Accountable Large-Universe Attribute-Based Encryption Supporting Any Monotone Access Structures

Yinghui Zhang[1,2(✉)], Jin Li[3], Dong Zheng[1], Xiaofeng Chen[4], and Hui Li[4]

[1] National Engineering Laboratory for Wireless Security,
Xi'an University of Posts and Telecommunications,
Xi'an 710121, People's Republic of China
yhzhaang@163.com
[2] State Key Laboratory of Cryptology,
P.O. Box 5159, Beijing 100878, People's Republic of China
[3] School of Computer Science, Guangzhou University,
Guangzhou 510006, People's Republic of China
[4] State Key Laboratory of Integrated Service Networks (ISN),
Xidian University, Xi'an 710071, People's Republic of China

Abstract. Ciphertext-policy attribute-based encryption (CP-ABE) is a promising cryptographic primitive for fine-grained access control on data outsourced to clouds. However, there still exists one critical functionality missing in existing CP-ABE schemes, which is the prevention of key abuse. Specifically, two kinds of key abuse problems are considered in this paper: malicious key sharing among colluding users, and key escrow problem of the semi-trusted authority. For a user, any malicious behavior including illegal key sharing should be traced. For the semi-trusted authority, it should be accountable for its misbehavior including illegal key re-distribution. For better performance and security, it is also indispensable to support large universe and full security in CP-ABE. To the best of our knowledge, none of the existing traceable CP-ABE schemes simultaneously supports large universe and full security. In this paper, we construct a white-box traceable CP-ABE scheme with weak public user traceability, weak public authority accountability and weak public auditing in the sense that no additional secret keys are needed. The scheme supports large universe, and attributes do not need to be pre-specified during the system setup phase. Our scheme is proven fully-secure in the random oracle model and it can take any monotonic access structures as ciphertext policies.

Keywords: Attribute-based encryption · User traceability · Authority accountability · Large universe · Full security · Weak public traceability

1 Introduction

Attribute-based encryption (ABE) is very promising in implementing flexible access control on the data outsourced to clouds. The notion of ABE was

J.K. Liu and R. Steinfeld (Eds.): ACISP 2016, Part I, LNCS 9722, pp. 509–524, 2016.
DOI: 10.1007/978-3-319-40253-6_31

introduced by Sahai and Waters [25] as a fuzzy version of identity-based encryption (IBE). Goyal *et al.* [8] further extended this idea and defined two complementary notions of ABE: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE).

A series of ABE work has been done to achieve better performance and security. In particular, large universe and full security are two significant and practical properties of ABE, which have attracted much attention in the research community. Lewko *et al.* [11] took the large universe issue into account and classified ABE into two flavors: the small universe ABE (SU-ABE) and the large universe ABE (LU-ABE). In SU-ABE constructions, attributes are fixed at the system setup phase. Furthermore, the system public parameters often depend on the amount of attributes in the system, and hence the scale of the attribute universe is polynomially bounded in the security parameter. In the case of LU-ABE, the attribute universe scale can be exponentially large. Obviously, LU-ABE is more practical than SU-ABE in that the system designer needs not to choose a bound of attributes at system setup. However, many prior schemes have to rely on a weaker security model known as selective security, where the attacker must specify some challenge ciphertext information before seeing the system public parameters. As a more stronger security model, full security [10] allows the attacker to adaptively choose challenge targets based on system public parameters.

Nevertheless, as a major issue remain to be solved, user traceability and authority accountability have severely limited the applications of ABE [10,11]. In a CP-ABE system, for example, the decryption keys are issued by an attribute authority based on users' attributes, which are usually shared by multiple users and hence are not uniquely linked to users' identification information. Obviously, as the foundation of one-to-many encryption mechanism, the characteristic of attribute sharing introduces the malicious user traceability issue: an explicitly leaked decryption key is non-traceable because the underlying attributes are shared by multiple users. On the other hand, the attribute authority is capable of re-distributing decryption keys for any user without any risk of being caught. So, if the attribute authority is not fully-trusted, it is indispensable to provide a method to make the authority accountable. In the above description, any user and the authority who exactly leak a decryption key to the third user intentionally or unintentionally will be identified, which is called white-box traceability. As a relatively stronger notion, black-box traceability can trace the malicious users and authority even if they only leak a decryption equipment instead of the decryption key.

There are some ABE solutions [7,13,15,16,18–21,30] for the purpose of user traceability and authority accountability. In these schemes, white-box user traceability [13,15,20,21,30] and black-box user accountability [7,16,18,19] are considered. Meanwhile, only schemes [15,16,18,21] achieve full security and schemes [7,19,20,30] support large universe. In particular, only the solutions [13,21,30] take one step further towards *authority accountability* although in the white-box model. As one of the latest work, the scheme [21] allows tracing and weak public

Table 1. Features comparison between authority accountable CP-ABE schemes

Schemes	Security	MAS	Large universe	Weak public UT	Weak public AA	Weak Public Auditing
[13]	selective (random)	×	×	√	√	–
[30]	selective (standard)	√	×	×	×	–
[21]	full (standard)	√	×	×	×	√
Ours	full (random)	√	√	√	√	√

auditing in the case of almost no storage. However, it is a small universe construction and has a security weakness shown later. In summary, it is necessary to efficiently add both user traceability and authority accountability to the original ABE while keeping the properties of large universe and full security.

Our Contribution. In this paper, we address the key abuse problems of CP-ABE while keeping desirable performance and security. The main contributions can be summarized as follows:

- We propose an authority accountable large universe CP-ABE scheme (AA-LU-CPABE) that simultaneously supports (1) weak public user traceability, (2) weak public authority accountability, (3) weak public auditing, (4) large universe, and (5) full security. The expression "weak public"[1] means that both traceability and auditing only involve decryption keys, which are indispensable in the white-box model, and no additional secret parameters such as master secret keys and identity tables are needed.
- For realizing user traceability, when a user queries for decryption key, his/her identity is inserted into a decryption key component, which is further implicitly signed as a fixed component such that the key owner is not able to re-randomize it. To achieve authority accountability, a user's decryption key is generated by the user himself based on a primary decryption key, which is jointly determined by both the authority and the user.
- The AA-LU-CPABE scheme needs almost no storage for tracing in that it does not need to maintain an identity table of users for tracing. In addition, our scheme is proven secure in the random oracle model against adaptive adversaries, and is highly expressive and can take any monotonic access structures as ciphertext policies.

Note that only schemes [13,21,30] support *authority accountability*. We compare our work with schemes [13,21,30] in Table 1, where MAS, UT and AA mean monotonic access structures, user traceability and authority accountability, respectively. The symbol "–" represents the corresponding scheme does not need auditing. All the schemes are realized in the white-box model. It's noted that only the proposed scheme simultaneously supports weak public traceability and auditing with large universe and full security.

[1] The expression "weak public" is similar to the term "partial public" in [20], in which only private user traceability is realized.

Related Work. Since the introduction of ABE [25], a plenty of researches have been done on flexible ABE schemes. The first CP-ABE scheme was proposed by Bethencourt *et al.* [4], which is proven secure in the generic group model. To improve the security proof, Cheung and Newport [6] proposed another CP-ABE construction and proved its security in the standard model. The construction supports the access structures of AND gate on different attributes. In order to further protect users' attribute privacy, anonymous CP-ABE has been studied [9,22,33]. A series of CP-ABE schemes have been proposed for better expressiveness, security and efficiency [1,27,29,31,32,34]. In particular, large universe, full security and traceability are important aspects to be considered.

The first large universe KP-ABE construction was given in [11], which achieves selective security under static assumptions in the standard model. They utilized the dual system framework on composite order groups to prove security. The first large universe CP-ABE scheme was proposed in [24], which is selectively secure under two q-type assumptions in the standard model. By utilizing dual vector spaces, Okamoto and Takashima [23] proposed the first fully secure unbounded CP-ABE scheme in the standard model. Lewko *et al.* [10] constructed a fully-secure CP-ABE scheme in the standard model, however, it fails to support large universe. Li *et al.* [13,14] first introduced the notion of accountable CP-ABE. The scheme [13] takes into account authority accountability which is achieved by embedding additional user-specific information into the attribute decryption key. Yu *et al.* [28] considered how to defend the key-abuse problem in KP-ABE. A user traceable multi-authority CP-ABE scheme was proposed in [12]. For the purpose of expressiveness, Liu *et al.* proposed white-box [15] and black-box [16–18] traceable CP-ABE schemes. These schemes cannot support large universe even if they are fully-secure in the standard model. Large universe CP-ABE schemes with user accountability were proposed in the white-box model in [20] and in the black-box model in [7,19], which are proven selectively-secure. Most of the above schemes fail to realize authority accountability while keeping expressive polices. For the sake of practicality, the solutions [21,30] take one step further towards authority accountability although in the white-box model. However, the scheme [30] is selectively-secure and the scheme [21] is a small universe construction. In general, if a system requires user traceability, authority accountability, (weak public) auditing and full security, only the scheme [21] may be adopted. Unfortunately, we found that the scheme [21] is not secure. In fact, after receiving from the authority c and the primary decryption key $SK_{pri} = \langle \overline{K}, \overline{T}, \overline{L}, \overline{L'}, \{\overline{K_i}\}_{i \in S} \rangle$, a user just sets $t_{id} = \frac{c}{t}$, where t is chosen by himself and $R_u = g^t$, and generates the final decryption key $SK_{id,S} = \langle K = \overline{K}(g^{\mu})^{t_{id}}, T = \overline{T}, L = \overline{L}, L' = \overline{L'}, R_u, t_{id}, \{K_i = \overline{K_i}\}_{i \in S} \rangle$, where g^{μ} is a public parameter. Then, the user randomly chooses a value c_0 and is able to re-randomize $SK_{id,S}$ based on the idea of changing c to $c \cdot c_0$. In this paper, to avoid the above security weakness, the attribute authority chooses two secrets c_0 and $\bar{c}$ and only $\bar{c}$ is sent to the user. Most importantly, c_0 and $\bar{c}$ are used to generate different components of a decryption key during the key generation phase. The details can be found in Sect. 4.1.

Organization. The remaining of this work is organized as follows. Some preliminaries are reviewed in Sect. 2. We then present the formal definition and security models for AA-LU-CPABE in Sect. 3. In Sect. 4, the proposed AA-LU-CPABE construction together with its security results are described. Finally, we conclude this paper in Sect. 5.

2 Preliminaries

Throughout this paper, for $\ell \in \mathbb{N}$, we denote by $[\ell]$ the set $\{1, 2, \cdots, \ell\}$. For a set S, $|S|$ represents its cardinality, and $s \in_R S$ means the variable s is chosen uniformly at random from S.

2.1 Cryptographic Background

Definition 1. (Composite Order Bilinear Groups). *Composite order bilinear groups are widely used in IBE and ABE systems, which are first introduced in [5] We denote by $\mathcal{G}$ a group generator, which takes a security parameter λ as inputs and outputs a description of a bilinear group $\mathbb{G}$. We define the output of $\mathcal{G}$ as $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, \hat{e})$, where p_1, p_2, p_3 are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $N = p_1p_2p_3$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map satisfying: (1) Bilinear: $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$ for all $a, b \in \mathbb{Z}_N$ and $g, h \in \mathbb{G}$, (2) Non-degenerate: There exists $g \in \mathbb{G}$ such that $\hat{e}(g, g)$ has order N in $\mathbb{G}_T$.*

Assume that group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map $\hat{e}$ are computable in polynomial time with respect to λ. Let $\mathbb{G}_{p_i}$ be the subgroup of order p_i in $\mathbb{G}$ for $1 \leq i \leq 3$. Note that for any $X_i \in \mathbb{G}_{p_i}$ and $X_j \in \mathbb{G}_{p_j}$, $\hat{e}(X_i, X_j) = 1$ holds for $i \neq j$.

Assumption 1. *(Subgroup Decision Problem for 3 Primes): Given a group generator $\mathcal{G}$, define the following distribution:*

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}, g \xleftarrow{R} \mathbb{G}_{p_1}, X_3 \xleftarrow{R} \mathbb{G}_{p_3},$$
$$D = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, X_3), T_1 \xleftarrow{R} \mathbb{G}_{p_1p_2}, T_2 \xleftarrow{R} \mathbb{G}_{p_1}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is defined to be: $\mathsf{Adv1}_{\mathcal{G},\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

Definition 2. *We say that $\mathcal{G}$ satisfies Assumption 1 if $\mathsf{Adv1}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 2. *Given a group generator $\mathcal{G}$, define the following distribution:*

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}, g, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, X_2, Y_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3, Y_3 \xleftarrow{R} \mathbb{G}_{p_3},$$
$$D = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, X_1X_2, X_3, Y_2Y_3), T_1 \xleftarrow{R} \mathbb{G}, T_2 \xleftarrow{R} \mathbb{G}_{p_1p_3}.$$

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is defined to be: $\mathsf{Adv2}_{\mathcal{G},\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

Definition 3. *We say that $\mathcal{G}$ satisfies Assumption 2 if $\mathsf{Adv2}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 3. *Given a group generator $\mathcal{G}$, define the following distribution:*

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, \hat{e}) \xleftarrow{R} \mathcal{G}, \alpha, s \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} \mathbb{G}_{p_1}, X_2, Y_2, Z_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3 \xleftarrow{R} \mathbb{G}_{p_3},$$
$$D = (N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g, g^{\alpha}X_2, X_3, g^{s}Y_2, Z_2), T_1 = \hat{e}(g,g)^{\alpha s}, T_2 \xleftarrow{R} \mathbb{G}_T.$$

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is defined to be: $\mathsf{Adv3}_{\mathcal{G},\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

Definition 4. *We say that $\mathcal{G}$ satisfies Assumption 3 if $\mathsf{Adv3}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

***ℓ*-SDH assumption** *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 $\ell+1$-tuple $(g, g^x, g^{x^2}, \ldots, g^{x^\ell})$ 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 problem in $\mathbb{G}$ if $\Pr[\mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^\ell}) = (c, g^{1/(c+x)}] \geq \epsilon$, where the probability is over the random choice of x in $\mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$.*

Definition 5. *We say that (ℓ, t, ϵ)-SDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the ℓ-SDH problem in $\mathbb{G}$.*

2.2 Zero-Knowledge Proof of Knowledge of Discrete Log

A zero-knowledge proof of knowledge (ZK-PoK) of discrete log protocol that enables a prover to prove to a verifier that it possesses the discrete log of a given group element in question. Efficient ZK-PoK of discrete log protocols can be found in [26]. A ZK-PoK protocol has the proof of knowledge property besides the zero-knowledge property. The property of zero-knowledge implies that there exists a simulator which is able to simulate the view of a verifier in the protocol without being given the witness as inputs. The proof of knowledge property implies the existence of a knowledge-extractor which interacts with the prover and extracts the witness using rewinding techniques [3].

2.3 Access Policy

Definition 6 (Access Structures [2]). *Let $\mathcal{U}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\mathcal{U}}$ is monotone if $\forall B \in \mathbb{A}$ and $C \in 2^{\mathcal{U}}$: if $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (resp. monotone access structure) on $\mathcal{U}$ is a collection (resp. monotone collection) $\mathbb{A}$ of non-empty subsets of $\mathcal{U}$, i.e., $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, otherwise, the sets are called the unauthorized sets.*

Definition 7 (Linear Secret Sharing Schemes (LSSS) [2]). *Let $\mathcal{U}$ be the attribute universe and $\mathbb{A}$ an access structure on $\mathcal{U}$. An LSSS can be used to represent an access structure $\mathbb{A} = (M, \rho)$, where M is an $\ell \times n$ matrix which is called the share-generating matrix and ρ maps a row of M into an attribute. An LSSS consists of two algorithms of secret sharing and reconstruction as below.*

- ***Share***$((M,\rho),s)$*: This algorithm is used to share a secret value* s *based on attributes. Considering a vector* $\boldsymbol{v}=(s,y_2,...,y_n)^T$*, where* $s\in\mathbb{Z}_p$ *is the secret to be shared and* $y_2,...,y_n\in_R\mathbb{Z}_p$*, then* $\lambda_i=M_i\cdot\boldsymbol{v}$ *is a share of the secret* s *which belongs to the attribute* $\rho(i)$*, where* M_i *is the* i*-th row of* M*.*
- ***Reconstruction***$(\lambda_1,...,\lambda_\ell,(M,\rho))$*: This algorithm is used to reconstruct* s *from secret shares. Let* $S\in\mathbb{A}$ *be any authorized set and* $I=\{i|\rho(i)\in S\}\subseteq\{1,2,...,\ell\}$*. Then there exists coefficients* $\{\omega_i\}_{i\in I}$ *such that* $\sum_{i\in I}\omega_i M_i=(1,0,...,0)$*, thus we have* $\sum_{i\in I}\omega_i\lambda_i=s$*.*

3 Formal Definition and Security Model

3.1 Formal Definition of AA-LU-CPABE

An AA-LU-CPABE scheme consists of six algorithms **Setup**, **KeyGen**, **Encrypt**, **Decrypt**, $\mathbf{Trace}_{\mathrm{wp}}$ and $\mathbf{Judge}_{\mathrm{wp}}$. They are detailed as follows:

- $\mathbf{Setup}(1^\lambda)\rightarrow(PK,MK)$: The setup algorithm is run by the attribute authority. On input a security parameter λ, it outputs the system public key PK and the master key MK.
- $\mathbf{KeyGen}(PK,MK,ID,S)\rightarrow SK_{ID,S}$: This is an interactive protocol between the authority and a user with an identity ID and a set of attributes S. The system public key PK and (ID,S) are the common inputs to the authority and the user. The master key MK is the private input to the authority. At the end of the protocol, a decryption key $SK_{ID,S}$ corresponding to (ID,S) is finally generated by the user based on a primary decryption key which is determined jointly by the authority and the user. Note that only S is implicitly included in $SK_{ID,S}$.
- $\mathbf{Encrypt}(PK,m,(M,\rho))\rightarrow CT$: On input the system public key PK, a message m and an access policy (M,ρ) specified by the encryptor, it generates a ciphertext CT as the encryption of m with respect to (M,ρ). Note that (M,ρ) is implicitly included in CT.
- $\mathbf{Decrypt}(PK,CT,SK_{ID,S})\rightarrow m$ or $\perp$: On input the system public key PK, a ciphertext CT of a message m under (M,ρ), and a decryption key $SK_{ID,S}$ associated with (ID,S), it outputs the message m if S satisfies (M,ρ), and the error symbol $\perp$ otherwise.
- $\mathbf{Trace}_{\mathrm{wp}}(PK,SK_{ID,S})\rightarrow ID$ or $\top$: On input the system public key PK and a decryption key $SK_{ID,S}$, the tracing algorithm first checks whether $SK_{ID,S}$ is well-formed or not. If $SK_{ID,S}$ is well-formed, it extracts the identity ID from $SK_{ID,S}$ and outputs ID to indicate that $SK_{ID,S}$ is linked to ID. Otherwise, it outputs a special symbol $\top$ to indicate that $SK_{ID,S}$ does not need to be traced. A decryption key is well-formed means that it passes a "key sanity check" which guarantees that the decryption key can be used in the well-formed decryption process.
- $\mathbf{Judge}_{\mathrm{wp}}(PK,SK_{ID,S},SK^*_{ID,S})\rightarrow$ guilty or innocent: This is an interactive protocol between a user (ID,S) with a decryption key $SK_{ID,S}$ and a public auditor. When the user is identified as a malicious user by the system based

on the traced key $SK^*_{ID,S}$, the auditor judges whether the user is guilty or innocent upon receiving $SK_{ID,S}$ from the user.

Remark 1. The tracing and judge algorithms need the decryption key $SK_{ID,S}$, which means the white-box model. However, no additional secret parameters are needed in our scheme. Note that master secret keys are required in the white-box schemes [21], identity tables are required in the white-box schemes [15,20,30], and suspected users' decryption keys are needed in the white-box scheme [30].

3.2 Security Models for AA-LU-CPABE

An AA-LU-CPABE scheme is secure if the following requirements are satisfied. First, it satisfies the standard semantic security for CP-ABE: ciphertext indistinguishability under chosen-plaintexts attacks (IND-CPA). Second, it is intractable for the authority to create a decryption key such that the $\mathbf{Trace}_{\text{wp}}$ algorithm outputs a user and the $\mathbf{Judge}_{\text{wp}}$ algorithm outputs guilty. Finally, it is infeasible for a user to create a decryption key such that the user is innocent based on the $\mathbf{Judge}_{\text{wp}}$ algorithm. Security for AA-LU-CPABE schemes are modeled in following three games between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

The IND-CPA game. The IND-CPA game for AA-LU-CPABE scheme is defined as follows:

- **Setup:** $\mathcal{B}$ chooses a security parameter λ, and runs the **Setup** algorithm and sends the system public key PK to $\mathcal{A}$.
- **Phase 1:** In addition to hash queries, the adversary $\mathcal{A}$ issues a polynomially bounded number of key generation queries:
  - **KeyGen Oracle** $\mathcal{O}_{KeyGen}$: $\mathcal{A}$ submits an identity ID and an attribute set S, $\mathcal{B}$ gives $\mathcal{A}$ the decryption key $SK_{ID,S}$.
- **Challenge:** Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs two equal length messages m_0 and m_1 from the message space and an access structure (M^*, ρ^*). It is noted that (M^*, ρ^*) cannot be satisfied by any of the queried attribute sets. $\mathcal{B}$ chooses a bit $b \in_R \{0,1\}$, computes CT^*=**Encrypt**$(PK, m_b, (M^*, \rho^*))$ and sends CT^* to $\mathcal{A}$.
- **Phase 2:** The same as **Phase 1** except that the queried attribute sets cannot match (M^*, ρ^*).
- **Guess:** $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$ and wins the game if $b' = b$.

The advantage of $\mathcal{A}$ in the IND-CPA game is defined as follows:

$$\mathsf{Adv}^{\text{IND-CPA}}_{\text{AA-LU-CPABE}}(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|.$$

Definition 8. *An AA-LU-CPABE scheme is fully-secure if no probabilistic polynomial-time (PPT) attacker can break the IND-CPA game with non-negligible advantage.*

The DishonestUser game. The DishonestUser game for AA-LU-CPABE scheme is defined as follows.

- **Setup:** $\mathcal{B}$ chooses a security parameter λ, and runs the **Setup** algorithm and sends the system public key PK to $\mathcal{A}$.
- **Key Query Phase:** For $i \in [t_q]$, while t_q is the number of key queries, $\mathcal{A}$ and $\mathcal{B}$ engage in the key generation protocol **KeyGen** to generate corresponding decryption keys SK_{ID_i,S_i} with respect to (ID_i, S_i). $\mathcal{A}$ gets the decryption keys $\{SK_{ID_i,S_i}\}_i \in [t_q]$ and runs key sanity checks to ensure that they are well-formed. It aborts if any check fails.
- **Key Forgery Phase:** $\mathcal{A}$ submits a decryption key $SK^*_{ID^*,S^*}$ corresponding to (ID^*, S^*) to $\mathcal{B}$, where $S^* \in \{S_1, S_2, \cdots, S_{t_q}\}$. If either of the following two cases is true, $\mathcal{A}$ wins the game.
  1. $\mathbf{Trace}_{\mathrm{wp}}(PK, SK^*_{ID^*,S^*}) \notin \{\top, ID_1, ID_2, \cdots, ID_{t_q}\}$.
  2. $\mathbf{Trace}_{\mathrm{wp}}(PK, SK^*_{ID^*,S^*}) = ID_j \in \{ID_1, ID_2, \cdots, ID_{t_q}\}$ and $\mathbf{Judge}_{\mathrm{wp}}(PK, SK_{ID_j,S_j}, SK^*_{ID^*,S^*}) \rightarrow \mathsf{innocent}$.

The advantage of $\mathcal{A}$ in the DishonestUser game is defined as follows:

$$\mathsf{Adv}^{\mathsf{DishonestUser}}_{\mathsf{AA\text{-}LU\text{-}CPABE}}(\mathcal{A}) = \Pr[\mathcal{A}\ wins].$$

Definition 9. *An AA-LU-CPABE scheme is DishonestUser secure if all PPT attackers have at most a negligible advantage in the above DishonestUser game.*

The DishonestAuthority Game. The DishonestAuthority game for AA-LU-CPABE scheme is defined as follows.

- **Setup:** $\mathcal{A}$ (as a malicious authority) generates the system public key PK, and sends PK, a user's (ID^*, S^*) to $\mathcal{B}$. $\mathcal{B}$ performs a sanity check on PK and (ID^*, S^*), and it aborts if the check fails.
- **Key Generation Phase:** $\mathcal{A}$ and $\mathcal{B}$ engage in the key generation protocol **KeyGen** to generate a decryption key SK_{ID^*,S^*} corresponding to (ID^*, S^*). $\mathcal{B}$ gets SK_{ID^*,S^*} and runs a key sanity check to ensure that it is well-formed. It aborts if the check fails.
- **Output:** $\mathcal{A}$ outputs a decryption key $SK^*_{ID^*,S^*}$ and succeeds if $\mathbf{Trace}_{\mathrm{wp}}(PK, SK^*_{ID^*,S^*}) \rightarrow ID^*$ and $\mathbf{Judge}_{\mathrm{wp}}(PK, SK_{ID^*,S^*}, SK^*_{ID^*,S^*}) \rightarrow \mathsf{guilty}$.

The advantage of $\mathcal{A}$ in the DishonestAuthority game is defined as:

$$\mathsf{Adv}^{\mathsf{DishonestAuthority}}_{\mathsf{AA\text{-}LU\text{-}CPABE}}(\mathcal{A}) = \Pr[\mathcal{A}\ wins].$$

Definition 10. *An AA-LU-CPABE scheme is DishonestAuthority secure if all PPT attackers have at most a negligible advantage in the above DishonestAuthority game.*

4 AA-LU-CPABE Construction

4.1 Construction

- **Setup**(1^λ)**:** The attribute authority takes a security parameter λ as inputs and runs the group generator $\mathcal{G}$ to get $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, \hat{e})$, where p_1, p_2, p_3

are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $N = p_1p_2p_3$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. Let $\mathbb{G}_{p_i}$ be the subgroup of order p_i in $\mathbb{G}$, and $g \in \mathbb{G}_{p_1}$ and $X_3 \in \mathbb{G}_{p_3}$ be the generator of $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_3}$, respectively. The attribute authority defines three collision-resistant hash functions $H_0 : \mathbb{G}_{p_1} \to \mathbb{G}_{p_1}$, $H_1 : \mathbb{G}_{p_1p_3} \to \mathbb{Z}_N$ and $H : \{0,1\}^* \to \mathbb{G}_{p_1}$. It also chooses two distinct primes p and q of equal length such that $p, q \notin \{p_1, p_2, p_3\}$, $\gcd(pq, (p-1)(q-1)) = 1$ and sets $n = pq$, $\omega = \varphi(n)$, where $\varphi(\cdot)$ is Euler's totient function. Then the attribute authority chooses $a, \alpha, \beta \in_R \mathbb{Z}_N$, $X_1 \in \mathbb{G}_{p_1}$ and computes $Y = \hat{e}(g,g)^{\alpha}$. Finally, the system public key is published as $PK = \langle N, n, \omega, g, g^a, g^{\beta}, X_1, Y\rangle$, and the master key is $MK = \langle a, \alpha, \beta, X_3\rangle$.

– **KeyGen**(PK, MK, ID, S)**:** Let S be the attribute set for the user ID who obtains the corresponding decryption key. The user and the attribute authority initiate the following key generation protocol.
  1. The user chooses $r \in_R \mathbb{Z}_N^*$ with $\gcd(r, N) = 1$ and computes $R_u = g^r$. Then it sends ID, the attribute set S and R_u to the attribute authority. Besides, it runs an interactive ZK-POK of the discrete log of R_u with respect to g with the attribute authority.
  2. If and only if the ZK-POK is valid, the attribute authority proceeds to do the following. It first chooses $t \in_R \mathbb{Z}_n^*$ with $t \notin \{p, q\}$, $c_0, \overline{c} \in_R \mathbb{Z}_N$ with $\gcd(\overline{c}, N) = 1$ and random elements R_0, R_0', R_0'', $\{R_i\}_{i\in S}$ in $\mathbb{G}_{p_3}$ based on X_3. Then the attribute authority sets $c = c_0 + \overline{c}$, $h = 1 + n$ and computes $\overline{T} = h^{ID}t^{\overline{c}n} \mod n^2$, $\overline{L_1} = g^{ac}R_0'$, $\overline{L_2} = g^{c_0}R_0''$, $\overline{K} = g^{\frac{\alpha}{a+\overline{T}+H_1(\overline{L_2})}} H_0(R_u)^{\frac{\beta}{a+\overline{T}+H_1(\overline{L_2})}} X_1^c R_0$, $\{\overline{K_i} = H(i)^{(a+\overline{T}+H_1(\overline{L_2}))c}R_i\}_{i\in S}$, and then sends $\overline{SK} = \langle \overline{K}, \overline{T}, \overline{L_1}, \overline{L_2}, \overline{c}, \{\overline{K_i}\}_{i\in S}\rangle$ to the user.
  3. The user checks whether

$$\hat{e}(\overline{K}, g^a g^{\overline{T}+H_1(\overline{L_2})}) = \hat{e}(\overline{L_1}(\overline{L_2}g^{\overline{c}})^{\overline{T}+H_1(\overline{L_2})}, X_1)\hat{e}(H_0(R_u), g^{\beta})Y,$$

$$\hat{e}(\overline{L_1}, g) = \hat{e}(\overline{L_2}g^{\overline{c}}, g^a), \text{ and } \forall i \in S,\ \hat{e}(H(i), \overline{L_1}(\overline{L_2}g^{\overline{c}})^{\overline{T}+H_1(\overline{L_2})}) = \hat{e}(\overline{K_i}, g).$$

The user aborts the interaction if one of the above checks fails. Otherwise, the users computes $T_0 = \frac{\overline{c}}{r}$ and sets the decryption key as

$$SK_{ID,S} = \langle K = \overline{K}g^{T_0}, T = \overline{T}, L_1 = \overline{L_1}, L_2 = \overline{L_2}, R_u, T_0, \{K_i = \overline{K_i}\}_{i\in S}\rangle.$$

– **Encrypt**$(PK, m, (M, \rho))$**:** The encryptor first chooses $s \in_R \mathbb{Z}_N$ and then sets a random vector $\boldsymbol{y} = (s, y_2, ..., y_n)^{\top}$, where $y_2, ..., y_n$ are used to share the encryption exponent s. For $i = 1, ..., \ell$, it calculates $\lambda_i = M_i \cdot \boldsymbol{y}$, where M_i is the vector corresponding to the ith row of M. Then it calculates $C = mY^s$, $C_0 = g^s$, $C_1 = (g^a)^s$, $C_2 = (g^{\beta})^s$, $\{C_{j,1} = X_1^{\lambda_j}H(\rho(j))^{-r_j}, C_{j,2} = g^{r_j}\}_{j\in[\ell]}$, where r_j is randomly chosen in $\mathbb{Z}_N$. Finally, the encryptor sets ciphertext as

$$CT = \langle C, C_0, C_1, C_2, \{C_{j,1}, C_{j,2}\}_{j\in[\ell]}\rangle.$$

– **Decrypt**$(PK, CT, SK_{ID,S})$**:** $CT = \langle C, C_0, C_1, C_2, \{C_{j,1}, C_{j,2}\}_{j\in[\ell]}\rangle$ under (A, ρ) is decrypted by a user (ID, S) with a decryption key $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i\in S}\rangle$ as follows. The decryptor first checks whether

S satisfies (A, ρ). If not, the algorithm returns $\perp$. Otherwise, there must exists coefficients $\{\omega_j \in \mathbb{Z}_N | \rho(j) \in S\}$ such that $\sum_{\rho(j)\in S} \omega_j M_j = (1, 0, ..., 0)$, so $\sum_{\rho(j)\in S} \omega_j \lambda_j = s$. Then, the decryptor computes $m = \frac{C}{B}$, where

$$B = \frac{\hat{e}(C_0^{T+H_1(L_2)} C_1, K)(\hat{e}(C_2, H_0(R_u))\hat{e}(C_0, (g^a g^{T+H_1(L_2)})^{T_0}))^{-1}}{\prod_{\rho(j)\in S}(\hat{e}(C_{j,1}, L_1(L_2 R_u^{T_0})^{T+H_1(L_2)})\hat{e}(C_{j,2}, K_{\rho(j)}))^{\omega_j}}.$$

- **Trace**$_{\text{wp}}(PK, SK_{ID,S})$**:** If $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i\in S}\rangle$ satisfying all the following checks, it is a well-formed decryption key, otherwise it is not well-formed and the algorithm outputs $\top$. Key Sanity Check is:
  1. $T \in \mathbb{Z}_{n^2}$, $T_0 \in \mathbb{Z}_N$, $K, L_1, L_2, R_u, K_i \in \mathbb{G}$.
  2. $\hat{e}(L_1, g) = \hat{e}(L_2 R_u^{T_0}, g^a)$.
  3. $\hat{e}(g^{-T_0} K, g^a g^{T+H_1(L_2)}) = \hat{e}(L_1(L_2 R_u^{T_0})^{T+H_1(L_2)}, X_1)\hat{e}(H_0(R_u), g^\beta)Y$.
  4. $\exists i \in S$, such that $\hat{e}(H(i), L_1(L_2 R_u^{T_0})^{T+H_1(L_2)}) = \hat{e}(K_i, g)$.

  If $SK_{ID,S}$ is well-formed, the algorithm will extract the identity ID from $T = h^{ID} t^{\bar{c}n} \mod n^2$ in $SK_{ID,S}$ as follows. Note that $T^\omega = h^{\omega ID} t^{\bar{c}n\omega} \mod n^2 = (1+n)^{\omega ID} t^{\bar{c}n\varphi(n)} \mod n^2 = 1+n\omega ID \mod n^2$, hence it outputs the identity $ID = \frac{(T^\omega \mod n^2)-1}{n\omega}$, which is used to identify the possible malicious user.
- **Judge**$_{\text{wp}}\big(PK, SK_{ID,S}, SK^*_{ID,S}\big)$**:** Suppose a user (ID, S) with the decryption key $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i\in S}\rangle$ is identified as a malicious user by the system based on the traced key

$$SK^*_{ID,S} = \langle K^*, T^*, L_1^*, L_2^*, R_u^*, T_0^*, \{K_i^*\}_{i\in S}\rangle,$$

but it claims to be innocent and framed by the system. The user and the judge interact in the following protocol.

1. The user sends the decryption key $SK_{ID,S}$ to the judge. The judge checks if $SK_{ID,S}$ passes the key sanity checks used in the tracing algorithm and aborts if the check fails.
2. Otherwise, the judge tests whether $T_0 = T_0^*$ or not. If no, it outputs innocent to indicate that the user is innocent and is framed by the system. Otherwise, it outputs guilty to indicate that $SK^*_{ID,S}$ is maliciously leaked by the user.

4.2 Security Analysis

Theorem 1. *If Assumptions 1, 2 and 3 hold, then the proposed AA-LU-CPABE scheme is semantically secure.*

Proof. It's noted that the proposed AA-LU-CPABE scheme Π is based on the CP-ABE scheme [10] denoted by Π_o. Because the scheme Π_o is adaptively chosen attribute sets and chosen plaintexts secure under the assumptions 1, 2 and 3, if we can reduce the security of Π to that of Π_o, then the proposed AA-LU-CPABE scheme is secure in the IND-CPA security model under the assumptions 1, 2 and 3. In the following, we will show that any PPT attacker $\mathcal{A}$ with a non-negligible advantage $\mathsf{Adv}^{\text{IND-CPA}}_{\text{AA-LU-CPABE}}(\mathcal{A}) = \epsilon$ in the proposed security model against Π can be

used to design a PPT simulator $\mathcal{B}$, which can break the security of Π_o with an advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{CPABE}}(\mathcal{B}) = \epsilon$. The simulator $\mathcal{B}$ acts as the challenger and interacts with $\mathcal{A}$ in the IND-CPA security model. The simulation proceeds as follows:

Setup. The challenger $\mathcal{B}$ receives public parameters $\langle N, g, g^{\gamma}, Y = \hat{e}(g,g)^{\alpha}, \{U_i = g^{u_i}\}_{i\in\mathcal{U}_o}\rangle$ from the challenger $\mathcal{B}_o$ of Π_o, where $\mathcal{U}_o$ is the attribute universe of Π_o and it satisfies $|\mathcal{U}_o| \geq q_H$, where q_H is the number of hash queries to H. $\mathcal{B}$ also chooses two distinct primes p and q of equal length such that $\gcd(pq, (p-1)(q-1)) = 1$ and sets $n = pq$, $\omega = \varphi(n)$. Then $\mathcal{B}$ chooses $a, \alpha, \beta \in_R \mathbb{Z}_N$, sets $X_1 = g^{\gamma}$ and the system public key as $PK = \langle N, n, \omega, g, g^a, g^{\beta}, X_1, Y\rangle$. During the game, $\mathcal{A}$ will consult $\mathcal{B}$ for answers to the random oracles H_0, H_1 and H. $\mathcal{B}$ keeps three tables $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}$ to store the answers used in H_0, H_1 and H, respectively. Finally $\mathcal{B}$ sends PK to $\mathcal{A}$.

Phase 1. The adversary $\mathcal{A}$ makes the following queries.

- **Hash Oracle** $\mathcal{O}_{H_0}(x_0)$: Whenever there is a query on H_0 for input x_0, $\mathcal{B}$ first looks if there is an item containing x_0 in $\mathcal{L}_0$. If it is, the previous defined value is returned. Otherwise, it chooses $\gamma' \in_R \mathbb{Z}_N$ and sets $H_0(x_0) = g^{\gamma'}$, adds the entry $\langle x_0, \gamma', H_0(x_0) = g^{\gamma'}\rangle$ to $\mathcal{L}_0$ and returns $g^{\gamma'}$.
- **Hash Oracle** $\mathcal{O}_{H_1}(x_1)$: Whenever there is a query on H_1 for input x_1, $\mathcal{B}$ first looks if there is an item containing x_1 in $\mathcal{L}_1$. If it is, the previous defined value is returned. Otherwise, it chooses $\gamma'' \in_R \mathbb{Z}_N$, adds the entry $\langle x_1, \gamma'', H_1(x_1) = \gamma''\rangle$ to $\mathcal{L}_1$ and returns γ''. Note that, during the process of answering key generation queries, $\mathcal{B}$ will adaptively update $\mathcal{L}_1$.
- **Hash Oracle** $\mathcal{O}_H(x)$: Whenever there is a query on H for input x, $\mathcal{B}$ first looks if there is an item containing x in $\mathcal{L}$. If it is, the previous defined value is returned. Otherwise, it sets $H(x) = U_x$, adds the entry $\langle x, H(x) = U_x\rangle$ to $\mathcal{L}$ and returns U_x.
- **KeyGen Oracle** $\mathcal{O}_{KeyGen}(ID, S)$: Suppose $\mathcal{A}$ summits an identity ID and an attribute set S in a secret key query. $\mathcal{B}$ sends S to $\mathcal{B}_o$ and obtains the corresponding decryption key $SK_S = \langle \widehat{K} = g^{\alpha} g^{\gamma\hat{c}} R_0, \widehat{L} = g^{\hat{c}} R_0', \{\widehat{K}_i = U_i^{\hat{c}} R_i\}_{i\in S}\rangle$. Recall that during the key generation protocol, the key applicant chooses $r \in_R \mathbb{Z}_N^*$ and computes $R_u = g^r$. Besides, the key applicant gives to the authority a zero-knowledge proof of knowledge of the discrete log of R_u with respect to g. The authority can extract r with all but negligible probability by using Extractor on the key applicant during the proof of knowledge protocol. $\mathcal{B}$ chooses $t \in_R \mathbb{Z}_n^*$, $\bar{c} \in_R \mathbb{Z}_N$, sets $h = 1 + n$ and computes $T = \overline{T} = h^{ID} t^{\bar{c}n} \bmod n^2$. Also, $\mathcal{B}$ chooses $R_0'' \in_R \mathbb{G}_{p_3}$ by using X_3, $\gamma'' \in_R \mathbb{Z}_N$ and implicitly sets $c_0 = \hat{c}/(a + T + \gamma'') - \bar{c} \mod N$, and hence $c = c_0 + \bar{c} = \hat{c}/(a + T + \gamma'')$. Then $\mathcal{B}$ sets $\overline{L_2} = \widehat{L}^{\frac{1}{a+T+\gamma''}} R_0'' = g^c R_0'^{\frac{1}{a+T+\gamma''}} R_0''$ and returns γ'' for the hash query $H_1(\overline{L_2})$, that is, $\mathcal{B}$ will add the entry $\langle \overline{L_2}, \gamma'', H_1(\overline{L_2}) = \gamma''\rangle$ to $\mathcal{L}_1$. Therefore, $c = \hat{c}/(a + T + H_1(\overline{L_2}))$. Subsequently, $\mathcal{B}$ computes $\overline{L_1} = \widehat{L}^{\frac{a}{a+T+H_1(\overline{L_2})}} = g^{ac} R_0'^{\frac{a}{a+T+H_1(\overline{L_2})}}$, $\overline{K} = (\widehat{K})^{\frac{1}{a+T+H_1(\overline{L_2})}} H_0(R_u)^{\frac{\beta}{a+T+H_1(\overline{L_2})}} = g^{\frac{\alpha}{a+\overline{T}+H_1(\overline{L_2})}} H_0(R_u)^{\frac{\beta}{a+\overline{T}+H_1(\overline{L_2})}} X_1^c R_0^{\frac{1}{a+T+H_1(\overline{L_2})}}$, and $\{\overline{K_i} = \widehat{K}_i = U_i^{\hat{c}} R_i = H(i)^{(a+\overline{T}+H_1(\overline{L_2}))c} R_i\}_{i\in S}$. Subsequently, $\mathcal{B}$ computes $T_0 = \frac{\bar{c}}{r}$ and sets $K =$

$\overline{K} g^{T_0}, L_1 = \overline{L_1}, L_2 = \overline{L_2}, \{K_i = \overline{K_i}\}_{i \in S}$. Finally, $\mathcal{B}$ returns the decryption key $SK_{ID,S} = \langle K, T, L_1, L_2, R_u, T_0, \{K_i\}_{i \in S} \rangle$.

Challenge. The adversary $\mathcal{A}$ summits two messages m_0 and m_1 of equal length and an LSSS access structure (M^*, ρ^*) to $\mathcal{B}$. Note that (M^*, ρ^*) cannot be satisfied by any of the queried attribute sets. Then $\mathcal{B}$ sends m_0, m_1 and (M^*, ρ^*) to $\mathcal{B}_o$ to obtain the challenge ciphertext of Π_o as follows.

$$\widehat{CT} = \langle \widehat{C} = m_b Y^s, \widehat{C}_0 = g^s, \{\widehat{C}_{j,1} = g^{\gamma \lambda_j} U_{\rho(j)}^{-r_j}, \widehat{C}_{j,2} = g^{r_j}\}_{j \in [\ell]} \rangle.$$

$\mathcal{B}$ sets $C = \widehat{C}$, $C_0 = \widehat{C}_0$, $C_1 = (\widehat{C}_0)^a = g^{as}$, $C_2 = (\widehat{C}_0)^\beta = g^{\beta s}$, $C_{j,1} = \widehat{C}_{j,1} = X_1^{\lambda_j} H(\rho(j))^{-r_j}$ and $C_{j,2} = \widehat{C}_{j,2}$. Finally, $\mathcal{B}$ gives to $\mathcal{A}$ the challenge ciphertext

$$CT = \langle C, C_0, C_1, C_2, \{C_{j,1}, C_{j,2}\}_{j \in [\ell]} \rangle.$$

Phase 2. The same as **Phase 1** except that the queried attribute sets cannot match (M^*, ρ^*).

Guess. The adversary $\mathcal{A}$ outputs a guess bit b' of b. $\mathcal{B}$ just sends b' to $\mathcal{B}_o$. It easily follows that $\mathsf{Adv}^{\text{IND-CPA}}_{\text{CPABE}}(\mathcal{B}) = \mathsf{Adv}^{\text{IND-CPA}}_{\text{AA-LU-CPABE}}(\mathcal{A}) = \epsilon$. ■

Theorem 2. *If Assumption 2 and ℓ-SDH assumption hold, then the proposed AA-LU-CPABE scheme is* DishonestUser *secure provided that $t_q < \ell$, where at most t_q key queries are issued during the* DishonestUser *security game.*

Proof. Suppose there is a PPT adversary $\mathcal{A}$ that wins the traceability game with a non-negligible advantage ϵ after making t_q key queries. Without loss of generality, assuming $\ell = t_q + 1$, we construct a PPT algorithm $\mathcal{B}$ that has a non-negligible advantage in breaking Assumption 2 or ℓ-SDH assumption. $\mathcal{B}$ is given instances as follows.

- $\mathcal{B}$ is given an instance of Assumption 2 problem: Let $\mathbb{G}$ be a bilinear group of order $N = p_1 p_2 p_3$, where p_1, p_2, p_3 are distinct primes, $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, and $\mathbb{G}_{p_i}$ be the subgroup of order p_i in $\mathbb{G}$, $\hat{g}, X_1 \in \mathbb{G}_{p_1}$, $X_2, Y_2 \in \mathbb{G}_{p_2}$ and $X_3, Y_3 \in \mathbb{G}_{p_3}$. $b \in \{0, 1\}$, and $X \in \mathbb{G}$ if $b = 0$, $X \in \mathbb{G}_{p_1 p_3}$ if $b = 1$. $\mathcal{B}$ is given an instance $\mathsf{IN}_{\mathsf{A2}} = (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, \hat{g}, X_1 X_2, X_3, Y_2 Y_3, X)$.
- $\mathcal{B}$ is given an instance of ℓ-SDH problem: Let $\mathbb{G}$ be a bilinear group of order $N = p_1 p_2 p_3$, where p_1, p_2, p_3 are distinct primes, $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, $\mathbb{G}_{p_i}$ be the subgroup of order p_i in $\mathbb{G}$, $a \in \mathbb{Z}_{p_1}^*$ and $\hat{g} \in \mathbb{G}_{p_1}$. $\mathcal{B}$ is given an instance $\mathsf{IN}_{\mathsf{SDH}} = (\mathbb{G}, \mathbb{G}_T, N, \hat{e}, \hat{g}, \hat{g}^a, \ldots, \hat{g}^{a^\ell}, p_1, p_2, p_3)$.

The goal of $\mathcal{B}$ is to output a bit $b' \in \{0, 1\}$ to determine $X \in \mathbb{G}$ or $X \in \mathbb{G}_{p_1 p_3}$ for solving the assumption 2 problem, and a pair (T_r, ω_r) satisfying $\omega_r = \hat{g}^{\frac{1}{a + T_r}}$ for solving the ℓ-SDH problem. $\mathcal{B}$ will make use of $\mathcal{A}$ to break Assumption 2 or ℓ-SDH assumption. After the setup phase, $\mathcal{A}$ can get the system public key from $\mathcal{B}$. During the key query phase, where at most t_q key queries are issued, $\mathcal{A}$ submits identity and attribute set pairs to $\mathcal{B}$ to get decryption keys. Then, after the key forgery phase, $\mathcal{A}$ submits a decryption key SK^* corresponding to

(ID^*, S^*) to $\mathcal{B}$. The detailed queries will be given in the full version of the paper due to the space limitation.

Finally, based on the complete probability formula, we can know that $\mathcal{B}$ can break Assumption 2 with advantage at leat $\frac{\epsilon}{8}$ or break ℓ-SDH assumption with advantage at leat $\frac{\epsilon}{8}$. ■

Theorem 3. *If the discrete log assumption holds in $\mathbb{G}_{p_1}$, then the proposed AA-LU-CPABE scheme is* DishonestAuthority *secure.*

Proof. Suppose there is a PPT adversary $\mathcal{A}$ that has a non-negligible advantage in winning the DishonestAuthority game for our AA-LU-CPABE scheme, we construct a PPT algorithm $\mathcal{B}$ that has a non-negligible advantage in breaking the discrete log assumption in $\mathbb{G}_{p_1}$. $\mathcal{B}$ proceeds as follows. $\mathcal{B}$ receives from $\mathcal{A}$ the system public key $PK = \langle N, n, \omega, g, g^a, g^\beta, X_1, Y\rangle$ and a user's identity and attribute set (ID^*, S^*). Then $\mathcal{B}$ sends g to the discrete log problem challenger and obtains an instance $(g, R_u = g^r)$ of the discrete log assumption.

$\mathcal{B}$ engages in the key generation protocol with $\mathcal{A}$ to get a decryption key for the user (ID^*, S^*). It sends R_u to $\mathcal{A}$ and provides a zero-knowledge proof of knowledge of the discrete log of R_u. On the other hand, $\mathcal{B}$ receives from $\mathcal{A}$ a primary decryption key $\overline{SK} = \langle \overline{K}, \overline{T}, \overline{L_1}, \overline{L_2}, \overline{c}, \{\overline{K_i}\}_{i\in S^*}\rangle$. Then, $\mathcal{B}$ performs the key sanity checks. $\mathcal{B}$ aborts the interaction if the checks fail. Otherwise, $\mathcal{B}$ chooses $r' \in_R \mathbb{Z}_N^*$, computes $T_0 = \frac{\overline{c}}{r'}$ and sets the decryption key based on the algorithm. Now with a non-negligible advantage, $\mathcal{A}$ outputs a decryption key $SK^*_{ID^*,S^*}$ and it succeeds in framing the user (ID^*, S^*). Hence, $SK^*_{ID^*,S^*}$ has the form of $SK^*_{ID^*,S^*} = \langle K = \overline{K} g^{T_0^*}, T, L_1, L_2, R_u, T_0^*, \{K_i\}_{i\in S^*}\rangle$. Finally, $\mathcal{B}$ calculates $\frac{\overline{c}}{T_0^*}$ as the solution of the discrete log problem (g, R_u). More details will be given in the full version of the paper due to the space limitation. ■

5 Conclusions and Future Work

In this paper, we propose an AA-LU-CPABE scheme that simultaneously supports user traceability, authority accountability and auditing in the white-box model. Our scheme needs almost no storage for tracing in that it does not need to maintain an identity table of users for tracing. The proposed AA-LU-CPABE scheme is proven secure in the random oracle model against adaptive adversaries, and it is highly expressive and can take any monotonic access structures as ciphertext policies. It would be interesting to construct AA-LU-CPABE schemes with desirable security and performance features in the black-box model.

Acknowledgements. We are grateful to the anonymous reviewers for their invaluable suggestions. This work is supported by National Natural Science Foundation of China (No. 61402366, 61472091, 61272037, 61472472, and 61272457), Program for New Century Excellent Talents in University (No. NCET-13-0946), Distinguished Young Scholars Fund of Department of Education, Guangdong Province (No. Yq2013126), Natural Science Basic Research Plan in Shaanxi Province (No. 2015JQ6236), and Scientific Research Program Funded by Shaanxi Provincial Education Department (No. 15JK1686).

References

1. Balu, A., Kuppusamy, K.: An expressive and provably secure ciphertext-policy attribute-based encryption. Inf. Sci. **276**, 354–362 (2014)
2. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Technion-Israel Institute of technology, Faculty of computer science (1996)
3. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, SP 2007, pp. 321–334. IEEE, Los Alamitos (2007)
5. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
6. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: CCS 2007, pp. 456–465. ACM, New York (2007)
7. Deng, H., Wu, Q., Qin, B., Mao, J., Liu, X., Zhang, L., Shi, W.: Who Is touching my cloud. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part I. LNCS, vol. 8712, pp. 362–379. Springer, Heidelberg (2014)
8. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006, pp. 89–98. ACM, New York (2006)
9. Lai, J., Deng, R.H., Li, Y.: Expressive CP-ABE with partially hidden access structures. In: ASIACCS 2012, pp. 18–19. ACM, New York (2012)
10. Lewko, A., 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 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
11. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011)
12. Li, J., Huang, Q., Chen, X., Chow, S.S.M., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: ASIACCS 2011, pp. 386–390. ACM, New York (2011)
13. Li, J., Ren, K., Kim, K.: A2be: Accountable attribute-based encryption for abuse free access control. Cryptology ePrint Archive, Report 2009/118 (2009)
14. Li, J., Ren, K., Zhu, B., Wan, Z.: Privacy-aware attribute-based encryption with user accountability. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 347–362. Springer, Heidelberg (2009)
15. Liu, Z., Cao, Z., Wong, D.S.: White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. IEEE Trans. Inf. Forensics Secur. **8**(1), 76–88 (2013)
16. Liu, Z., Cao, Z., Wong, D.S.: Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on ebay. In: CCS 2013, pp. 475–486. ACM, New York (2013)
17. Liu, Z., Cao, Z., Wong, D.S.: Traceable CP-ABE: how to trace decryption devices found in the wild. IEEE Trans. Inf. Forensics Secur. **10**(1), 55–68 (2015)
18. Liu, Z., Wong, D.S.: Traceable CP-ABE on prime order groups: Fully secure and fully collusion-resistant blackbox traceable. Cryptology ePrint Archive, Report 2015/850 (2015)

19. Liu, Z., Wong, D.S.: Practical ciphertext-policy attribute-based encryption: traitor tracing, revocation, and large universe. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) Applied Cryptography and Network Security. LNCS, vol. 9092, pp. 127–146. Springer, Switzerland (2015)
20. Ning, J., Cao, Z., Dong, X., Wei, L., Lin, X.: Large universe ciphertext-policy attribute-based encryption with white-box traceability. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 55–72. Springer, Heidelberg (2014)
21. Ning, J., Dong, X., Cao, Z., Wei, L.: Accountable Authority Ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015, Part II. LNCS, vol. 9327, pp. 270–289. Springer, Switzerland (2015)
22. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) Applied Cryptography and Network Security. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008)
23. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
24. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS 2013, pp. 463–474. ACM, New York (2013)
25. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
26. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
27. Shi, Y., Zheng, Q., Liu, J., Han, Z.: Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation. Inf. Sci. **295**, 221–231 (2015)
28. Yu, S., Ren, K., Lou, W., Li, J.: Defending against key abuse attacks in KP-ABE enabled broadcast systems. In: Chen, Y., Dimitriou, T.D., Zhou, J. (eds.) SecureComm 2009. LNICST, vol. 19, pp. 311–329. Springer, Heidelberg (2009)
29. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: ASIACCS 2010, pp. 261–270. ACM, New York (2010)
30. Zhang, X., Jin, C., Li, C., Wen, Z., Shen, Q., Fang, Y., Wu, Z.: Ciphertext-policy attribute-based encryption with user and authority accountability. In: Thuraisingham, B., et al. (eds.) SecureComm 2015. LNICST, vol. 164, pp. 500–518. Springer, Heidelberg (2015). doi:10.1007/978-3-319-28865-9_27
31. Zhang, Y., Chen, X., Li, J., Li, H., Li, F.: Fdr-abe: Attribute-based encryption with flexible and direct revocation. In: INCoS 2013, pp. 38–45. IEEE, Los Alamitos (2013)
32. Zhang, Y., Chen, X., Li, J., Li, H., Li, F.: Attribute-based data sharing with flexible and direct revocation in cloud computing. KSII Transactions on Internet &. Inf. Syst. **8**(11), 4028–4049 (2014)
33. Zhang, Y., Chen, X., Li, J., Wong, D.S., Li, H.: Anonymous attribute-based encryption supporting efficient decryption test. In: ASIACCS 2013, pp. 511–516. ACM, New York (2013)
34. Zhang, Y., Zheng, D., Chen, X., Li, J., Li, H.: Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) ProvSec 2014. LNCS, vol. 8782, pp. 259–273. Springer, Heidelberg (2014)