{A}^*) \rightarrow ct$. It gives ct to the attacker.
- **Query Phase 2** : This is the same as query phase 1. The attacker asks for the secret key for the sets $(id_{Q_1+1}, S_{Q_1+1}), ..., (id_Q, S_Q)$ with the same restriction: $\forall_i \in [Q] : S_i \notin \mathbb{A}^*$.
- **Guess** : The attacker outputs a guess $\beta' \in \{0, 1\}$ for β.

The advantage of an attacker is defined to be $Adv = Pr[\beta' = \beta] - 1/2$ in this game.

Definition 1. *A traceable large universe ciphertext-policy attribute-based encryption system is selectively secure if all probabilistic polynomial-time (PPT) attackers have at most negligible advantage in λ in the above security game.*

2.3 Traceability

In this subsection, we give the traceability definition for our T-LU-CP-ABE. It is described by a game between an attacker and a challenger. The phases of the game are as follows:

- **Setup** : Here the challenger calls the $\texttt{Setup}(1^\lambda)$ algorithm and sends the public parameters pp to the attacker.
- **Key Query** : The attacker submits the sets of attributes $(id_1, S_1), ..., (id_q, S_q)$ to request the corresponding decryption keys.
- **Key Forgery** : The attacker will output a decryption key sk_*. If $\texttt{Trace}(pp, \mathcal{INS}_{(\bar{t},\bar{n})}, msk, sk_*) \neq \top$ and $\texttt{Trace}(pp, \mathcal{INS}_{(\bar{t},\bar{n})}, msk, sk_*) \notin \{id_1, ..., id_q\}$, then the attacker wins the game. The advantage of an attacker in this game is defined to be $Pr[Trace(pp, \mathcal{INS}_{(\bar{t},\bar{n})}, msk, sk_*) \notin \{\top, id_1, ..., id_q\}]$.

Definition 2. *A traceable large universe ciphertext-policy attribute-based encryption system is fully traceable if there has no polynomial time attacker have non-negligible advantage in the above game.*

3 Background

3.1 Notation

We define $[l] = \{1, 2, ..., l\}$ for $l \in \mathbb{N}$. By PPT we denote probabilistic polynomial-time. We denote $\mathbb{Z}_p^{l\times n}$ be the set of matrices of size $l \times n$ with elements in $\mathbb{Z}_p$. The set of row vectors of length n (i.e. $\mathbb{Z}_p^{1\times n}$) and the set of column vectors of length n (i.e. $\mathbb{Z}_p^{n\times 1}$) are the two special subsets. We denote $(s_1, s_2, ..., s_n)$ be a row vector and $(s_1, s_2, ..., s_n)^\perp$ be a column vector. By v_i we denote the i-th element in a vector $\boldsymbol{v}$. And by $M\boldsymbol{v}$ we denote the inner product of matrix M with vector $\boldsymbol{v}$. We define $\mathcal{F}(U_1 \to U_2)$ be the set of functions from set U_1 to U_2.

We denote $GD = (p, \mathbb{G}, \mathbb{G}_T, e)$ be the groups and the bilinear mapping description where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups of prime order p and $e : \mathbb{G} \times \mathbb{G}_T$ is a bilinear map.

3.2 Access Policy

This subsection presents the definition of access structure referred to [2, 32].

Definition 3. *(Access Structure [2]) : Let U denote the attribute universe. A collection $\mathbb{A} \in 2^U$ of non-empty sets of attributes is an access structure on U. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets. A collection $\mathbb{A} \in 2^U$ is called monotone if $\forall B, C \in \mathbb{A} : if$ $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.*

The main idea in ABE is that the role of the users is taken by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes. For CP-ABE, if a user of the system posses an authorized set of attributes then he can decrypt the ciphertext, otherwise, he can't get any information from ciphertext if the set he possed is unauthorized. In our construction, we restrict our attention to monotone access structure.

3.3 Linear Secret-Sharing Schemes

It is shown in [2] that a linear secret sharing scheme can realize any monotone access structure. In this subsection, we will present the definition of linear secret-sharing scheme (LSSS) referred to [2, 32].

Definition 4. *(Linear Secret-Sharing Schemes (LSSS) [2, 32]). Let U denote the attribute universe and p denote a prime. A secret-sharing scheme $\prod$ with domain of secrets $\mathbb{Z}_p$ realizing access structure on U in called linear (over $\mathbb{Z}_p$) if*

1. The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.

2. For each access structure $\mathbb{A}$ on U, there exists a matrix M with l rows and n columns called the share-generating matrix. For $i = 1, ..., l$, we define a function ρ labels row i of M with attribute $\rho(i)$ from the attribute universe U, i.e. $\rho \in \mathcal{F}([l] \to U)$. When we consider the column vector $\boldsymbol{v} = (s, r_2, ..., r_n)^{\perp}$, where $r_2, ..., r_n \in \mathbb{Z}_p$ are randomly chosen. Then $M\boldsymbol{v} \in \mathbb{Z}_p^{l \times 1}$ is the vector of l shares of the secret s according to $\prod$. The share $(M\boldsymbol{v})_j$ "belongs" to attribute $\rho(j)$, where $j \in [l]$.

As shown in [2], every linear secret-sharing scheme according to the above definition also enjoys the linear reconstruction property, defined as follows: Let $\prod$ be an LSSS for the access structure $\mathbb{A}$, $S \in \mathbb{A}$ be any authorized set and let $I \subset \{1, 2, ..., l\}$ be defined as $I = \{i \in [l] \wedge \rho(i) \in S\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that for any valid shares $\{\lambda_i = (M\boldsymbol{v})_i\}_{i \in I}$ of a secret s according to $\prod$, then $\sum_{i \in I} \omega_i \lambda_i = s$. Additionally, it is shown in [2] that these constants $\{\omega_i\}_{i \in I}$ can be found in time polynomial in the size of the share-generating matrix M. On the other hand, for any unauthorized set S', no such constants $\{\omega_i\}$ exist.

Also note that if we encode the access structure as a monotonic Boolean formula over attributes, there exists a generic algorithm by which we can generate the corresponding access policy in polynomial time [2, 18].

In our construction, an LSSS matrix (M, ρ) will be used to express an access policy associated to a ciphertext.

3.4 Prime Order Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. The bilinear map e has the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ can both be computed efficiently. Notice that the map $e(\cdot, \cdot)$ is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

3.5 Assumptions

We adopt the "q-type" assumption of [32] as this construction's assumption.

Assumption 1 ("q-type" assumption [32]) *We define the q-type problem as follows. Initially choose a group generation algorithm with input the security parameter, pick a random group element $g \in \mathbb{G}$, and $q+2$ random exponents $d, s, b_1, b_2, ..., b_q \in \mathbb{Z}_p$. If the attacker is given the group description $(p, \mathbb{G}, \mathbb{G}_T, e)$ and $\boldsymbol{y}$ including the following terms:*

$$
\begin{array}{ll}
g, g^s & \\
g^{d^i}, g^{b_j}, g^{sb_j}, g^{d^i b_j}, g^{d^i/b_j^2} & \forall (i,j) \in [q,q] \\
g^{d^i/b_j} & \forall (i,j) \in [2q,q] \text{ with } i \neq q+1 \\
g^{d^i b_j / b_{j'}^2} & \forall (i,j,j') \in [2q,q,q] \text{ with } j \neq j' \\
g^{sd^i b_j / b_{j'}}, g^{sd^i b_j / b_{j'}^2} & \forall (i,j,j') \in [q,q,q] \text{ with } j \neq j',
\end{array}
$$

it is hard for the attacker to distinguish $e(g,g)^{sd^{q+1}} \in \mathbb{G}_T$ from an element which is randomly chosen from $\mathbb{G}_T$.

An algorithm $\mathcal{A}$ that outputs $\beta \in \{0,1\}$ has advantage ϵ in solving the above assumption if $|\Pr[\mathcal{A}(\boldsymbol{y}, e(g,g)^{sd^{q+1}}) = 0] - \Pr[\mathcal{A}(\boldsymbol{y}, R) = 0]| \geq \epsilon$.

Definition 5. *We say that the q-type assumption holds if no PPT algorithm has a non-negligible advantage in solving the q-type problem.*

We define our l-SDH assumption according to [4, 9].

Assumption 2 *(l-SDH assumption [4, 9]) : Let $\mathbb{G}$ be a bilinear group of prime order p and g be a generator of $\mathbb{G}$, the l-Strong Diffie-Hellman (l-SDH) problem in $\mathbb{G}$ is defined as follows: given a $(l+1)$-tuple $(g, g^x, g^{x^2}, ..., g^{x^l})$ as inputs, output a pair $(c, g^{1/(c+x)}) \in \mathbb{Z}_p \times \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving l-SDH in $\mathbb{G}$ if $Pr[\mathcal{A}(g, g^x, g^{x^2}, ..., g^{x^l}) = (c, g^{1/(c+x)})] \geq \epsilon$, where the probability is over the random choice of x in $\mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$.*

Definition 6. *We say that the (l, t, ϵ)-SDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least in solving the l-SDH problem in $\mathbb{G}$.*

3.6 Shamir's $(\bar{t}, \bar{n})$ threshold scheme

It is well known for Shamir's $(\bar{t}, \bar{n})$ threshold scheme [35] (or Shamir's secret sharing scheme) in cryptography. The essential idea of that scheme is that $\bar{t}$ points on a $\bar{t}-1$ degree curve are sufficient to confirm such a curve, that is, $\bar{t}$ points are enough to determine a $\bar{t}-1$ degree polynomial. For a $(\bar{t}, \bar{n})$ threshold scheme, a secret can be divided into $\bar{n}$ parts (or even more), which are sent to each participant a unique part. All of them can be used to reconstruct the secret. Suppose that the secret is assumed to be an element in a finite field $\mathbb{F}_p^*$. Choose $\bar{t}-1$ random coefficients $a_1, a_2, \cdots, a_{\bar{t}-2} \in \mathbb{F}_p$ and $a_{\bar{t}-1} \in \mathbb{F}_p^*$ and set the secret in the constant term a_0. Note that, we have such a polynomial: $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{\bar{t}-1} x^{\bar{t}-1}$. Every participant is given a point (x, y) on the above curve, that is, the input to the polynomial x and its output $y = f(x)$. Given a subset with any $\bar{t}$ points, we can recover the constant term a_0 using the Lagrange interpolation.

4 Our T-LU-CP-ABE System

In this section we propose the construction of our new large universe CP-ABE system with white-box traceability.

4.1 Construction

- $\texttt{Setup}(1^\lambda) \rightarrow (pp, msk)$: The algorithm runs the group generator algorithm $\mathfrak{g}(1^\lambda)$ and gets the groups and the bilinear mapping description $GD = (p, \mathbb{G}, \mathbb{G}_T, e)$, where $(\mathbb{G}, \mathbb{G}_T)$ are groups of order p and e is the bilinear mapping. Let $U = \mathbb{Z}_p$ be the attribute universe. The algorithm randomly chooses $g, u, h, w, v \in \mathbb{G}$ and $\alpha, a \in \mathbb{Z}_p$. Besides, the algorithm chooses a probabilistic encryption scheme (Enc, Dec) [8] from a binary string to $\mathbb{Z}_p^*$ with different secret key $\bar{k_1}, \bar{k_2}$. Furthermore, it initializes an instance of Shamir's $(\bar{t}, \bar{n})$ threshold scheme $\mathcal{INS}_{(\bar{t},\bar{n})}$ [8] [35] and keeps $f(x)$ [9] and $\bar{t}-1$ points $\{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$ secret. It sets $(GD, g, u, h, w, v, g^a, e(g,g)^\alpha)$ as pp and $(\alpha, a, \bar{k_1}, \bar{k_2})$ as msk.
- $\texttt{KeyGen}(1^\lambda, pp, msk, id, S = \{A_1, A_2, ..., A_k\} \subseteq \mathbb{Z}_p) \rightarrow sk_{id,S}$: The algorithm computes: $x = Enc_{\bar{k_1}}(id), y = f(x), c = Enc_{\bar{k_2}}(x||y)$. Note that the computing result c is not distinguished from a random number [10]. And it randomly chooses $r, r_1, r_2, ...r_k \in \mathbb{Z}_p$. The decryption key $sk_{id,S}$ is set as follows:

$$\langle K = g^{\alpha/(a+c)} w^r, K' = c, L = g^r, L' = g^{ar},$$
$$\{K_{\tau,1} = g^{r_\tau}, K_{\tau,2} = (u^{A_\tau} h)^{r_\tau} v^{-(a+c)r}\}_{\tau \in [k]} \rangle$$

- $\texttt{Encrypt}(1^\lambda, pp, m \in \mathbb{G}_T, (M, \rho) \in (\mathbb{Z}_p^{l \times n}, \mathcal{F}([l] \rightarrow \mathbb{Z}_p))) \rightarrow ct$: The algorithm takes the public parameters pp, a plaintext message m and randomly chooses $\overrightarrow{y} = (s, y_2, ..., y_n)^\perp \in \mathbb{Z}_p^{n \times 1}$, where s is the random secret to be shared according to Subsection 3.3. It gets the vector of the shares $\overrightarrow{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_l)$ by computing the inner product $\lambda_i = M_i \overrightarrow{y}$, where M_i is the i-th row of M. Then it randomly picks l exponents $t_1, t_2, ..., t_l \in \mathbb{Z}_p$. The ciphertext ct is set as follows:

$$\langle (M, \rho), C = m \cdot e(g,g)^{\alpha s}, C_0 = g^s, C_0' = g^{as},$$
$$\{C_{i,1} = w^{\lambda_i} v^{t_i}, C_{i,2} = (u^{\rho(i)} h)^{-t_i}, C_{i,3} = g^{t_i}\}_{i \in [l]} \rangle$$

  It outputs the ciphertext ct.

[8] In our system, it requires $\bar{n}$ is greater than the number of the total users.

[9] If all of the users register and get the secret keys at the beginning of system initialization, the system could secretly store $f(0)$ instead of the polynomial $f(x)$ since the storage for $f(x)$ is much larger than that of $f(0)$.

[10] Due to the definition of probabilistic encryption, x is not distinguished from a random number. In addition, f is linear function and thus y is also a random number. Therefore, c, combined with x and y and through a probabilistic encryption, can also be a random number.

– $\texttt{Decrypt}(1^\lambda, pp, sk_{id,S}, ct) \to m$ or $\perp$: The algorithm first computes the set of rows in M that produces a share to attributes in S, that is, $I = \{i : \rho(i) \in S\}$. If the attribute set S is not an authorized set of the access policy, then it cannot satisfy the access structure of (M, ρ), the algorithm outputs $\perp$. Otherwise, the algorithm lets $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ be a set of constants such that $\sum_{i\in I} \omega_i M_i = (1, 0, ..., 0)$, where M_i is the matrix M's i-th row. Note that $\sum_{i\in I} \omega_i \lambda_i = s$ if the attribute set S is authorized, and there may exists other different ways to choose the values of ω_i to satisfy this. Then it computes:

$$E = e(K, C_0^{K'} C_0') = e(g, g)^{\alpha s} e(w, g)^{(a+c)sr}$$

$$D = \prod_{i\in I} (e(L^{K'} L', C_{i,1}) e(K_{\tau,1}, C_{i,2}) e(K_{\tau,2}, C_{i,3}))^{\omega_i} = e(g, w)^{(a+c)rs}$$

$$F = E/D = e(g, g)^{\alpha s}$$

where τ is the attribute $\rho(i)$'s index in S (it depends on i). It outputs the plaintext $m = C/F$.

Correctness:

$$F = \frac{E}{D} = \frac{e(g, g)^{\alpha s} e(w, g)^{(a+c)sr}}{\prod_{i\in I} D_1 \cdot D_2 \cdot D_3 \cdot D_4 \cdot D_5} = \frac{e(g, g)^{\alpha s} e(w, g)^{(a+c)sr}}{e(g, w)^{(a+c)r \sum_{i\in I} \omega_i \lambda_i}} = e(g, g)^{\alpha s}$$

where

$$D_1 = e(g, w)^{(a+c)r\lambda_i \omega_i},\ D_2 = e(g, v)^{(a+c)rt_i\omega_i},\ D_3 = e(g, u^{\rho(i)} h)^{-r_\tau t_i \omega_i},$$
$$D_4 = e(u^{\rho(i)} h, g)^{r_\tau t_i \omega_i},\ D_5 = e(v, g)^{-(a+c)rt_i\omega_i}.$$

– $\texttt{Trace}(pp, \mathcal{INS}_{(\bar{t},\bar{n})}, msk, sk) \to id$ or $\intercal$: If the sk is not in the form of $sk = (K, K', L, L', \{K_{\tau,1}, K_{\tau,2}\}_{\tau\in k})$ and can not pass the key sanity check, the algorithm will output $\intercal$. Otherwise, sk is a well-formed decryption key, and the algorithm will do as follows:

(1) The algorithm extracts $(x^* = x, y^* = y)$ from $x||y = Dec_{\bar{k_2}}(K')$ in sk.
(2) If $(x^* = x, y^* = y) \in \{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$, the algorithm computes $Dec_{\bar{k_1}}(x^*)$ to get id to identify the malicious user (with id). Otherwise, go to (3).
(3) The algorithm recovers the secret a_0^* of $\mathcal{INS}_{(\bar{t},\bar{n})}$ by interpolating with $\bar{t} - 1$ points $\{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$ and $(x^* = x, y^* = y)$. If $a_0^* = f(0)$, it computes $Dec_{\bar{k_1}}(x^*)$ to get id to find out the malicious user. Otherwise, the algorithm outputs $\intercal$.

Key Sanity Check: $K' \in \mathbb{Z}_p^*, K, L, L', K_{i,1}, K_{i,2} \in \mathbb{G}$ [11].

[11] Here only a brief check is needed.

4.2 Selective Security Proof

In the selective security proof, although we can proof that directly based on the Assumption 1 as [32] does, for simplicity, we will reduce the selective security of our T-LU-CP-ABE scheme to that of Rouselakis and Waters's scheme in [32] which is proved selectively secure under Assumption 1.

For simplicity, we denote by $\Sigma_{lucpabe}$, $\Sigma_{tlucpabe}$ the LU-CP-ABE scheme in [32] and our scheme respectively. Note that the security model of our scheme $\Sigma_{tlucpabe}$ is almost same with that of the scheme $\Sigma_{lucpabe}$ in [32], excepting every key query is companied with an explicit identity.

Lemma 1. *[32] If the assumption 1 holds, then the LU-CP-ABE scheme $\Sigma_{lucpabe}$ is selectively secure.*

Selective security of our new T-LU-CP-ABE:

Lemma 2. *If the LU-CP-ABE scheme $\Sigma_{lucpabe}$ is selectively secure in the game of [32], then our new T-LU-CP-ABE scheme $\Sigma_{tlucpabe}$ is selectively secure in the game of Subsection 2.2.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ with a challenge matrix M^* that has advantage $Adv_{\mathcal{A}}\Sigma_{tlucpabe}$ in selectively breaking our T-LU-CP-ABE scheme $\Sigma_{tlucpabe}$, where M^* is an $l \times n$ matrix satisfies the restriction that $l, n \leq q$. We construct a PPT algorithm $\mathcal{B}$ that has advantage $Adv_{\mathcal{B}}\Sigma_{lucpabe}$ in selectively breaking the underlying LU-CP-ABE scheme $\Sigma_{lucpabe}$, which equals to $Adv_{\mathcal{A}}\Sigma_{tlucpabe}$.

- **Initialization** : $\mathcal{B}$ gets a challenge policy (M^*, ρ^*) from $\mathcal{A}$ and then sends this received challenge policy to $\Sigma_{lucpabe}$. Note that M^* is an $l \times n$ matrix, where $l, n \leq q$, and $\rho^* \in \mathcal{F}([l] \to \mathbb{Z}_p)$.
- **Setup** : $\Sigma_{lucpabe}$ gives $\mathcal{B}$ the public parameter $pp_{\Sigma_{lucpabe}}$ as follows:

$$
\begin{array}{ll}
g = g & w = g^d \\
v = g^{\tilde{v}} \cdot \prod_{(j,k)\in[l,n]} (g^{d^k/b_j})^{M^*_{j,k}} & u = g^{\tilde{u}} \cdot \prod_{(j,k)\in[l,n]} (g^{d^k/b_j^2})^{M^*_{j,k}} \\
h = g^{\tilde{h}} \cdot \prod_{(j,k)\in[l,n]} (g^{d^k/b_j^2})^{-\rho^*(j)M^*_{j,k}} & e(g,g)^{\alpha} = e(g^d, g^{d^q}) \cdot e(g,g)^{\tilde{\alpha}}
\end{array}
$$

  Then $\mathcal{B}$ randomly chooses $a \in \mathbb{Z}_p$, adds g^a to $pp_{\Sigma_{lucpabe}}$ as a new public parameter pp and gives the new $pp = (D, g, u, h, w, v, g^a, e(g,g)^{\alpha})$ to $\mathcal{A}$. Finally, it initializes an instance of Shamir's $(\bar{t}, \bar{n})$ threshold scheme and keeps $f(x)$ and $\bar{t} - 1$ points $\{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$ secret.
- **Query Phase 1** : The adversary $\mathcal{A}$ will submit (id, S) to $\mathcal{B}$ to query a decryption key, then $\mathcal{B}$ submits S to $\Sigma_{lucpabe}$ and gets the corresponding

decryption key as follows:

$$\hat{K_0} = g^{\tilde{\alpha}}(g^d)^{\tilde{r}} \prod_{i=2}^{n}(g^{d^{q+2-i}})^{w_i}, \quad \hat{K_1} = g^{\tilde{r}} \prod_{i\in[n]}(g^{d^{q+1-i}})^{w_i}$$

$$\hat{K_{\tau,2}} = g^{\tilde{r_\tau}} \cdot \prod_{i'\in[l],\rho^*(i')\notin S}(g^{b_{i'}})^{\frac{\tilde{r}}{A_\tau-\rho^*(i')}} \cdot \prod_{(i,i')\in[n,l],\rho^*(i')\notin S}(g^{b_{i'}d^{q+1-i}})^{\frac{w_i}{A_\tau-\rho^*(i')}}$$

$$\hat{K_{\tau,3}} = \hat{\Psi}\cdot\hat{\Phi}\ , \quad where$$

$$\hat{\Psi} = (u^{A_\tau}h)^{\tilde{r}_\tau}\cdot(\hat{K_{\tau,2}}/g^{\tilde{r}_\tau})^{\tilde{u}A_\tau+\tilde{h}} \cdot \prod_{(i',\,j,\,k)\,\in\,[l,\,l,\,n],\,\rho^*(i')\,\notin\,S}(g^{\frac{b_{i'}d^k}{b_j^2}})^{\frac{\tilde{r}(A_\tau-\rho^*(j))M^*_{j,k}}{A_\tau-\rho^*(i')}}$$

$$\cdot \prod_{(i,\,i',\,j,\,k)\,\in\,[n,\,l,\,l,\,n]\ \rho^*(i')\,\notin\,S,\,(j\,\neq\,i'\,\vee\,i\,\neq\,k)}(g^{\frac{b_{i'}d^{q+1+k-i}}{b_j^2}})^{\frac{(A_\tau-\rho^*(j))w_iM^*_{j,k}}{A_\tau-\rho^*(i')}}$$

$$\hat{\Phi} = v^{-\tilde{r}}\prod_{i\in[n]}(g^{d^{q+1-i}})^{-\tilde{v}w_i} \cdot \prod_{(i,\,j,\,k)\,\in\,[n,\,l,\,n],\,i\,\neq\,k}(g^{\frac{d^{q+1+k-i}}{b_j}})^{-w_iM^*_{j,k}}$$

$\mathcal{B}$ computes $x = Enc^-_{k_1}(id), y = f(x), c = Enc^-_{k_2}(x||y)$ ($c \in \mathbb{Z}_p^*$) and $1/(a+c)$ modulo p. Then $\mathcal{B}$ sets $r = \tilde{r}/(a+c)$ and $K' = c$ implicitly and randomly chooses $g' \in \mathbb{G}$, then computes

$$K = (\hat{K_0})^{\frac{1}{a+c}} = g^{\frac{\tilde{\alpha}}{a+c}}((g^d)^{\tilde{r}}\prod_{i=2}^{n}(g^{d^{q+2-i}})^{w_i})^{\frac{1}{a+c}}$$

$$L = (\hat{K_1})^{\frac{1}{a+c}} = g^{\frac{\tilde{r}}{a+c}}(\prod_{i\in[n]}(g^{d^{q+1-i}})^{w_i})^{\frac{1}{a+c}} = g^r(\prod_{i\in[n]}(g^{d^{q+1-i}})^{w_i})^{\frac{1}{a+c}}$$

$$L' = (\hat{K_1})^{\frac{a}{a+c}}g' = g^{\frac{a\tilde{r}}{a+c}}(\prod_{i\in[n]}(g^{d^{q+1-i}})^{w_i})^{\frac{a}{a+c}}g' = g^{ar}(\prod_{i\in[n]}(g^{d^{q+1-i}})^{w_i})^{\frac{a}{a+c}}g'$$

$$K_{\tau,1} = \hat{K_{\tau,2}}, \quad K_{\tau,2} = \hat{K_{\tau,3}} = \hat{\Psi}\cdot\hat{\Phi}$$

$\mathcal{B}$ sends the decryption key $sk_{id,S} = \langle K, K', L, L', \{K_{\tau,1}, K_{\tau,2}\}_{\tau\in[|S|]}\rangle$ to $\mathcal{A}$. Note that g' makes L' uncorrelated to L.

– **Challenge** : The attacker $\mathcal{A}$ outputs two equal length messages (m_0, m_1) and sends them to $\mathcal{B}$. Then $\mathcal{B}$ submits (m_0, m_1) to $\Sigma_{lucpabe}$, and gets the challenge ciphertext as follows:

$$\langle(M^*,\rho^*),\ \hat{C} = m_\beta\cdot T\cdot e(g,g^s)^{\tilde{\alpha}},\ \hat{C_0} = g^s,\ \hat{C_{i,3}} = g^{t_i} = (g^{sb_i})^{-1},$$
$$\hat{C_{i,1}} = w^{\tilde{\lambda}_i}\cdot(g^{sb_i})^{-\tilde{v}}\cdot\prod_{(j,\,k)\,\in\,[l,\,n],\,j\,\neq\,i}(g^{sd^kb_i/b_j})^{-M^*_{j,k}},$$
$$\hat{C_{i,2}} = (g^{sb_i})^{-(\tilde{u}\rho^*(i)+\tilde{h})}\cdot\prod_{(j,\,k)\,\in\,[l,\,n],\,j\,\neq\,i}(g^{sd^kb_i/b_j^2})^{-(\rho^*(i)-\rho^*(j))M^*_{j,k}}\rangle$$

where T is the challenge term and g^s the corresponding term of the assumption in $\Sigma_{lucpabe}$.
Then $\mathcal{B}$ sets $C = \hat{C}, C_0 = \hat{C_0}, C_0' = (\hat{C_0})^a = g^{as}, C_{i,1} = \hat{C_{i,1}}, C_{i,2} = \hat{C_{i,2}}, C_{i,3} = \hat{C_{i,3}}$. Finally, $\mathcal{B}$ gives the challenge ciphertext $ct = \langle(M^*,\rho^*), C, C_0, C_0', \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i\in[l]}\rangle$ to $\mathcal{A}$.

– **Query Phase 2** : This phase is the same with Phase 1.

– **Guess** : The attacker outputs his guess β', and gives it to $\mathcal{B}$. Then $\mathcal{B}$ gives β' to $\Sigma_{lucpabe}$.
Since the distributions of the public parameters, decryption keys and challenge ciphertext in the above game are the same as that in the real scheme, we have $Adv_{\mathcal{B}}\Sigma_{lucpabe}$=$Adv_{\mathcal{A}}\Sigma_{tlucpabe}$.

Theorem 1. *If Assumption 1 holds, then our T-LU-CP-ABE scheme is selectively secure.*

Proof. It follows directly from Lemma 1 and Lemma 2.

4.3 Traceability Proof

In this subsection, we will give the traceability proof of our T-LU-CP-ABE scheme based on l-SDH assumption. We use a proof method from [4] and [24].

Theorem 2. *If the l-SDH assumption holds, then our T-LU-CP-ABE scheme is fully traceable provided that $q < l$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that has non-negligible advantage in winning the traceability game after making q key queries, w.l.o.g., assuming $l = q + 1$, we construct a PPT algorithm that has non-negligible advantage in breaking the l-SDH assumption. $\mathcal{B}$ is given an instance of l-SDH assumption problem as follows.

Let be $\mathbb{G}$ a bilinear group of order p and g be a generator of $\mathbb{G}$, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$ be a bilinear map, $a \in \mathbb{Z}_p^*$. $\mathcal{B}$ is given an instance $\mathcal{INS}_{SDH} = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, p, e, \bar{g}, \bar{g}^a, \bar{g}^{a^2}, ..., \bar{g}^{a^l})$. $\mathcal{B}$'s goal is to output a bit $\beta' \in \{0, 1\}$ to determine a tuple $(c_i, w_i) \in \mathbb{Z}_p^* \times \mathbb{G}$ satisfying $w_i = \bar{g}^{1/(a+c_i)}$ for solving the l-SDH problem. Before starting the traceability game with $\mathcal{A}$, $\mathcal{B}$ takes $(\mathbb{G}, \mathbb{G}_{\mathbb{T}}, p, e, \bar{g}, \bar{g}^a, \bar{g}^{a^2}, ..., \bar{g}^{a^l})$ as inputs and sets $B_i = \bar{g}^{a^i}$ for $i = 0, 1, ..., l$, then interacts with $\mathcal{A}$ in the traceability game as follows:

– **Setup** : $\mathcal{B}$ randomly chooses q distinct values $c_1, ..., c_q \in \mathbb{Z}_p^*$. Let $f(y)$ be the polynomial $f(y) = \prod_{i=1}^{q}(y + c_i)$. Expand $f(y)$ and write $f(y) = \sum_{i=0}^{q} \alpha_i y^i$. Where $\alpha_0, ..., \alpha_q \in \mathbb{Z}_p$ are the coefficients of the polynomial $f(y)$. Then $\mathcal{B}$ computes $g \leftarrow \prod_{i=0}^{q}(B_i)^{\alpha_i} = \bar{g}^{f(a)}$ and $g^a \leftarrow \prod_{i=1}^{q+1}(B_i)^{\alpha_{i-1}} = \bar{g}^{f(a)\cdot a}$. $\mathcal{B}$ randomly chooses $\alpha, \theta \in \mathbb{Z}_p$ and $u, h, v \in \mathbb{G}$, and then gives $\mathcal{A}$ the public parameter $pp = (GD, g, u, h, w = g^\theta, v, g^a, e(g, g)^\alpha)$. Also, it initializes an instance of Shamir's $(\bar{t}, \bar{n})$ threshold scheme and keeps $f(x)$ and $\bar{t} - 1$ points $\{(x_1, y_1), (x_2, y_2), ..., (x_{\bar{t}-1}, y_{\bar{t}-1})\}$ secret.
– **Key Query** : $\mathcal{A}$ submits (id_i, S_i) to $\mathcal{B}$ to query a decryption key. When it comes to i-th query, we assume $i \leq q$. Let $f_i(y)$ be the polynomial $f_i(y) = f(y)/(y+c_i) = \prod_{j=1, j\neq i}^{q}(y+c_j)$. Expand $f_i(y)$ and write $f_i(y) = \sum_{j=0}^{q-1} \beta_j y^j$. $\mathcal{B}$ computes $\sigma_i \leftarrow \prod_{j=0}^{q-1}(B_j)^{\beta_j} = \bar{g}^{f_i(a)} = \bar{g}^{f(a)/(a+c_i)} = g^{1/(a+c_i)}$.
$\mathcal{B}$ chooses $r, r_1, ..., r_k \in \mathbb{Z}_p$ randomly, then gives $\mathcal{A}$ the sk_{id_i,S_i} as follows:

$$\langle K = (\sigma_i)^\alpha w^r = g^{\alpha/(a+c_i)} w^r, K' = c_i, L = g^r, L' = g^{ar},$$
$$\{K_{\tau,1} = g^{r_\tau}, K_{\tau,2} = (u^{A_\tau} h)^{r_\tau} (v^a \cdot v^{c_i})^{-r} = (u^{A_\tau} h)^{r_\tau} v^{-(a+c_i)r}\}_{\tau\in[k]}\rangle$$

– **Key Forgery** : $\mathcal{A}$ submits a decryption key sk_* to $\mathcal{B}$. Note that the distributions of public parameters and decryption keys in the above game are the same as that in the real scheme. Let $\epsilon_{\mathcal{A}}$ denote the event that $\mathcal{A}$ wins the game, i.e. sk_* is in the form of $sk_* = (K, K', L, L', \{K_{\tau,1}, K_{\tau,2}\}_{\tau\in[|S_*|]})$ and satisfies the key sanity check and $K' \neq \{c_1, .., c_q\}$. If $\epsilon_{\mathcal{A}}$ does not happen, $\mathcal{B}$ chooses a random bit $\beta' \in \{0, 1\}$ and a tuple $(c_s, w_s) \in \mathbb{Z}_p \times \mathbb{G}$ as the solution to l-SDH problem. If $\epsilon_{\mathcal{A}}$ happens, using long division $\mathcal{B}$ writes the polynomial f as $f(y) = \gamma(y)(y + K') + \gamma_{-1}$ for some polynomial $\gamma(y) = \sum_{i=0}^{q-1}(\gamma_i y^i)$ and some $\gamma_{-1} \in \mathbb{Z}_p$. Note that $\gamma_{-1} \neq 0$, since $f(y) = \prod_{i=1}^{q}(y + c_i), c_i \in \mathbb{Z}_p^*$ and $K' \neq \{c_1, .., c_q\}$, as thus $y + K'$ does not divide $f(y)$. Then $\mathcal{B}$ computes $gcd(\gamma_{-1}, p)$. And $\mathcal{B}$ computes a tuple $(c_s, w_s) \in \mathbb{Z}_p \times \mathbb{G}$ as follows:

Assuming $L = g^r$ where $r \in \mathbb{Z}_p$ is unknown, we have $L' = g^{ar}, K = g^{\alpha/(a+K')}w^r$. Then $\mathcal{B}$ computes $1/\gamma_{-1} \pmod p$ (since $gcd(\gamma_{-1}, p) = 1$) and then does following computation:

$$\sigma \leftarrow (K/L^\theta)^{\alpha^{-1}} = g^{1/(a+K')} = \bar{g}^{f(a)/(a+K')} = \bar{g}^{\gamma(a)}\bar{g}^{\frac{\gamma_{-1}}{(a+K')}}$$
$$w_s \leftarrow (\sigma \cdot \prod_{i=0}^{q-1} B_i^{-\gamma_i})^{1/\gamma_{-1}} = \bar{g}^{1/(a+K')}, \quad c_s \leftarrow K' \mod p \in \mathbb{Z}_p$$

Note that $e(\bar{g}^a \cdot \bar{g}^{c_s}, w_s) = e(\bar{g}^a \cdot \bar{g}^{K'}, \bar{g}^{1/(a+K')}) = e(\bar{g}, \bar{g})$. Therefore (c_s, w_s) is a solution for the l-SDH problem.

Now we evaluate the advantage of $\mathcal{B}$ in breaking l-SDH assumption.

Let $\epsilon_{SDH}(c_s, w_s)$ denote the event that (c_s, w_s) is a solution for the l-SDH problem, which can be verified by checking whether $e(\bar{g}^a \cdot \bar{g}^{c_s}, w_s) = e(\bar{g}, \bar{g})$ holds. Note that when $\mathcal{B}$ randomly chooses (c_s, w_s), $\epsilon_{SDH}(c_s, w_s)$ happens with negligible probability, say zero for simplicity. And the probability of (c_s, w_s) satisfies $e(\bar{g}^a \cdot \bar{g}^{c_s}, w_s) = e(\bar{g}, \bar{g})$ is 1 in the case of ($\mathcal{A}$ *win* $\wedge$ $gcd(\gamma_{-1}, p) = 1$) when $\mathcal{B}$ outputs (c_s, w_s).

So $\mathcal{B}$ solves the l-SDH problem with probability

$$\begin{aligned}
&Pr[\epsilon_{SDH}(c_s, w_s)] \\
&= Pr[\epsilon_{SDH}(c_s, w_s)|\overline{\mathcal{A}\ win}] \cdot Pr[\overline{\mathcal{A}\ win}] + \\
&\quad Pr[\epsilon_{SDH}(c_s, w_s)|\mathcal{A}\ win \wedge gcd(\gamma_{-1}, p) \neq 1] \cdot Pr[\mathcal{A}\ win \wedge gcd(\gamma_{-1}, p) \neq 1] + \\
&\quad Pr[\epsilon_{SDH}(c_s, w_s)|\mathcal{A}\ win \wedge gcd(\gamma_{-1}, p) = 1] \cdot Pr[\mathcal{A}\ win \wedge gcd(\gamma_{-1}, p) = 1] \\
&= 0 + 0 + 1 \cdot Pr[\mathcal{A}\ win \wedge gcd(\gamma_{-1}, p) = 1] = Pr[\mathcal{A}\ win \wedge gcd(\gamma_{-1}, p) = 1] \\
&= Pr[\mathcal{A}\ win] \cdot Pr[gcd(\gamma_{-1}, p) = 1] = \tfrac{\epsilon_{\mathcal{A}}}{2}
\end{aligned}$$

$$Pr[\beta' = \beta] = Pr[\beta' = \beta|\epsilon_{SDH}(c_s, w_s)] \cdot Pr[\epsilon_{SDH}(c_s, w_s)] = \tfrac{1}{2} \cdot \tfrac{\epsilon_{\mathcal{A}}}{2} = \tfrac{\epsilon_{\mathcal{A}}}{4}$$

Therefore $\mathcal{B}$ can solve the l-SDH problem with non-negligible advantage, which conflicts with the l-SDH assumption.

5 Extensions

5.1 Transform from One-Use T-LU-CP-ABE to Multi-Use T-LU-CP-ABE

The construction in our system is a *one-use* T-LU-CP-ABE construction. Since the ρ in our system is an injective function for each access policy associated to a

ciphertext. During the row label of the share-generating matrix, the attributes are only used once. This kind of construction is called one-use CP-ABE.

We can extend our new T-LU-CP-ABE system to a *multi-use* system using the encoding technique in [16]: we take k copies of each attribute A instead of a single attribute. Then we have new "attributes": $\{A:1,...,A:k\}$. Now we can label a row of the access matrix $\mathbb{A}$ with $\{A:i\}$. Thus the attribute can be used multiple times. Note that the size of the public parameters do not grow linearly with the number of the involved attributes, so that the size of the public parameters will remain the same size under this transformation. Besides the access matrix's size does not change under this transformation either, thus the size of the ciphertext also remains the same size. This makes our T-LU-CP-ABE system more suitable for commercial applications.

5.2 Revocable T-LU-CP-ABE

Through our new T-LU-CP-ABE system proposed in this paper, it is easy to trace the malicious user who leak his/her decryption key for benefits. This evokes another significant issue to be considered: how to revoke the malicious users. Several work has focused on designing revocable ABE [31, 33]. With the technology of ciphertext delegation and piecewise key generation introduced in [33], we can achieve a revocable T-LU-CP-ABE construction. Furthermore, since we make use of the Shamir's $(\bar{t}, \bar{n})$ threshold scheme in the `Trace` algorithm, the system only need store $\bar{t}-1$ tuples in system for tracing, rather than an identify table T which contains all users' identifies. This brings an obvious advantage that the system need not update the identify table T when some users are revoked.

6 Conclusion and Future Work

In this work, we have presented a practical large universe CP-ABE system supporting white-box traceability. Specifically, we have achieved the property of white-box traceability in CP-ABE, which could trace the malicious users leaking the partial or modified decryption keys to others for profits. We have also obtained the property of large universe in white-box traceable CP-ABE where the attributes' size is unbounded and the public parameters' size does not grow linearly with the number of attributes. In addition, we optimize the system in tracing the malicious users to cut down the storage cost for traceability and to make the system efficient in the user revocation. Based on the above advantages, our new system could be applied to many scenarios such as pay-TV systems and social networks. As far as we known, this is the first practical CP-ABE system that simultaneously supports white-box traceability and large universe. We have also proved our new system selectively secure in the standard model.

In our future work, we will focus on the stronger notion for traceability named black-box traceability. In that scenario, the malicious users leak their decryption devices instead of decryption keys. Specifically, the malicious users could hide the decryption algorithm by tweaking it, as well as the decryption keys. In this

case, due to the fact that the decryption keys and decryption algorithm are both not well-formed, the new system supporting white-box traceability in this paper will fail. It will be our future work to obtain a large universe CP-ABE system, which supports black-box traceability.

There is another important issue about public auditing we need to pay attention to. Suppose a user Bob is identified as a malicious user by the system, but claims to be innocent and framed by the system. It is a big problem to judge whether Bob is in fact innocent or not. In this case, the suspected user does not trust the system and the system needs to provide some evidence persuasive enough to prove that the suspected user is guilty. To address this issue, a public auditor which is played by a trusted third party needs to be introduced. However, the suspected user does not want the public auditor to know the private information since in the `Trace` phase the auditor will obtain Bob's decryption keys and be able to decrypt all the data that Bob has. Achieving a traceable large universe CP-ABE system with public auditors is still an open problem, and we will keep working on it.

7 Acknowledgements

We are grateful to the anonymous reviewers for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China (Grant No. 61371083, 61373154 and 61033014), the Prioritized Development Projects of the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20130073130004) and the Natural Science Foundation of Shanghai of Yang-Fan Plan (Grant No. 14YF1410400).

References

1. Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography–PKC 2011*, pages 90–108. Springer, 2011.
2. Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
3. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE, 2007.
4. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.
5. Melissa Chase. Multi-authority attribute based encryption. In *Theory of Cryptography*, pages 515–534. Springer, 2007.
6. Melissa Chase and Sherman SM Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 121–130. ACM, 2009.
7. Ling Cheung and Calvin Newport. Provably secure ciphertext policy abe. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 456–465. ACM, 2007.

8. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.
9. Vipul Goyal. Reducing trust in the pkg in identity based cryptosystems. In *Advances in Cryptology-CRYPTO 2007*, pages 430–447. Springer, 2007.
10. Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *Automata, Languages and Programming*, pages 579–591. Springer, 2008.
11. Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 427–436. ACM, 2008.
12. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. ACM, 2006.
13. Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium*, page 3, 2011.
14. Susan Hohenberger and Brent Waters. Attribute-based encryption with fast decryption. In *Public-Key Cryptography–PKC 2013*, pages 162–179. Springer, 2013.
15. Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Advances in Cryptology–EUROCRYPT 2012*, pages 318–335. Springer, 2012.
16. Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology–EUROCRYPT 2010*, pages 62–91. Springer, 2010.
17. Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *Theory of Cryptography*, pages 455–479. Springer, 2010.
18. Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 568–588. Springer, 2011.
19. Allison Lewko and Brent Waters. Unbounded hibe and attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 547–567. Springer, 2011.
20. Allison Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Advances in Cryptology–CRYPTO 2012*, pages 180–198. Springer, 2012.
21. Jin Li, Qiong Huang, Xiaofeng Chen, Sherman SM Chow, Duncan S Wong, and Dongqing Xie. Multi-authority ciphertext-policy attribute-based encryption with accountability. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 386–390. ACM, 2011.
22. Jin Li, Kui Ren, and Kwangjo Kim. A2be: Accountable attribute-based encryption for abuse free access control. *IACR Cryptology ePrint Archive*, 2009:118, 2009.
23. Zhen Liu, Zhenfu Cao, and Duncan S Wong. Blackbox traceable cp-abe: how to catch people leaking their keys by selling decryption devices on ebay. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 475–486. ACM, 2013.
24. Zhen Liu, Zhenfu Cao, and Duncan S. Wong. White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. *IEEE Transactions on Information Forensics and Security*, 8(1):76–88, 2013.
25. Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing-Based Cryptography–Pairing 2008*, pages 57–74. Springer, 2008.

26. Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *Advances in Cryptology–ASIACRYPT 2009*, pages 214–231. Springer, 2009.
27. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Advances in Cryptology–CRYPTO 2010*, pages 191–208. Springer, 2010.
28. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Advances in Cryptology–CRYPTO 2010*, pages 191–208. Springer, 2010.
29. Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203. ACM, 2007.
30. Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography*, pages 422–439. Springer, 2012.
31. Jun-lei Qian and Xiao-lei Dong. Fully secure revocable attribute-based encryption. *Journal of Shanghai Jiaotong University (Science)*, 16:490–496, 2011.
32. Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 463–474. ACM, 2013.
33. Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 199–217. Springer, 2012.
34. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
35. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
36. Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography–PKC 2011*, pages 53–70. Springer, 2011.