Fully Secure Multi-authority Ciphertext-Policy Attribute-Based Encryption without Random Oracles

Zhen Liu[1,2,⋆], Zhenfu Cao[1,⋆], Qiong Huang[3], and Duncan S. Wong[2], and Tsz Hon Yuen[4]

[1] Shanghai Jiao Tong University, Shanghai, China
{liuzhen@,zfcao@cs.}sjtu.edu.cn
[2] City University of Hong Kong, Hong Kong S.A.R., China
{zhenliu7@student.,duncan@}cityu.edu.hk
[3] South China Agricultural University, Guangzhou, China
csqhuang@alumni.cityu.edu.hk
[4] The University of Hong Kong, Hong Kong S.A.R., China
thyuen@cs.hku.hk

Abstract. Recently Lewko and Waters proposed the first fully secure multi-authority ciphertext-policy attribute-based encryption (CP-ABE) system in the random oracle model, and leave the construction of a fully secure multi-authority CP-ABE in the standard model as an open problem. Also, there is no CP-ABE system which can completely prevent individual authorities from decrypting ciphertexts. In this paper, we propose a new multi-authority CP-ABE system which addresses these two problems positively. In this new system, there are multiple Central Authorities (CAs) and Attribute Authorities (AAs), the CAs issue identity-related keys to users and are not involved in any attribute related operations, AAs issue attribute-related keys to users and each AA manages a different domain of attributes. The AAs operate independently from each other and do not need to know the existence of other AAs. Messages can be encrypted under any monotone access structure over the entire attribute universe. The system is adaptively secure in the standard model with adaptive authority corruption, and can support large attribute universe.

Keywords: Attribute based encryption, ciphertext-policy, multi-authority.

1 Introduction

In traditional cryptosystems, messages are encrypted for receivers who are identified by the keys they are holding. The same holds for Identity-Based Encryption (IBE) [20,3] where user public keys are binary strings, such as email

⋆ The authors are supported by the National Nature Science Foundation of China No. 61033014, No. 60970110 and No. 60972034, and by the National 973 Program No. 2007CB311201.

V. Atluri and C. Diaz (Eds.): ESORICS 2011, LNCS 6879, pp. 278–297, 2011.

addresses, which uniquely identify the users. In some applications that require access control, messages are required to be encrypted for multiple receivers who are identified by their roles, rather than their actual identities. For example, Alice may want to encrypt some message to all PhD students and alumni in the Department of Computer Science of University X, that is, she wants to do the encryption with an access policy such as ("UNIV.X.COMPUTER SCIENCE" **AND** ("UNIV.X.PhD STUDENT" **OR** "UNIV.X.ALUMNI")), so that only those receivers whose attributes satisfy this policy can perform the decryption successfully. As a counterexample, suppose Bob is a Computer Science undergraduate student, he cannot decrypt even if he colludes with another student, say Tom, who is a PhD student but in the Mathematics department.

Attribute-Based Encryption (ABE) [19] provides a solution to the application above. In ABE (Ciphertext-Policy ABE or CP-ABE as an example), an access policy defined over a set of attributes is associated with each encrypted message, and each user in the system has a private key obtained from an authority (e.g., the UNIV.X Registry) corresponding to the user's attributes (or credentials). If a user's attributes satisfy the access policy of a ciphertext, the user can decrypt the ciphertext. In most of the ABE systems [10,18,2,7,9,21,11,17], attributes are managed by a single authority. In some applications however, this may not desirable. For example, Alice encrypts a message with access policy ("UNIV.X.COMPUTER SCIENCE" **AND** "UNIV.X.ALUMNI" **AND** "GOOGLE.ENGINEER") so that only receivers who are the computer science alumni of University X and currently working as an engineer for Google can decrypt. The authority UNIV.X Registry may only manage attributes for the students, staff and alumni of University X, while Google Registry may be the authority handling its employees' attributes. A single-authority ABE may not be appropriate in this scenario.

Multi-authority ABE systems are proposed to address this issue. For the systems in [5,14,6,15,16], they are selectively secure where the adversary has to commit to the access policy before seeing the public parameters. Recently in [13], Lewko and Waters proposed a new one. Although their system may become inefficient for large attribute universe, it is the first fully secure multi-authority ABE system and is proven secure in the random oracle model. Multi-authority ABE also helps alleviate the extent of trust on authorities. In a single-authority ABE system [19,10,18,2,7,9,21,11,17], the authority can decrypt all ciphertexts. In some multi-authority ABE systems [5,15,16], there is still a central authority which can decrypt all ciphertexts. In [14,6], the multi-authority ABE systems do not have a central authority. They are Key-Policy ABE (KP-ABE), and the techniques do not seem to apply to CP-ABE. In the multi-authority CP-ABE [13], no single authority can decrypt all ciphertexts and each authority can only decrypt ciphertexts that the associated access policy can be satisfied by the authority's own domain of attributes. For example, although neither UNIV.X Registry nor Google Registry alone can decrypt a ciphertext with access policy ("UNIV.X.ALUMNI" **AND** "GOOGLE.ENGINEER"), UNIV.X Registry can decrypt a ciphertext with access policy ("UNIV.X.COMPUTER SCIENCE" **AND** "UNIV.X.ALUMNI").

Table 1. A Comparison between existing work and this work

	Multi-Authority	Adaptively Secure	Standard Model	Prevent Decryption by Individual Authorities	Support Large Attribute Universe	KP/CP
[19,10,18]	×	×	√	×	√	KP
[2]	×	×	×	×	√	CP
[7,9,21]	×	×	√	×	√	CP
[11,17]	×	√	√	×	√	KP+CP
[5]	√	×	√	×	√	KP
[14]	√	×	√	√	√	KP
[6]	√	×	√	√	√	KP
[15,16]	√	×	√	×	√	CP
[13]	√	√	×	Partially	×	CP
this work	√	√	√	√	√	CP

1.1 Our Results

We propose a new multi-authority CP-ABE system which has multiple Central Authorities (CAs) and Attribute Authorities (AAs). The CAs issue identity-related keys to users but do not involve in any attribute-related operations. AAs issue attribute-related keys to users. Each AA manages a different attribute domain and operates independently from other AAs. A party may join the system to be an AA by simply registering itself to the CAs, and then publishing its attribute-related public parameters. In the proposed system, no authority can independently decrypt any ciphertext. We show that the system is adaptively secure in the standard model which captures adaptive authority corruption. Its access policy can be any monotone access structure and the system supports large attribute universe. The efficiency of the system is also comparable to the corresponding single-authority CP-ABE system. Table 1 shows a comparison in properties and security levels between current ABE systems and this new system.

1.2 System Architecture

Fig. 1 shows the architecture of the multi-authority CP-ABE system. The system has D Central Authorities, $CA_1, \ldots, CA_D$, and K Attribute Authorities, $AA_1, \ldots, AA_K$. Each AA manages a different domain of attributes (e.g., AA_1 manages U_1, and so on). When a user joins the system, each CA issues an identity-related key to the user. Then the user obtains an attribute-related key corresponding to the attributes that the user entitled from an AA (e.g., UNIV.X Registry). In practice, one may imagine that there could have multiple CAs run by different organizations while all of them are governed under some ordinance made by the government, then universities and companies can join the system as AAs. Each AA manages its own attribute domain and the AAs operate independently from each other. The trust on the CAs by the users in the system can also be alleviated as it is unlikely to have all the CAs collude if some appropriate governmental policies and business measures are put into place to govern the practice of the CAs.

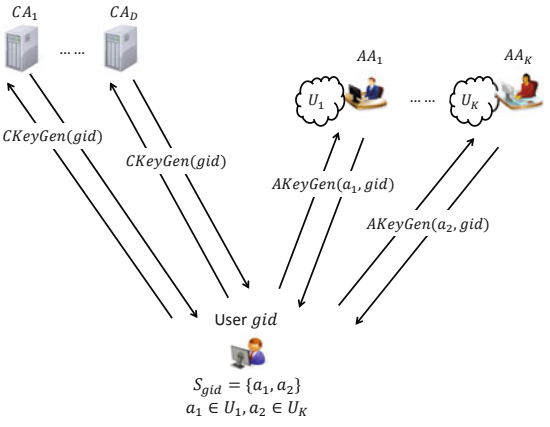

Fig. 1. Architecture of Multi-Authority CP-ABE

1.3 Related Work

Attribute-Based Encryption (ABE) was introduced by Sahai and Waters [19] and classified into Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) by Goyal et al. [10]. In KP-ABE, attributes are associated with ciphertexts and formulas (or policies) defined over attributes are associated with users' secret keys. In CP-ABE, attributes are associated with users' secret keys and policies are associated with ciphertexts. The single-authority ABE systems proposed in [19,10,18] are KP-ABE, those proposed in [2,7,9,21] are CP-ABE, and [11,17] cover both KP-ABE and CP-ABE.

Chase [5] proposed the first multi-authority ABE system where there are one CA (Central Authority) and multiple AAs (Attribute Authorities). The CA issues identity-related keys to users and the AAs manage attributes and issue attribute-related keys. A user's keys from different AAs are linked together by the user's *global identifier*. The expressiveness of the system is limited and only "AND" policy between the AAs is supported. Also, the CA can decrypt all ciphertexts. Lin et al. [14] remove the CA using a threshold technique where the set of authorities is fixed ahead of time and all authorities must interact during the system setup. The system cannot defend against collusion attack by m or more users where m is a system parameter chosen at setup. Chase and Chow [6] also remove the central authority using a distributed PRF (pseudo random function) technique. However, the expressiveness is as limited as the original Chase's system, and their technique does not seem to apply to CP-ABE. While [5,14,6] focus on KP-ABE, Müller, Katzenbeisser, and Eckert [15,16] proposed the first multi-authority CP-ABE system where there are one CA and multiple AAs. The AAs operate independently from each other and therefore is flexible and practical. However, the CA in the system can still decrypt all ciphertexts.

In [13], Lewko and Waters proposed a new multi-authority CP-ABE system. Different from all previous multi-authority ABE systems, which are all

selectively secure, this new system is adaptively secure. The system is expressive, supporting any monotone access structures. There is no central authority and each authority in the system operates independently from other authorities. The system is proven secure in the random oracle model, and does not efficiently support large attribute universe. In addition, each authority can still independently decrypt ciphertexts, if the associated access policies can be satisfied by the attributes managed by the authority.

Paper Organization. In the next section, we define the multi-authority CP-ABE and formalize its security model. Some background such as number-theoretic assumptions and access structures are reviewed in Sec. 3. Our scheme is described in Sec. 4, and some extensions are proposed in Sec. 5. In Sec. 6, the scheme is compared with some existing schemes and the paper is concluded in Sec. 7.

2 Definition and Security Model

There are three sets of entities in a Multi-Authority Ciphertext-Policy Attribute-Based Encryption (MA-CP-ABE) system: (1) Central Authorities (CAs), (2) Attribute Authorities (AAs) and (3) users. Let $CA_1, \ldots, CA_D$ be central authorities and $\mathbb{D} = \{1, \ldots, D\}$ the index set of the CAs, that is, using $d \in \mathbb{D}$ to denote the index of central authority CA_d. Let $AA_1, \ldots AA_K$ be attribute authorities and $\mathbb{K} = \{1, \ldots, K\}$ the index set of the AAs. Each user has a global identifier denoted as gid. The CAs are responsible for issuing keys to users according to their global identifiers. The AAs are responsible for issuing keys corresponding to attributes, and each AA manages a different attribute domain (e.g., AA_i manages attributes for a university registry, AA_j manages attributes for a company registry, etc.). Let U_k be the attribute domain managed by AA_k where $U_i \cap U_j = \emptyset$ for all $i \neq j \in \mathbb{K}$, and $U = \bigcup_{k=1}^{K} U_k$ be the attribute universe.

2.1 Definition

An MA-CP-ABE system consists of the following seven algorithms:

$\mathsf{GlobalSetup}(\lambda) \rightarrow (\mathsf{GPK})$. The algorithm takes as input the security parameter λ and outputs the global public parameter GPK of the system.

$\mathsf{CASetup}(\mathsf{GPK}, d) \rightarrow (\mathsf{CPK}_d, \mathsf{CAPK}_d, \mathsf{CMSK}_d)$. Each CA_d runs the algorithm with GPK and its index d as input, and produces master secret key CMSK_d and public parameters $(\mathsf{CPK}_d, \mathsf{CAPK}_d)$. CAPK_d will be used by AAs only.

$\mathsf{AASetup}(\mathsf{GPK}, k, U_k) \rightarrow (\mathsf{APK}_k, \mathsf{ACPK}_k, \mathsf{AMSK}_k)$. Each AA_k runs the algorithm with GPK, its index k and its attribute domain U_k as input, and produces master secret key AMSK_k and public parameters $(\mathsf{APK}_k, \mathsf{ACPK}_k)$. ACPK_k will be used by CAs only.

$\mathsf{Encrypt}(M, \mathbb{A}, \mathsf{GPK}, \{\mathsf{CPK}_d | d \in \mathbb{D}\}, \{\mathsf{APK}_k\}) \rightarrow CT$. The algorithm takes as input a message M, an access policy $\mathbb{A}$ defined over the attribute universe U, the global public parameter GPK, CAs' public parameters $\{\mathsf{CPK}_d | d \in \mathbb{D}\}$, and the related AAs' public parameters $\{\mathsf{APK}_k\}$. It outputs a ciphertext CT which contains the access policy $\mathbb{A}$.

$\mathsf{CKeyGen}(gid, \mathsf{GPK}, \{\mathsf{ACPK}_k | k \in \mathbb{K}\}, \mathsf{CMSK}_d) \rightarrow (\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d})$. When a user with global identifier gid visits CA_d for obtaining a key, CA_d runs the algorithm, which takes as input gid, GPK, $\{\mathsf{ACPK}_k | k \in \mathbb{K}\}$, and CA_d's master secret key CMSK_d. It outputs a ***user-central-key*** $(\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d})$, where $\mathsf{ucpk}_{gid,d}$ is called ***user-central-public-key***.

$\mathsf{AKeyGen}(att, \{\mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}, \mathsf{GPK}, \{\mathsf{CAPK}_d | d \in \mathbb{D}\}, \mathsf{AMSK}_k) \rightarrow \mathsf{uask}_{att,gid}$ or $\perp$. When a user requests a secret key for attribute att from AA_k, AA_k runs the algorithm, which takes as input att, $\{\mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}$, GPK, $\{\mathsf{CAPK}_d | d \in \mathbb{D}\}$ and AMSK_k. If all $\mathsf{ucpk}_{gid,d}$s are valid, the algorithm outputs a ***user-attribute-key*** $\mathsf{uask}_{att,gid}$, otherwise it outputs $\perp$. For a user gid with attribute set S_{gid}, the user's ***decryption-key*** is defined as

$$\mathsf{DK}_{gid} = (\{\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}, \{\mathsf{uask}_{att,gid} | att \in S_{gid}\}).$$

$\mathsf{Decrypt}(CT, \mathsf{GPK}, \{\mathsf{APK}_k\}, \mathsf{DK}_{gid}) \rightarrow M$ or $\perp$. The algorithm takes as input a ciphertext CT associated with access policy $\mathbb{A}$, GPK, the related attribute authorities' public parameters $\{\mathsf{APK}_k\}$, and a decryption-key DK_{gid} with attribute set S_{gid}. If S_{gid} satisfies the access policy $\mathbb{A}$, the algorithm outputs the message M, otherwise it outputs $\perp$ indicating the failure of decryption.

2.2 Security Model

The security of MA-CP-ABE is defined by the following game run between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$. $\mathcal{A}$ can corrupt CAs and AAs by specifying $\mathbb{K}_c \subset \mathbb{K}$ and $\mathbb{D}_c \subset \mathbb{D}$ after seeing the public parameters[1], where $\mathbb{D} \setminus \mathbb{D}_c \neq \emptyset$ and $\mathbb{K} \setminus \mathbb{K}_c \neq \emptyset$. Without loss of generality, we assume that $\mathcal{A}$ corrupts all CAs but one, i.e., $|\mathbb{D} \setminus \mathbb{D}_c| = 1$.

Setup

- $\mathsf{GlobalSetup}$, $\mathsf{CASetup}(\mathsf{GPK}, d)$ $(d = 1, \ldots, D)$ and $\mathsf{AASetup}(\mathsf{GPK}, k, U_k)$ $(k = 1, \ldots, K)$ are run by the challenger $\mathcal{B}$. GPK, $\{\mathsf{CPK}_d, \mathsf{CAPK}_d | d \in \mathbb{D}\}$ and $\{\mathsf{APK}_k, \mathsf{ACPK}_k | k \in \mathbb{K}\}$ are given to the adversary $\mathcal{A}$.
- $\mathcal{A}$ specifies an index $d^* \in \mathbb{D}$ as the only uncorrupted CA and specifies a set $\mathbb{K}_c \subset \mathbb{K}$ of AAs to be corrupted where $\mathbb{K} \setminus \mathbb{K}_c \neq \emptyset$. Let $\mathbb{D}_c = \mathbb{D} \setminus \{d^*\}$. $\{\mathsf{CMSK}_d | d \in \mathbb{D}_c\}$ and $\{\mathsf{AMSK}_k | k \in \mathbb{K}_c\}$ are given to $\mathcal{A}$.

Key Query Phase 1. User-central-key and user-attribute-key can be obtained by querying the following oracles:

$\mathsf{CKQ}(gid, d)$ where $d = d^*$: $\mathcal{A}$ queries with a pair (gid, d), where gid is a global identifier and $d = d^*$, and obtains the corresponding user-central-key $(\mathsf{ucsk}_{gid,d^*}, \mathsf{ucpk}_{gid,d^*})$.

[1] This is stronger than the static corruption model used in [5,6,13], where the adversary has to specify the authorities to corrupt before seeing the public parameters. But on the other aspect, it is weaker than the model in [13], where the corrupted authorities are set by the adversary.

$\mathsf{AKQ}(att, \{\mathsf{ucpk}_{gid,d}|d \in \mathbb{D}\}, k)$ where $k \in \mathbb{K} \setminus \mathbb{K}_c$: $\mathcal{A}$ queries with (att, $\{\mathsf{ucpk}_{gid,d}|d \in \mathbb{D}\}, k$), where $k \in \mathbb{K} \setminus \mathbb{K}_c$ is the index of an uncorrupted AA, $\{\mathsf{ucpk}_{gid,d}|d \in \mathbb{D}\}$ are gid's user-central-public-keys, and att is an attribute in U_k. The oracle returns a user-attribute-key $\mathsf{uask}_{att,gid}$ or $\perp$ if $\{\mathsf{ucpk}_{gid,d}\}$ are invalid.

Challenge Phase. $\mathcal{A}$ submits two equal-length messages M_0, M_1, and an access policy $\mathbb{A}$. $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$ and sends to $\mathcal{A}$ an encryption of M_β under $\mathbb{A}$.

Key Query Phase 2. $\mathcal{A}$ further queries as in **Key Query Phase 1**.

Guess. $\mathcal{A}$ submits a guess β' for β.

For a gid, the related attribute set is defined as

$$S_{gid} = \{att \mid \mathsf{AKQ}(att, \{\mathsf{ucpk}_{gid,d}|d \in \mathbb{D}\}, k) \text{ is made by } \mathcal{A}\}.$$

$\mathcal{A}$ wins the game if $\beta' = \beta$ under the **restriction** that there is no S_{gid} such that $S_{gid} \cup (\bigcup_{k_c \in \mathbb{K}_c} U_{k_c})$ can satisfy the challenge access policy $\mathbb{A}$. The advantage of $\mathcal{A}$ is defined as $|\Pr[\beta = \beta'] - 1/2|$.

Definition 1. *An MA-CP-ABE system is secure if for all polynomial-time adversary $\mathcal{A}$ in the game above, the advantage of $\mathcal{A}$ is negligible.*

Remarks: We assume that a user with global identifier gid requests for the central key from each CA_d only once, i.e., for each gid there is only one set of user-central-keys, $\{\mathsf{ucpk}_{gid,d}|d \in \mathbb{D}\}$. This is not a restriction, but can help simplify the system description. Using obscure notations such as $\mathsf{ucpk}_{gid,d,t}$ and $S_{gid,d,t}$ where t is a time stamp can remove this assumption. In the security model above, $\mathcal{A}$ has the master secret keys $\{\mathsf{CMSK}_d|d \in \mathbb{D}_c\}$, so the user only needs to query $\mathsf{CKQ}(gid, d^*)$ for getting $(\mathsf{ucsk}_{gid,d^*}, \mathsf{ucpk}_{gid,d^*})$, and the user can generate $\{(\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d})|d \in \mathbb{D}_c\}$ if they are needed for querying AKQ.

3 Background

3.1 Access Policy

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

In ABE, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ contains the authorized sets of attributes. As of previous work in ABE, we focus on monotone access structures in this paper. It is shown in [1] that any monotone access structure can be realized by a linear secret sharing scheme. Here we use the definition from [1,21].

Definition 3 (Linear Secret-Sharing Schemes (LSSS) [21]). *A secret sharing scheme Π over a set of parties $\mathbb{P}$ is called linear (over $\mathbb{Z}_p$) if*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix A called the share-generating matrix for Π. The matrix A has l rows and n columns. For $i = 1, \ldots, l$, the i^{th} row of A is labeled by a party $\rho(i)$ (ρ is a function from $\{1, \ldots, l\}$ to $\mathbb{P}$). When we consider the column vector $\boldsymbol{v} = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $A\boldsymbol{v}$ is the vector of l shares of the secret s according to Π. The share $(A\boldsymbol{v})_i$ belongs to party $\rho(i)$.*

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

3.2 Number-Theoretic Assumptions

Our MA-CP-ABE system works on composite order bilinear groups [4]. Let $\mathcal{G}$ be the group generator, which takes a security parameter λ and outputs $(p_1, p_2, p_3, G, G_T, e)$ where p_1, p_2, p_3 are distinct primes, G and G_T are cyclic groups of order $N = p_1 p_2 p_3$, and $e : G \times G \to G_T$ is a map such that: (1) (Bilinear) $\forall g, h \in G, a, b \in \mathbb{Z}_N, e(g^a, h^b) = e(g, h)^{ab}$, (2) (Non-Degenerate) $\exists g \in G$ such that $e(g, g)$ has order N in G_T. Assume that group operations in G and G_T as well as the bilinear map e are computable in polynomial time with respect to λ. Let G_{p_1}, G_{p_2} and G_{p_3} be the subgroups of order p_1, p_2 and p_3 in G, respectively. Note that for any $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ where $i \neq j$, $e(h_i, h_j) = 1$.

For an element $T \in G$, T can (uniquely) be written as the product of an element of G_{p_1}, an element of G_{p_2}, and an element of G_{p_3}, and they are referred to as the "G_{p_1} part of T", "G_{p_2} part of T" and "G_{p_3} part of T", respectively. In the assumptions below, let $G_{p_1 p_2}$ and $G_{p_1 p_3}$ be the subgroups of order $p_1 p_2$ and $p_1 p_3$ in G, respectively. Similarly, an element in $G_{p_1 p_2}$ can be written as the product of an element of G_{p_1} and an element of G_{p_2}, and an element in $G_{p_1 p_3}$ can be written as the product of an element of G_{p_1} and an element of G_{p_3}.

Assumption 1 (Subgroup decision problem for 3 primes). *[12] Given a group generator $\mathcal{G}$, define the following distribution: $\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}$, $g \xleftarrow{R} G_{p_1}$, $X_3 \xleftarrow{R} G_{p_3}$, $D = (\mathbb{G}, g, X_3)$, $T_1 \xleftarrow{R} G_{p_1 p_2}$, $T_2 \xleftarrow{R} G_{p_1}$. The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 is:*

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 4. *$\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 2. *[12] Given $\mathcal{G}$, define the following distribution:* $\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}$, $g, X_1 \xleftarrow{R} G_{p_1}$, $X_2, Y_2 \xleftarrow{R} G_{p_2}$, $X_3, Y_3 \xleftarrow{R} G_{p_3}$, $D = (\mathbb{G}, g, X_1X_2, X_3, Y_2Y_3), T_1 \xleftarrow{R} G$, $T_2 \xleftarrow{R} G_{p_1p_3}$. *The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 is:*

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 5. *$\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 3. *[12] Given $\mathcal{G}$, define the following distribution:* $\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}$, $\alpha, s \xleftarrow{R} \mathbb{Z}_N$, $g \xleftarrow{R} G_{p_1}$, $X_2, Y_2, Z_2 \xleftarrow{R} G_{p_2}$, $X_3 \xleftarrow{R} G_{p_3}$, $D = (\mathbb{G}, g, g^{\alpha}X_2, X_3, g^sY_2, Z_2)$, $T_1 = e(g, g)^{\alpha s}$, $T_2 \xleftarrow{R} G_T$. *The advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 is:*

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 6. *$\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

4 Our Multi-authority CP-ABE

4.1 Outline

Before describing our construction of MA-CP-ABE, we briefly review the CP-ABE scheme of [11] below, and then outline the ideas behind our construction.

$\mathsf{Setup}(\lambda, U) \rightarrow (\mathsf{PK}, \mathsf{MSK})$. Let G be a bilinear group of order $N = p_1p_2p_3$, and G_{p_i} be the subgroup of order p_i in G. Randomly choose $\alpha, a \in \mathbb{Z}_N$ and $g \in G_{p_1}$. For each attribute $att \in U$, randomly choose $s_{att} \in \mathbb{Z}_N$. Let X_3 be a generator of G_{p_3}. The public parameters are $\mathsf{PK} = (N, g, g^a, e(g, g)^{\alpha}, T_{att} = g^{s_{att}} \ \forall att \in U)$, and the master secret key is $\mathsf{MSK} = (\alpha, X_3)$.

$\mathsf{KeyGen}(\mathsf{MSK}, S, \mathsf{PK}) \rightarrow \mathsf{SK}$. The algorithm randomly chooses $r \in \mathbb{Z}_N$, $R_0, R_0' \in G_{p_3}$, and for each $att \in S$ it randomly picks $R_{att} \in G_{p_3}$. The secret key is: $K = g^{\alpha}g^{ar}R_0$, $L = g^rR_0'$, $K_{att} = T_{att}^rR_{att}$ $\forall att \in S$.

$\mathsf{Encrypt}((A, \rho), \mathsf{PK}, M) \rightarrow CT$. A is an $l \times n$ matrix and ρ maps each row A_x of A to an attribute $\rho(x)$. The algorithm chooses a random vector $\boldsymbol{v} = (s, v_2, \ldots, v_n) \in \mathbb{Z}_N^n$, and for each row A_x of A, it randomly picks $r_x \in \mathbb{Z}_N$. The ciphertext is: $\langle C = M \cdot e(g, g)^{\alpha s}, C' = g^s, C_x = g^{aA_x \cdot \boldsymbol{v}}T_{\rho(x)}^{-r_x}, C_x' = g^{r_x} \ \ \forall x \in \{1, 2, \ldots, l\}\rangle$ along with (A, ρ).

$\mathsf{Decrypt}(CT, \mathsf{PK}, \mathsf{SK}) \rightarrow M$. The algorithm computes constants $\omega_x \in \mathbb{Z}_N$ such that $\sum_{\rho(x) \in S} \omega_x A_x = (1, 0, \ldots, 0)$, then

$$e(C', K) / \prod_{\rho(x) \in S} \big(e(C_x, L) \cdot e(C_x', K_{\rho(x)})\big)^{\omega_x} = e(g, g)^{\alpha s}.$$

Then M can be recovered as $C / e(g, g)^{\alpha s}$.

Now we outline the ideas used in our construction of MA-CP-ABE. We start with building a one-CA-multi-AA system.

One Central Authority and Multiple Attribute Authorities. In the KeyGen algorithm of the underlying CP-ABE system, K and L are uncorrelated to any attributes, and $\forall att \in S$

$$\begin{aligned} K_{att} &= T_{att}^{r} R_{att} \\ &= (g^{s_{att}})^{r} R_{att} = (g^{r})^{s_{att}} R_{att} = (L R_0'^{-1})^{s_{att}} R_{att} = L^{s_{att}} R_0'^{-s_{att}} R_{att} \\ &= L^{s_{att}} R'_{att} \end{aligned}$$

i.e., K_{att} can be computed from L and s_{att} without knowing the value of r. We can get a one-CA-multi-AA system as shown in Fig. 2, where different attribute authorities manage different domains of attributes and a central authority holds the master secret key and generates K and L for users.

While the L is submitted to AAs by the users, the malicious users may launch a collusion attack by submitting the same L. e.g., $L_{Bob} = L_{Tom}$ will allow them put their attribute keys together to decrypt some ciphertexts that they are not authorized to. To defend against this attack, the AAs should verify whether the L is really issued by the CA to the corresponding user. A signature scheme, which is existentially unforgeable under adaptive chosen message attacks (UF-CMA) [8] can be introduced, that is, an adversary cannot generate a valid signature for a new message. Müller, Katzenbeisser, and Eckert [15,16] proposed a similar construction based on Waters' CP-ABE [21]. Their system is selectively secure with non-adaptive key query and the collusion attack above is not considered.

Using signature scheme in multi-authority CP-ABE system was also discussed by Lewko and Waters [13], in a way that, the AAs certify users' identities and their attributes, and a CA issues attribute keys to users according to their identity-attribute certificates. As they mentioned, the CA is demanding as all attribute keys are issued by the CA. Also, the CA will know all the attributes of each user.

Multiple Central Authorities and Multiple Attribute Authorities. In the above one-CA-multi-AA system, the CA can decrypt all ciphertexts because it holds the master secret key α. We use a (D, D) threshold policy to distribute the master secret key to D central authorities to get a multi-CA-multi-AA system as shown in Figure 3. Each central authority (CA_d) publishes a $e(g,g)^{\alpha_d}$ and holds the α_d secretly. The encryptor masks M by $\prod e(g,g)^{\alpha_d s}$. A user gid with attribute set S_{gid} will visit each CA_d to get $(K_{gid,d}, L_{gid,d})$, and then submit $\{L_{gid,d} \mid d \in \mathbb{D}\}$ to related $\{AA_k\}$ where $S_{gid} \cap U_k \neq \emptyset$. For any attribute $att \in S_{gid} \cap U_k$, AA_k will generate $K_{att,gid,d}$ from $L_{gid,d}$ for each $d \in \mathbb{D}$, and issue $\{K_{att,gid,1}, \ldots, K_{att,gid,D}\}$ to gid.

However, this is insecure when some CAs are corrupted, e.g., two malicious users, Bob and Tom, corrupt CA_1. Assume $a_1 \in S_{Tom}$, $a_2 \notin S_{Tom}$, $a_2 \in S_{Bob}$. Normally Tom should not get $K_{a_2,Tom,D}$ to reconstruct $e(g,g)^{\alpha_D s}$ for a ciphertext associated with policy (a_1 **AND** a_2). But Bob can make $L_{Bob,1} = L_{Tom,D}$ because he controls CA_1, then submits $(a_2, L_{Bob,1})$ to AA_K, and AA_K will use $L_{Bob,1}$(actually, $L_{Tom,D}$) to generate $K_{a_2,Bob,1}$ for "Bob", which is actually $K_{a_2,Tom,D}$ for Tom.

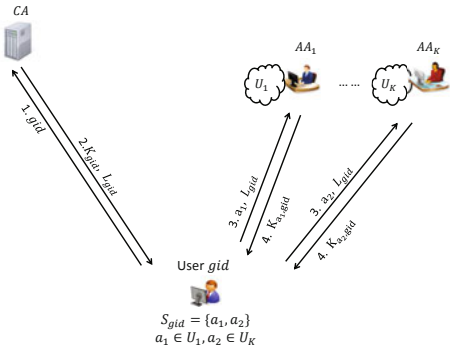

Fig. 2. One-CA-Multi-AA

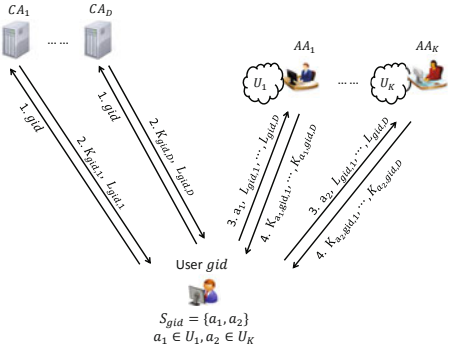

Fig. 3. Multi-CA-Multi-AA

Our solution is to have AA_k check if CA_d has *honestly* generated $L_{gid,d}$. In our construction, when CA_d generates a $L_{gid,d} = g^r R'$, it must generate $\Gamma_{gid,d,k} = V_{k,d}^r R$ for each $k \in \mathbb{K}$ for showing the knowledge of r. This idea is also borrowed from the underlying CP-ABE scheme. In particular, given ($L = g^r R'_0$, $K_{att} = T_{att}^r R_{att}$ $\forall att \in S$) and $T_{att'}$ where $att' \notin S$, an attacker cannot construct $K_{att'} = T_{att'}^r R_{att'}$.

4.2 Construction

$\mathsf{GlobalSetup}(\lambda) \rightarrow (\mathsf{GPK})$. Let G be a bilinear group of order $N = p_1 p_2 p_3$ (3 distinct primes), and G_{p_i} be the subgroup of order p_i in G. The algorithm randomly chooses $g, h \in G_{p_1}$. Let X_3 be a generator of G_{p_3}.
The global public parameter is published as $\mathsf{GPK} = (N, g, h, X_3, \Sigma_{sign})$, where $\Sigma_{sign} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is the description of an UF-CMA secure signature scheme.

$\mathsf{CASetup}(\mathsf{GPK}, d) \rightarrow (\mathsf{CPK}_d, \mathsf{CAPK}_d, \mathsf{CMSK}_d)$. CA_d runs the KeyGen algorithm of Σ_{sign} to generate sign key pair $(\mathsf{SignKey}_d, \mathsf{VerifyKey}_d)$, and chooses a random exponent $\alpha_d \in \mathbb{Z}_N$.
CA_d publishes its public parameter $\mathsf{CPK}_d = e(g,g)^{\alpha_d}, \mathsf{CAPK}_d = \mathsf{VerifyKey}_d$.
CA_d sets its master secret key $\mathsf{CMSK}_d = (\alpha_d, \mathsf{SignKey}_d)$.

$\mathsf{AASetup}(\mathsf{GPK}, k, U_k) \rightarrow (\mathsf{APK}_k, \mathsf{ACPK}_k, \mathsf{AMSK}_k)$. For each $att \in U_k$, AA_k randomly chooses $s_{att} \in \mathbb{Z}_N$ and sets $T_{att} = g^{s_{att}}$. For each $d \in \mathbb{D}$, AA_k randomly chooses $v_{k,d} \in \mathbb{Z}_N$ and sets $V_{k,d} = g^{v_{k,d}}$.
AA_k publishes its public parameter $\mathsf{APK}_k = \{T_{att} | att \in U_k\}, \mathsf{ACPK}_k = \{V_{k,d} | d \in \mathbb{D}\}$.
AA_k sets its master secret key $\mathsf{AMSK}_k = (\{s_{att} | att \in U_k\}, \{v_{k,d} | d \in \mathbb{D}\})$.

$\mathsf{Encrypt}(M, \mathbb{A} = (A, \rho), \mathsf{GPK}, \{\mathsf{CPK}_d | d \in \mathbb{D}\}, \{\mathsf{APK}_k\}) \rightarrow CT$. M is the message to be encrypted, $\mathbb{A}$ is the access policy which is expressed by an LSSS matrix (A, ρ), where A is an $l \times n$ matrix and ρ maps each row A_x of A to an attribute $\rho(x)$. Here it is required that ρ will not map two different rows to a same attribute.

The algorithm chooses a random vector $\boldsymbol{v} = (s, v_2, \ldots, v_n) \in \mathbb{Z}_N^n$, and for each $x \in \{1, 2, \ldots l\}$, it randomly picks $r_x \in \mathbb{Z}_N$. Let $A_x \cdot \boldsymbol{v}$ be the inner product of the x^{th} row of A and the vector $\boldsymbol{v}$. The ciphertext is

$$C = M \cdot \prod_{d=1}^{D} e(g,g)^{\alpha_d \cdot s},\ C' = g^s,$$
$$\{C_x = h^{A_x \cdot \boldsymbol{v}} T_{\rho(x)}^{-r_x}, C'_x = g^{r_x} \mid x \in \{1, 2, \ldots l\}\}$$

along with the access policy $\mathbb{A} = (A, \rho)$.

$\mathsf{CKeyGen}(gid, \mathsf{GPK}, \{V_{k,d} | k \in \mathbb{K}\}, \mathsf{CMSK}_d) \rightarrow (\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d})$. When a user submits his gid to CA_d to request the user-central-key, CA_d randomly chooses $r_{gid,d} \in \mathbb{Z}_N$ and $R_{gid,d}, R'_{gid,d} \in G_{p_3}$, then sets

$$\mathsf{ucsk}_{gid,d} = g^{\alpha_d} h^{r_{gid,d}} R_{gid,d}, \quad L_{gid,d} = g^{r_{gid,d}} R'_{gid,d}.$$

For $k = 1$ to K, CA_d randomly picks $R_{gid,d,k} \in G_{p_3}$ and computes

$$\Gamma_{gid,d,k} = V_{k,d}^{r_{gid,d}} R_{gid,d,k}.$$

CA_d computes $\sigma_{gid,d} = \mathsf{Sign}(\mathsf{SignKey}_d, gid||d||L_{gid,d}||\Gamma_{gid,d,1}|| \ldots ||\Gamma_{gid,d,K})$. Let $\mathsf{ucpk}_{gid,d} = (gid, d, L_{gid,d}, \{\Gamma_{gid,d,k} \mid k \in \mathbb{K}\}, \sigma_{gid,d})$.

$\mathsf{AKeyGen}(att, \{\mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}, \mathsf{GPK}, \{\mathsf{VerifyKey}_d | d \in \mathbb{D}\}, \mathsf{AMSK}_k) \rightarrow \mathsf{uask}_{att,gid}$ or $\perp$. When a user submits his $\{\mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}$ to AA_k to request the user-attribute-key for attribute $att \in U_k$,

1. For $d = 1$ to D, AA_k parses $\mathsf{ucpk}_{gid,d}$ into $(gid, d, L_{gid,d}, \{\Gamma_{gid,d,k} | k \in \mathbb{K}\}, \sigma_{gid,d})$ and checks whether

$$valid \leftarrow \mathsf{Verify}(\mathsf{VerifyKey}_d, gid||d||L_{gid,d}||\Gamma_{gid,d,1}|| \ldots ||\Gamma_{gid,d,K}, \sigma_{gid,d}) \tag{1}$$

$$e(g, \Gamma_{gid,d,k}) = e(V_{k,d}, L_{gid,d}) \neq 1. \tag{2}$$

   If there is any failure, AA_k outputs $\perp$ to user to imply the submitted $\{\mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}$ are invalid.
2. For $d = 1$ to D, AA_k randomly picks $R'_{att,gid,d} \in G_{p_3}$, and sets

$$\mathsf{uask}_{att,gid,d} = (\Gamma_{gid,d,k})^{s_{att}/v_{k,d}} R'_{att,gid,d}. \tag{3}$$

   Note that

$$\begin{aligned}
\mathsf{uask}_{att,gid,d} &= (\Gamma_{gid,d,k})^{s_{att}/v_{k,d}} R'_{att,gid,d} \\
&= (V_{k,d}^{r_{gid,d}} R_{gid,d,k})^{s_{att}/v_{k,d}} R'_{att,gid,d} \\
&= (g^{v_{k,d} \cdot r_{gid,d}} R_{gid,d,k})^{s_{att}/v_{k,d}} R'_{att,gid,d} \\
&= T_{att}^{r_{gid,d}} (R_{gid,d,k})^{s_{att}/v_{k,d}} R'_{att,gid,d}
\end{aligned}$$

   As $(R_{gid,d,k})^{s_{att}/v_{k,d}} R'_{att,gid,d}$ is in G_{p_3} and $R'_{att,gid,d}$ is randomly chosen, we can write

$$\mathsf{uask}_{att,gid,d} = T_{att}^{r_{gid,d}} R_{att,gid,d}. \tag{4}$$

Without knowing the value of $r_{gid,d}$, by running (3), AA_k can compute the value as (4).

3. AA_k outputs user-attribute-key $\mathsf{uask}_{att,gid}$ to user where

$$\begin{aligned}\mathsf{uask}_{att,gid} &= \prod_{d=1}^{D} \mathsf{uask}_{att,gid,d} = \prod_{d=1}^{D} T_{att}^{r_{gid,d}} R_{att,gid,d} \\ &= T_{att}^{\sum_{d=1}^{D} r_{gid,d}} \prod_{d=1}^{D} R_{att,gid,d} \\ &= T_{att}^{\sum_{d=1}^{D} r_{gid,d}} R_{att,gid}\end{aligned} \tag{5}$$

$\mathsf{Decrypt}(CT, \mathsf{GPK}, \{\mathsf{APK}_k\}, \mathsf{DK}_{gid}) \rightarrow M$. The ciphertext CT is parsed into $\langle C, C', \{C_x, C'_x | x \in \{1, 2, \ldots, l\}\}, \mathbb{A} = (A, \rho)\rangle$, and the decryption-key DK_{gid} is parsed into $(\{\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}, \{\mathsf{uask}_{att,gid} | att \in S_{gid}\})$.

The algorithm computes

- $\mathsf{ucsk}_{gid} = \prod_{d=1}^{D} \mathsf{ucsk}_{gid,d} = g^{\sum_{d=1}^{D} \alpha_d} h^{\sum_{d=1}^{D} r_{gid,d}} \prod_{d=1}^{D} R_{gid,d} = g^{\alpha} h^{r_{gid}} R_{gid}$,

  with $\alpha = \sum_{d=1}^{D} \alpha_d$, $r_{gid} = \sum_{d=1}^{D} r_{gid,d}$ and $R_{gid} = \prod_{d=1}^{D} R_{gid,d}$.
- $L_{gid} = \prod_{d=1}^{D} L_{gid,d} = g^{\sum_{d=1}^{D} r_{gid,d}} \prod_{d=1}^{D} R'_{gid,d} = g^{r_{gid}} R'_{gid}$,

  with $R'_{gid} = \prod_{d=1}^{D} R'_{gid,d}$.

Note that $\forall att \in S_{gid}$, $\mathsf{uask}_{att,gid} = T_{att}^{\sum_{d=1}^{D} r_{gid,d}} R_{att,gid} = T_{att}^{r_{gid}} R_{att,gid}$.

If S_{gid} satisfies the access policy (A, ρ), the algorithm computes constants $\omega_x \in \mathbb{Z}_N$ such that $\sum_{\rho(x) \in S_{gid}} \omega_x A_x = (1, 0, \ldots, 0)$. Then it computes

$$e(C', \mathsf{ucsk}_{gid}) \Big/ \prod_{\rho(x) \in S_{gid}} \left(e(C_x, L_{gid}) \cdot e(C'_x, \mathsf{uask}_{\rho(x),gid})\right)^{\omega_x} = e(g, g)^{\alpha s}.$$

While $C = M \cdot \prod_{d=1}^{D} e(g, g)^{\alpha_d \cdot s} = M \cdot e(g, g)^{s \sum_{d=1}^{D} \alpha_d} = M \cdot e(g, g)^{s\alpha}$, M can be recovered from $C / e(g, g)^{\alpha s}$.

In the above system, it is required that an attribute appears at most once in an LSSS matrix (A, ρ). This restriction is crucial to the security proof. As in [11], we call such a system as a One-Use system, and we can use the encoding technique in [11] to extend our system to a Multi-Use system. In Appendix A, we analyze the security of the system above.

5 Extensions

5.1 Large Universe Construction

In the construction in Sec.4.2, the size of the public parameters of AA_k is linear in $|U_k|$. We can modify our scheme to get a large universe construction by using

a technique similar to that in [10]. For each AA_k, let n_k denote the maximum size of the set $S_{gid} \cap U_k$ for any user gid. In addition, we let $H : \{0,1\}^* \to \mathbb{Z}_N$ be a collision-resistant hash function so that we can use arbitrary strings as attributes, and let $\psi : U \mapsto \mathbb{K}$ be a function that maps an attribute to the index of the corresponding attribute authority. The $\mathsf{AASetup}(\mathsf{GPK}, k, U_k)$ algorithm is modified to

$\mathsf{AASetup}(\mathsf{GPK}, k, n_k)$

AA_k chooses $n_k + 1$ random exponents $a_{k,0}, a_{k,1}, \ldots, a_{k,n_k} \in \mathbb{Z}_N$ and sets $F_{k,i} = g^{a_{k,i}} (i = 0, 1, \ldots, n_k)$.

For each $d \in \mathbb{D}$, AA_k randomly chooses $v_{k,d} \in \mathbb{Z}_N$ and sets $V_{k,d} = g^{v_{k,d}}$.

AA_k publishes its public parameter $\mathsf{APK}_k = \{F_{k,i} \mid i = 0, 1, \ldots, n_k\}, \mathsf{ACPK}_k = \{V_{k,d} \mid d \in \mathbb{D}\}$.

AA_k sets its master secret key $\mathsf{AMSK}_k = (\{a_{k,i} \mid i = 0, 1, \ldots, n_k\}, \{v_{k,d} \mid d \in \mathbb{D}\})$.

Let $q_k(x) = \sum_{i=0}^{n_k} a_{k,i} x^i$, $F_k(x) = g^{q_k(x)} = \prod_{i=0}^{n_k} (F_{k,i})^{x^i}$, and for any $att \in \{0,1\}^*$, $s_{att} = q_{\psi(att)}(H(att))$, $T_{att} = F_{\psi(att)}(H(att))$. Then $T_{att} = g^{s_{att}}$. Note that for any $att \in \{0,1\}^*$, the encryptor can compute T_{att} from public parameters, and the corresponding AA_k can compute s_{att} from its master secret key.

5.2 Improving Performance and Robustness

In the construction in Sec.4.2, the trust on each central authority is minimized so that the central authorities could not decrypt any ciphertext unless all central authorities are involved. However, the robustness of the system is limited. Each central authority must remain active because a user must obtain his user-central-keys from each central authority. A threshold policy will be an effective way to balance the trust on each central authority and the robustness of the system.

The Setup phase is executed by a trusted party. The trusted party chooses a random $\alpha \in \mathbb{Z}_N$ and determines a threshold policy (D, Δ) where $1 < D \leq \Delta$, then generates Δ shares $\alpha_1, \alpha_2, \ldots, \alpha_\Delta$. α_d is securely distributed to CA_d to be its master secret key. The trusted party publishes $e(g,g)^\alpha$ and (D, Δ) to global public parameters, and then discards α.

In such a system, the encryptor will mask plaintext M with $e(g,g)^{\alpha s}$. A user needs to visit any D central authorities to obtain his user-central-keys so that he can get his decryption-key.

Only when D central authorities are involved, they can decrypt a ciphetext. The system will work until more than $\Delta - D$ central authorities fail. When $D = \Delta$, it is the system proposed in Sec.4.2.

The detail of such a system will be presented in the full version.

6 Comparison

In Table 2, we compare the single-authority CP-ABE in [11], the multi-authority CP-ABE in [13] and our MA-CP-ABE system. In the table, l is the number of

Table 2. Comparison

	CP-ABE Scheme in [11]	MA-CP-ABE Scheme in [13]	Our MA-CP-ABE
Standard Model	$\surd$	$\times$	$\surd$
Multi-Authority	$\times$	$\surd$	$\surd$
Prevent Decryption by Individual Authorities	$\times$	Partially	$\surd$
Size of Ciphertext	$2l+2$	$3l+1$	$2l+2$
Size of SK	$\|S\|+2$	$\|S\|$	$\|S\|+D(K+2)$
Pairing computation of decryption	$2\|I\|+1$	$2\|I\|$	$2\|I\|+1$
Size of PK	$\|U\|+3$	$2\|U\|$	$\|U\|+3+D$
Large Universe Construction	$\surd$	$\times$	$\surd$

rows of the LSSS matrix (A, ρ), S is the attribute set of the secret key, $|I|$ is the number of rows of (A, ρ) that are used in the decryption, U is the attribute universe, D is the number of CAs, and K is the number of AAs. All the three systems are fully secure, and realize any LSSS access structure.

In [11], the authority can decrypt all ciphertexts; in [13], no authority can decrypt all ciphertexts, but each authority can independently decrypt some ciphertexts; in our MA-CP-ABE scheme, no authority can independently decrypt any ciphertext. While the user and the encryptor will not use the public parameters CAPK_d and ACPK_k, we do not count them in the size of PK. The total size of these keys is $D + D \cdot K$. The size of PK of our system shown in the table is that of the construction in Sec.4.2. For the large universe construction in Sec.5.1, the size of PK is $\sum_{k=1}^{K}(n_k + 1) + 3 + D$ which is not related to the size of U. It is worth noticing that introducing multiple CAs is to prevent some CAs from decrypting ciphertexts. Hence D could be a small value.

7 Conclusion

In this work, we constructed a multi-authority CP-ABE scheme where different domains of attributes are managed by different attribute authorities and no authority can independently decrypt any ciphertext. The proposed system is proved fully secure in the standard model, realizes any monotone access structure, and has almost same efficiency as the underlying CP-ABE scheme. In addition, the proposed system can be extended to support large attribute universe.

References

1. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Ph.D. thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society, Los Alamitos (2007)

3. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
4. 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)
5. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
6. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM Conference on Computer and Communications Security, pp. 121–130. ACM, New York (2009)
7. Cheung, L., Newport, C.C.: Provably secure ciphertext policy abe. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 456–465. ACM, New York (2007)
8. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
9. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded Ciphertext Policy Attribute Based Encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
10. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 89–98. ACM, New York (2006)
11. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (Hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
12. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
13. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)
14. Lin, H., Cao, Z., Liang, X., Shao, J.: Secure threshold multi authority attribute based encryption without a central authority. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 426–436. Springer, Heidelberg (2008)
15. Müller, S., Katzenbeisser, S., Eckert, C.: Distributed attribute-based encryption. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 20–36. Springer, Heidelberg (2009)
16. Müller, S., Katzenbeisser, S., Eckert, C.: On multi-authority ciphetext-policy attribute-based encryption. Bulletin of the Korean Mathematical Society 46(4), 803–819 (2009)
17. 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)

18. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 195–203. ACM, New York (2007)
19. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
20. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
21. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)

A Security Analysis

Let Π denote the main construction, we modify Π to Π' as follows.

– In the $\mathsf{AKeyGen}$ algorithm, it outputs $\mathsf{uask}_{att,gid} = \{\mathsf{uask}_{att,gid,d} | d \in \mathbb{D}\}$ rather than $\mathsf{uask}_{att,gid} = \prod_{d=1}^{D} \mathsf{uask}_{att,gid,d}$. i.e., gid's decryption-key is

$$\begin{aligned}\mathsf{DK}_{gid} =&(\{\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}, \{\mathsf{uask}_{att,gid} \mid att \in S_{gid}\})\\ =&(\{\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}, \{\{\mathsf{uask}_{att,gid,d} | d \in \mathbb{D}\} | att \in S_{gid}\})\\ =&(\{\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d} | d \in \mathbb{D}\}, \{\{\mathsf{uask}_{att,gid,d} | att \in S_{gid}\} | d \in \mathbb{D}\})\\ =&\{(\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d}, \{\mathsf{uask}_{att,gid,d} | att \in S_{gid}\}) | d \in \mathbb{D}\}\\ =&\{\mathsf{usk}_{gid,d} | d \in \mathbb{D}\}\end{aligned}$$

where $\mathsf{usk}_{gid,d} = (\mathsf{ucsk}_{gid,d}, \mathsf{ucpk}_{gid,d}, \{\mathsf{uask}_{att,gid,d} | att \in S_{gid}\})$ is called gid's ***user-key*** related to d.

– In the $\mathsf{Decrypt}$ algorithm,
  1. For $d = 1$ to D, the algorithm uses $\mathsf{usk}_{gid,d}$ to reconstruct $e(g,g)^{\alpha_d s}$:

$$e(C', \mathsf{ucsk}_{gid,d}) / \prod_{\rho(x) \in S_{gid}} \left(e(C_x, L_{gid,d}) \cdot e(C'_x, \mathsf{uask}_{\rho(x),gid,d})\right)^{\omega_x} = e(g,g)^{\alpha_d s}. \tag{6}$$

  2. The algorithm recovers M by

$$M = C / \prod_{d=1}^{D} e(g,g)^{\alpha_d s}. \tag{7}$$

Note that the user and the attacker will get more information in Π', the security of Π' will imply the security of Π. We show the security of Π' in the following.

In the security model, CA_{d^*} is the only uncorrupted central authority and no $S_{gid} \cup (\bigcup_{k_c \in \mathbb{K}_c} U_{k_c})$ can satisfy the challenge access policy. It means that the adversary could not request keys to form a usk_{gid,d^*} to reconstruct $e(g,g)^{\alpha_{d^*} s}$. In our proof, the challenger will respond the adversary as in real attack for all key queries related to $d \neq d^*$. On the key queries related to d^*, we use the proof technique of [11] to provide the answers.

Before we give our proof, we need to define two additional structures: semi-functional ciphertexts and keys. We choose random values $z_{att} \in \mathbb{Z}_N$ associated to the attributes.

Semi-functional Ciphertext. A semi-functional ciphertext is formed as follows. Let g_2 denote a generator of G_{p_2} and c a random exponent modulo N. Besides the random vector $\boldsymbol{v} = (s, v_2, \ldots, v_n)$ and the random values $\{r_x | x \in \{1, 2, \ldots, l\}\}$, we also choose a random vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{Z}_N^n$ and random values $\{\gamma_x \in \mathbb{Z}_N | x \in \{1, 2, \ldots, l\}\}$. Then:

$$C' = g^s g_2^c, \{C_x = h^{A_x \cdot \boldsymbol{v}} T_{\rho(x)}^{-r_x} g_2^{A_x \boldsymbol{u} + \gamma_x z_{\rho(x)}}, C'_x = g^{r_x} g_2^{-\gamma_x} \mid x \in \{1, 2, \ldots, l\}\}.$$

Semi-functional Key. For a gid, a semi-functional user-key usk_{gid,d^*} will take on one of two forms. Exponents $r_{gid,d^*}, \delta, b \in \mathbb{Z}_N$, $\{w_{k,d^*} \in \mathbb{Z}_N | k \in \mathbb{K}\}$, and elements $R_{gid,d^*}, R'_{gid,d^*} \in G_{p_3}, \{R_{att,gid,d^*} \in G_{p_3} | att \in S_{gid}\}, \{R_{gid,d^*,k} \in G_{p_3} | k \in \mathbb{K}\}$ are chosen randomly.

– Type 1:
The user-central-key $(\mathsf{ucsk}_{gid,d^*}, \mathsf{ucpk}_{gid,d^*})$ is formed as

$$\begin{aligned}
&\mathsf{ucsk}_{gid,d^*} = g^{\alpha_{d^*}} h^{r_{gid,d^*}} R_{gid,d^*} g_2^{\delta}, \quad L_{gid,d^*} = g^{r_{gid,d^*}} R'_{gid,d^*} g_2^{b}, \\
&\Gamma_{gid,d^*,k} = V_{k,d^*}^{r_{gid,d^*}} R_{gid,d^*,k} g_2^{b w_{k,d^*}} (k = 1, 2, \ldots K), \\
&\sigma_{gid,d^*} = \mathsf{Sign}(\mathsf{SignKey}_{d^*}, gid || d^* || L_{gid,d^*} || \Gamma_{gid,d^*,1} || \ldots || \Gamma_{gid,d^*,K}), \\
&\mathsf{ucpk}_{gid,d^*} = (gid, d^*, L_{gid,d^*}, \{\Gamma_{gid,d^*,k} | k \in \mathbb{K}\}, \sigma_{gid,d^*}).
\end{aligned}$$

$\forall att \in S_{gid}$, the derived $\mathsf{uask}_{att,gid,d^*}$ is formed as

$$\mathsf{uask}_{att,gid,d^*} = T_{att}^{r_{gid,d^*}} R_{att,gid,d^*} g_2^{b z_{att}}.$$

– Type 2:
The user-central-key $(\mathsf{ucsk}_{gid,d^*}, \mathsf{ucpk}_{gid,d^*})$ is formed as

$$\begin{aligned}
&\mathsf{ucsk}_{gid,d^*} = g^{\alpha_{d^*}} h^{r_{gid,d^*}} R_{gid,d^*} g_2^{\delta}, \quad L_{gid,d^*} = g^{r_{gid,d^*}} R'_{gid,d^*}, \\
&\Gamma_{gid,d^*,k} = V_{k,d^*}^{r_{gid,d^*}} R_{gid,d^*,k} (k = 1, 2, \ldots K), \\
&\sigma_{gid,d^*} = \mathsf{Sign}(\mathsf{SignKey}_{d^*}, gid || d^* || L_{gid,d^*} || \Gamma_{gid,d^*,1} || \ldots || \Gamma_{gid,d^*,K}), \\
&\mathsf{ucpk}_{gid,d^*} = (gid, d^*, L_{gid,d^*}, \{\Gamma_{gid,d^*,k} | k \in \mathbb{K}\}, \sigma_{gid,d^*}).
\end{aligned}$$

$\forall att \in S_{gid}$, the derived $\mathsf{uask}_{att,gid,d^*}$ is formed as

$$\mathsf{uask}_{att,gid,d^*} = T_{att}^{r_{gid,d^*}} R_{att,gid,d^*}.$$

Note that both the semi-functional user-keys of type 1 and type 2 satisfy (1) and (2), and that type 2 is a special case of type 1 with $b = 0$.

When a normal usk_{gid,d^*} and a semi-functional ciphertext, or a semi-functional usk_{gid,d^*} and a normal ciphertext, are used in computation (6), $e(g,g)^{\alpha_{d^*} s}$ is got,

and this value could be used in the computation (7). When a semi-functional usk_{gid,d^*} and a semi-functional ciphertext are used in computation (6), $e(g,g)^{\alpha_{d^*}s} \cdot e(g_2,g_2)^{c\delta - bu_1}$ is got. The additional term $e(g_2,g_2)^{c\delta-bu_1}$ will hinder the computation (7). We call a semi-functional user-key of type 1 *nominally semi-functional* if $c\delta - bu_1 = 0$.

The security of Π' relies on Assumptions 1, 2, 3. We use a hybrid argument over a sequence of games. The first game $\mathbf{Game}_{\mathsf{Real}}$ is the real security game. In the final game $\mathbf{Game}_{\mathsf{Final}}$, all user-keys related d^*, $\{\mathsf{usk}_{gid,d^*}\}$, are semi-functional of type 2 and the ciphertext is a semi-functional encryption of a random message, independent of the two messages provided by $\mathcal{A}$.

$\mathbf{Game}_{\mathrm{Real}}$. The challenge ciphertext is normal. All CKQs are answered with normal user-central-key. All AKQs are answered with user-attribute-key generated by running the normal $\mathsf{AKeyGen}$ algorithm.

$\mathbf{Game}_0$. The challenge ciphertext is semi-functional. All CKQs are answered with normal user-central-key. All AKQs are answered with user-attribute-key generated by running the normal $\mathsf{AKeyGen}$ algorithm.

Let q denote the number of CKQ made by $\mathcal{A}$. For j from 1 to q, we consider the following games:

$\mathbf{Game}_{j,1}$. In this game, the challenge ciphertext is semi-functional. The first $j-1$ CKQs are answered with semi-functional user-central-key of type 2; the j^{th} CKQ is answered with semi-functional user-central-key of type 1; and the remaining CKQs are answered with normal user-central-key. All AKQs are answered with user-attribute-key generated by running the normal $\mathsf{AKeyGen}$ algorithm.

$\mathbf{Game}_{j,2}$. In this game, the challenge ciphertext is semi-functional. The first $j-1$ CKQs are answered with semi-functional user-central-key of type 2; the j^{th} CKQ is answered with semi-functional user-central-key of type 2; and the remaining CKQs are answered with normal user-central-key. All AKQs are answered with user-attribute-key generated by running the normal $\mathsf{AKeyGen}$ algorithm.

$\mathbf{Game}_{\mathrm{Final}}$. In this game, the challenge ciphertext is a semi-functional encryption of a random message, independent of the two messages provided by the adversary. All CKQs are answered with semi-functional user-central-key of type 2. All AKQs are answered with user-attribute-key generated by running the normal $\mathsf{AKeyGen}$ algorithm.

Note that in all the games, all AKQs are answered with user-attribute-key generated by running normal $\mathsf{AKeyGen}$ algorithm. In the proofs, we will show that the derived $\mathsf{uask}_{att,gid,d^*}$ is decided by the corresponding user-central-key $(\mathsf{ucsk}_{gid,d^*}, \mathsf{ucpk}_{gid,d^*})$, i.e., if $(\mathsf{ucsk}_{gid,d^*}, \mathsf{ucpk}_{gid,d^*})$ is semi-functional of type 1 (respectively, type 2), then the derived $\mathsf{uask}_{att,gid,d^*}$ is also semi-functional of type 1 (respectively, type 2). Consequently, usk_{gid,d^*} is decided by the corresponding $(\mathsf{ucsk}_{gid,d^*}, \mathsf{ucpk}_{gid,d^*})$ as well. Note that in $\mathbf{Game}_0$ all user-central-keys related to d^* are normal and in $\mathbf{Game}_{q,2}$ all user-central-keys related to d^* are semi-functional of type 2. It means that in $\mathbf{Game}_0$ all user-keys usk_{gid,d^*}

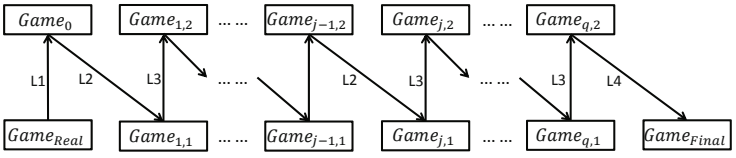

Fig. 4. Indistinguishable games. L1 denotes Lemma 1, and so on.

are normal and in $\mathbf{Game}_{q,2}$ all user-keys usk_{gid,d^*} are semi-functional of type 2. We show these games are indistinguishable in the following four lemmas (see Fig. 4), the proofs of which will appear in the full version.

Lemma 1. *Given a UF-CMA signature scheme Σ_{sign}, suppose there exists a poly-time algorithm $\mathcal{A}$ such that $\boldsymbol{Game}_{Real}Adv_{\mathcal{A}} - \boldsymbol{Game}_0 Adv_{\mathcal{A}} = \epsilon$. We can construct a poly-time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Lemma 2. *Use $\boldsymbol{Game}_{0,2}$ to denote $\boldsymbol{Game}_0$. Given a UF-CMA signature scheme Σ_{sign}, suppose there exists a poly-time algorithm $\mathcal{A}$ such that $\boldsymbol{Game}_{j-1,2}Adv_{\mathcal{A}} - \boldsymbol{Game}_{j,1}Adv_{\mathcal{A}} = \epsilon$. We can construct a poly-time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 2.*

Lemma 3. *Given a UF-CMA signature scheme Σ_{sign}, suppose there exists a poly-time algorithm $\mathcal{A}$ such that $\boldsymbol{Game}_{j,1}Adv_{\mathcal{A}} - \boldsymbol{Game}_{j,2}Adv_{\mathcal{A}} = \epsilon$. We can construct a poly-time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Lemma 4. *Given a UF-CMA signature scheme Σ_{sign}, suppose there exists a poly-time algorithm $\mathcal{A}$ such that $\boldsymbol{Game}_{q,2}Adv_{\mathcal{A}} - \boldsymbol{Game}_{Final}Adv_{\mathcal{A}} = \epsilon$. We can construct a poly-time algorithm $\mathcal{B}$ with advantage $\frac{\epsilon}{D}$ in breaking Assumption 3.*

Theorem 1. *If the signature scheme Σ_{sign} is UF-CMA secure and Assumptions 1, 2, and 3 hold, then our MA-CP-ABE scheme is secure.*

Proof. If Assumptions 1, 2 and 3 hold, and the signature scheme Σ_{sign} is UF-CMA secure, then we have shown by the previous lemmas that the real security game is indistinguishable from $\mathbf{Game}_{\mathsf{Final}}$, in which the value of β is information-theoretically hidden from the adversary. Hence the adversary can not attain a non-negligible advantage in breaking Π', which implies the adversary can not attain a non-negligible advantage in breaking our MA-CP-ABE scheme Π. □