Information Sciences 180 (2010) 2618–2632

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Secure threshold multi authority attribute based encryption without a central authority ☆

Huang Lin, Zhenfu Cao *, Xiaohui Liang, Jun Shao

Department of Computer Science and Engineering, Shanghai Jiao Tong University, China

ARTICLE INFO

Article history:
Received 12 December 2008
Received in revised form 6 February 2010
Accepted 9 March 2010

Keywords:
Threshold multi authority ABE
Without a central authority

ABSTRACT

An attribute based encryption scheme (ABE) is a cryptographic primitive in which every user is identified by a set of attributes, and some function of these attributes is used to determine the ability to decrypt each ciphertext. Chase proposed the first multi authority ABE scheme which requires a fully trusted central authority who has the ability to decrypt each ciphertext in the system. This central authority would endanger the whole system if it is corrupted. This paper provides a threshold multi authority fuzzy identity based encryption (MA-FIBE) scheme without a central authority for the first time. An encrypter can encrypt a message such that a user could only decrypt if he has at least d_k of the given attributes about the message for at least $t+1$, $t \leqslant n/2$ honest authorities of all the n attribute authorities in the proposed scheme. This paper considers a stronger adversary model in the sense that the corrupted authorities are allowed to distribute incorrect secret keys to the users. The security proof is based on the secrecy of the underlying distributed key generation protocol and joint zero secret sharing protocol and the standard decisional bilinear Diffie–Hellman assumption. The proposed MA-FIBE could be extended to the threshold multi authority attribute based encryption (MA-ABE) scheme, and both key policy based and ciphertext policy based MA-ABE schemes without a central authority are presented in this paper. Moreover, several other extensions, such as a proactive large universe MA-ABE scheme, are also provided in this paper.

1. Introduction

The primitive idea of identity based encryption (IBE) was first proposed by Shamir [20] in 1984. Many identity based cryptographic concepts [2,13,15,21,22] have been proposed since then. In an IBE system, a user is identified by a unique string, the sender has to know the identity of the receiver before encrypting a message for this user. However, this would not always be a realistic scenario since the sender might only know the "attribute" of the receivers rather than the exact identity of the receiver sometimes. For instance [4], in a secure database of an intelligence agency, one may specify a certain document can be accessed by agents in the counterspy program. In this situation, it is much more natural to encrypt to the single attribute "counterspy", instead of an enumerative list of all agents in the program. Fuzzy identity based encryption, a generalization of IBE, was first proposed by Sahai and Waters [19] (denoted as SW05) in 2005 to deal with this circumstance. In the FIBE system, the user is identified by a certain attribute set, and the ciphertext is encrypted under another attribute set. The user is able to decrypt the ciphertext when the intersection of the above two attribute sets is larger than a certain

☆ This is the full version of the accepted paper "Secure threshold multi authority attribute based encryption without a central authority" of 9th INDOCRYPT, 2008.

* Corresponding author.
E-mail address: zfcao@cs.sjtu.edu.cn (Z. Cao).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.03.004

preset threshold. For example, a sender might encrypt a message which could be decrypted by those who are in any two of the following three programs: "anti-drug" program, "anti-terror" program, and "counterspy" program. In the FIBE system proposed by SW05, there would be an authority, monitoring these three attributes, each of which is responsible for distributing secret keys to the users after verifying whether the users are indeed in the claimed program.

However, there are commonly three departments each of which is responsible for one program and thus monitors the corresponding attribute in practice. The benefit of three independent departments is noticeable: first, the other authorities could still be trusted even when certain authorities in the system are corrupted. In other words, the trust on one single authority is reduced and distributed through all the existing authorities. Second, it would reduce the burden of the authority since the task of monitoring all these attributes and distributing secret keys corresponding to these attributes is now shared by the other authorities. Therefore, the following open problem is presented in SW05: is it possible to construct an attribute encryption scheme in which many different authorities operate simultaneously, each handing out secret keys for a different set of attributes.

1.1. Previous work

To our best knowledge, there is only one existing multi authority attribute based encryption scheme proposed by Chase [3]. Her scheme allows any polynomial number of independent authorities to monitor attributes and distribute secret keys. An encryption chooses, for each authority, a number d_k and a set of attributes. He can then encrypt a message such that a user could only decrypt if he has at least d_k of the given attributes for each authority k. Chase's scheme mainly employs the following *two techniques*: The first is to require that every user has a global identifier (*GID*), and the second is to use a fully trusted central authority.

The global *GID* is used to prevent a collusion attack between different users. More specifically, a user who only has enough secret keys from a certain set of authorities might collude with another user who only has enough keys from the rest authorities to decrypt a ciphertext. The global *GID* could be considered as a randomness embedding to each user's secret keys. This technique is also adopted in this paper.

The secret keys for each user could be considered as an evaluation of a function on *GID* in Chase's scheme. This evaluation is ephemeral and decoupled from the master secret key y_0 but the ability to decrypt is required to depend on the users' attribute rather than their individual *GID* (this is what distinguishes ABE from traditional IBE). The central authority is the second tool used in Chase's scheme in order to guarantee the above property. The central authority is responsible to provide a final secret key to integrate the secret keys from the other attribute authorities such that the decryption process would be independent of *GID*. The central authority would be able to decrypt each ciphertext in the Chase's system because it masters the system secret key. In other words, now that the central authority has to be fully trusted, the trust is not totally distributed through all the authorities. The security of the whole system would be totally broken if the central authority is corrupted. Furthermore, it would also increase the computation and communication cost to run and maintain such a fully trusted authority in the system.

There is a possible attack, other than the collusion attack, which is ignored in Chase's paper. Any qualified user could not be able to decrypt the ciphertext in her system even if there exists only one authority who does not distribute the correct secret keys.

Many ABE schemes have been proposed until now, such as [17,16,10,1,4,9].

1.2. Our contribution

This paper focuses on removing the central authority from multi authority ABE scheme. It is difficult to remove the central authority while preventing the collusion attack and keeping the decryption process independent of the identifier of each user. As indicated in the above, it is the central authority which is responsible to integrate the secret keys from the other attribute authorities in Chase's scheme, and thus to integrate these secret keys without the central authority would be an obstacle in our system. Another difficulty is that the integration must be done at the last decryption step as Chase's scheme did. The integration aims to emancipate the users from the restriction of individual identifier, which means this integration should not be completed before the final decryption step because the collusion attack might be possible if the users are free from the bondage of *GID*.

In this paper, we replace the pseudo random function used in Chase's scheme by a polynomial. After that, we apply the key distribution technique and the joint zero secret sharing technique [8] to the proposed scheme.[1] We will show that all the above difficulties could be overcome by these simple modifications and the first secure threshold multi authority fuzzy identity based encryption scheme without a central authority could be constructed.

In the proposed scheme, an encrypter can encrypt a message such that a user could only decrypt if he has at least d_k of the given attributes about the message for at least $t+1,\ t \leqslant n/2$ honest authorities. All the authorities only need to communicate with each other without revealing any private information to the others during the initialization (or the periodical update step when it comes to the proactive multi authority attribute scheme), and they could operate independently and

[1] Compared to the previous schemes [11,6,5,7] which also adopt these two techniques as their basic tools, these two techniques are used in a rather different manner in our construction which we call it parallel manner (this would be further explained in Section 2.3), we considered this as one of our novelties.

keep autonomous in the left steps of the scheme. In other words, the whole relatively costly process is transparent to the users, and would not cause any additional inconvenience to them. The trust is truly distributed between each authority in the proposed system, and the system does not need to pay any extra communication or computation cost since the central authority is removed. The security of the proposed scheme could be reduced to the secrecy of the underlying distributed key distribution protocol (DKG) [8] and joint zero secret sharing protocol (JZSS) and standard decisional BDH assumption.

Our adversary is allowed to distribute incorrect secret keys deliberately. This fault behavior is possible since the trust laid upon each authority is now reduced and each one could be corrupted in any way. Our proposed system always contains at least $t+1$ honest authorities who would distribute correct secret keys (due to the restriction for the $(t-1,\ n)$-threshold adversary, here $t \leqslant n/2$), and those who obtain enough keys from these honest authorities could still decrypt the respective ciphertext. In other words, the faulty behavior of authorities would not disrupt our proposed system.[2]

The proposed threshold MA-FIBE scheme could be extended to multi authority attribute based encryption scheme. In fact, four secure MA-ABE schemes without a central authority, two of which are key policy based and the other two of which are ciphertext policy based, are also presented in this paper.

Several other extensions of the proposed scheme are introduced in this paper. How to convert a large universe attribute scheme into a proactive scheme is presented here. A proactive multi authority attribute scheme implies the secret keys held by the authorities could be updated without changing the public parameters of the whole system. This would result in a more convenient system for the users in the sense that all the encrypters do not need to regenerate their ciphertexts created in the original system before the renewal. This also enhances the security level of the system because the adversary has to attack the system successfully during a shortened period of interval compared with the adversary to the underlying multi authority scheme.

1.3. Organization

At first, some preliminaries will be introduced in Section 2. Then, the security model would be given in Section 3. In Section 4, the construction of threshold MA-FIBE scheme without a central authority is provided. After that, Section 5 would indicate how to convert the proposed MA-FIBE scheme into MA-ABE scheme with similar properties. Some extensions are shown in Section 6. How to add the proactive property to the proposed multi authority large universe attribute scheme is introduced in this section.

2. Preliminaries

2.1. Decisional BDH assumption

The bilinear maps [16] is crucial to our construction, some basic facts related to bilinear maps are introduced here.

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p. Let g be a generator of G and e be a bilinear map, $e: G_1 \times G_1 \to G_2$. The bilinear map e has the following properties:

1. *Bilinearity*: for all $u,\ v \in G_1$ and $a,\ b \in Z_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. *Non degeneracy:* $e(g,g) \neq 1$.

G_1 is a bilinear group if the group operation in G_1 and the bilinear map $e: G_1 \times G_1 \to G_2$ are both efficiently computable. Note that the map e is symmetric since $e(g^a, g^b) = e(g,g)^{ab} = e(g^b, g^a)$.

The security proof of the proposed scheme relies on decisional bilinear Diffie–Hellman assumption, its definition is shown as follow [3]:

Definition 1 (*Decisional bilinear Diffie–Hellman assumption*). Let $a,\ b,\ c, z \leftarrow Z_q$ be chosen randomly, G be a group of prime order q and generator g. The decisional BDH assumption is that no probabilistic polynomial time algorithm $\mathcal{B}$ can distinguish the tuple $(A = g^a, B = g^b, C = g^c, e(g,g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, e(g,g)^z)$ with more than a negligible advantage. The advantage of $\mathcal{B}$ is

$$|Pr[\mathcal{B}(A, B, C, e(g,g)^{abc})] - Pr[\mathcal{B}(A, B, C, e(g,g)^z)] = 0|$$

where the probability is taken over the random choice of the generator g, the random choice of $a,\ b,\ c,\ z$ in Z_p, and the random bits consumed by $\mathcal{B}$.

2.2. Communication model

The communication model of this paper is basically the same to that of [8]. Our computation model is composed of a set of n attribute authorities which could be modeled as a PPT machine. Related definitions can be found in [14].

[2] Chase's scheme would not guarantee this, and even qualified users do not have the ability to decrypt if even one authority distributes incorrect secret keys.

They are connected by a complete network of private (i.e. untappable) point-to-point channels. In addition, all the authorities have access to a dedicated broadcast channel. By dedicated we mean that if the player P_i broadcast a message, it will be received by every other players and recognized as coming from P_i.

A partially synchronous communication model is assumed in this paper. In other words, messages sent on either a point-to-point or the broadcast channel are received by their recipients within some fixed time bound. A failure of a communication channel to deliver a message within this time bound can be treated as a failure of the sending authorities. For simplicity of discussion, we assume that the authorities are equipped with synchronized clocks.

2.3. Distributed key generation protocol (DKG) and joint zero secret sharing protocol (JZSS)

Distributed key generation protocol (denoted as DKG as shown in Appendix A) and joint zero secret sharing protocol (denoted as JZSS) are two vital components of the proposed construction. A more concrete introduction to these two techniques could be found in Sections 4.4 and 4.5 of [7]. Note that distributed key generation protocol is denoted as Joint-Exp-RSS and joint zero secret sharing scheme is denoted as Joint-Uncond-Secure-ZSS for short in this paper. The original construction of DKG protocol could be found in [8].

In the DKG protocol, the players collectively choose shares corresponding to a (t,n)-secret sharing of a random value σ. At the end of such a protocol each player P_i has a share σ_i, where $(\sigma_1,\ldots,\sigma_n) \overset{(t,n)}{\rightarrow} \sigma$, and σ is uniformly distributed over the interpolation field, and the protocol also outputs g^σ as a public key. There's *no trusted dealer*, who masters the secret σ in the regular secret sharing scheme, in the DKG protocol. This secret can only be reconstructed through the cooperation of at least $t+1$ honest players. This property plays an important role during the proposed construction since it is the reason why the central authority could be *removed*.

The secrecy [8] of DKG protocol could be defined as: no information on the secret σ can be learned by a (t,n)-threshold static adversary which corrupts at most t players except for what's implied by the value $y = g^\sigma \ mod \ p$. The simulation of the DKG protocol could be found in Fig. 4 of [7].

The JZSS protocol is similar to the DKG protocols but instead the players collectively choose shares corresponding to a known value zero *without a trusted dealer*. The JZSS protocol can be deduced from the joint unconditionally secure random secret sharing protocol with a slight modification as shown in [7]. The players would obtain "zero-shares" from the JZSS protocol since all these shares form a (t,n)-secret sharing of 0. By adding such "zero-shares" to existing shares of some secret σ, one obtains a randomization of the shares of σ without changing the secret. This is the typical way to use the JZSS protocol.

However, JZSS protocol is used in a different manner from the typical way during our basic constructions. m JZSS protocols along with two DKG protocols are executed independently during the initialization to hide non-constant terms of the secret polynomial of each authority from the rest authorities while ensuring that all these terms would combine to 0. Informally, the JZSS protocol and DKG protocol are traditionally used in a *serial* way, while they run *parallel* in our basic construction, and a polynomial is used to tie up these independent protocols. As a result, we consider a $(t-1,n)$-threshold static adversary attacking the JZSS protocol, then the secrecy of the JZSS protocol could be stated as: the share σ_i of the honest players are information theoretically secure to the $(t-1,n)$-threshold static adversary. Note that the adversary should not be allowed to corrupt t players as in the DKG protocol because the adversary would be able to compute the shares of the honest players since the shared secret 0 is known to the adversary. The proof of the secrecy of the JZSS protocol is straightforward.

2.4. Access trees

- *Access Tree.* Let $\mathcal{T}$ represent an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold gate. num_x is the number of children of a node x and $k_x, 0 < k_x \leqslant num_x$ is its threshold gate. $\Phi_{\mathcal{T}}$ denotes the set of all the non-leaf nodes in the tree $\mathcal{T}$, and $\Psi_{\mathcal{T}}$ denotes the set of all the non-leaf nodes at depth $d-1$, where d is the depth of $\mathcal{T}$, and the root of a tree is fixed to be at level 0 here. $\mathrm{att}(x)$ denotes the attribute associated with the leaf node x.
- *Satisfying an Access Tree.* Let $\mathcal{T}$ be an access tree with root r. Denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at the node x. Hence $\mathcal{T}$ is the same as $\mathcal{T}_r$. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, it is denoted as $\mathcal{T}_x(\gamma) = 1$. $\mathcal{T}_x(\gamma)$ is computed recursively as follows: if x is a non-leaf node, evaluate $\mathcal{T}_x(\gamma)$ for all children of node x. $\mathcal{T}_x(\gamma)$ returns 1 iff at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 iff $\mathrm{att}(x) \in \gamma$.
- *Universal Access Tree.* Given a pair of integer values (d, num), define a complete num-ary tree $\mathcal{T}$ of depth d, where each non-leaf node has a threshold value num. Next, $num-1$ new leaf nodes assigned with *dummy* attributes are attached to each non-leaf node x while the threshold value num is left intact. The resultant tree $\mathcal{T}$ is called the universal tree. All the leaf nodes are empty here.
- *Bounded Access Tree.* We say that $\mathcal{T}'$ is a (d, num)-bounded access tree if it has depth $d' \leqslant d$, and each non-leaf node in $\mathcal{T}'$ exhibits a cardinality at most num. Cardinality refers to the number of children of a node.
- *Normal Form Access Tree.* A (d, num) normal form access tree means it has depth $d' = d$, and all the leaves in $\mathcal{T}'$ are at depth d. Any (d, num)-bounded access tree $\mathcal{T}'$ could be converted to the (d, num) normal form in a top-down manner, starting from its root node r'. For a node x at level l_x in $\mathcal{T}'$. If the depth d_x of the subtree $\mathcal{T}'_x$ is less than $d - l_x$, then insert a vertical chain of $(d - l_x - d_x)$ nodes, each of which has cardinality 1 and threshold 1, between x and its parent node $\mathrm{parent}(x)$. Repeat the procedure recursively for each child of x.

– *Map Between Access Trees.* A map between the nodes of a (d, num) normal form access trees $\mathcal{T}'$ and a universal access tree $\mathcal{T}$ is defined in a top-down manner. First, the root of $\mathcal{T}'$ is mapped to the root of $\mathcal{T}$. For a node x' in $\mathcal{T}'$ which is mapped to x in $\mathcal{T}$, let $z'_1, \ldots, z'_{num_{x'}}$ be the child nodes of x', ordered according to their index values. Then, for each child $z'_i, i \in [1, num_{x'}]$ of x' in $\mathcal{T}'$, set the child z_i with index value $\text{index}(z'_i)$ of x in $\mathcal{T}$ as the map of z'. This procedure is performed recursively, until each node in $\mathcal{T}'$ is mapped to a corresponding node in $\mathcal{T}$. To capture the above node mapping procedure, a public function $\text{map}(\cdot)$ that takes a node in $\mathcal{T}'$ as input and returns the corresponding node in $\mathcal{T}$.

3. Adversary and security model

The proposed multi authority system consists of n attribute authorities. Each attribute authority monitors a set of n_k attributes in the universe of attributes $\mathcal{U}$. The universe $\mathcal{U}$ consists of N attributes where $N = \sum_{k=1}^{n} n_k$.

The proposed multi authority scheme consists of the following algorithms:

Setup: A randomized algorithm takes as input the security parameter κ and outputs all the system public parameters *PK* and the secret key *SK* for attribute authorities.

Secret key distribution (SKD) $(SK, GID, \mathcal{A}_u)$*:* A randomized algorithm takes as input the authorities's secret key *SK*, a user *u*'s *GID*, and a set of attributes $\mathcal{A}_u^k$ in the authority AA_k's domain (We will assume that the user's claim of these attributes has been verified before this algorithm is run, $\mathcal{A}_u = \left\{\mathcal{A}_u^k, k = 1, \ldots, n\right\}$). Output a secret key D_u for the user *u*.

Encryption (ENC) $(\mathcal{A}_C, M, PK)$: A randomized algorithm takes as input an attribute set $\mathcal{A}_C$ of a message *M*, the system public parameters *PK* and outputs the ciphertext C.

Decryption(DNC) (C, D_u): A deterministic algorithm takes as input a ciphertext C, which was encrypted under an attribute set $\mathcal{A}_C$ and decryption key D_u. Output a message m if $\left|\mathcal{A}_C^k \bigcap \mathcal{A}_u^k\right| \geqslant d_k$ for at least $t+1$ honest[3] attribute authorities.

As in [10,3], our scheme is proved secure in the selective attribute set model (SAS), in which the adversary must select the challenge attribute set he wishes to attack before receiving the parameters of the system.

Our adversary is static, i.e. chooses up to $t-1$, $t \leqslant n/2$ corrupted authorities, and it is denoted as $(t-1, n)$ threshold adversary. At last, the authority controlled by the adversary here is allowed to distribute incorrect secret keys to the users.

Consider the following game:

Setup The $(t-1, n)$-threshold adversary sends a list of attribute sets $\mathcal{A}_{C^*} = \mathcal{A}_{C^*}^1, \ldots, \mathcal{A}_{C^*}^n$, one for each authority. He must also provide a list of corrupted authorities (up to $t-1$).

The simulator generates system parameters (which means it has to simulate the DKG protocol twice and the JZSS protocol m times). The adversary should be given the following information after the generation of system parameters: the public keys for all the honest authorities, and the secret keys for all corrupted authorities.

Secret key queries (SKD) (the total number of these queries are m for each authority).

The requirements for the queries are:

for each *GID* and any $t+1$ authorities the adversary query, there must be at least one honest authority k from which the adversary requests fewer than d_k of the attributes given in $\mathcal{A}_{C^*}^k$

the adversary never queries the same authority twice with the same *GID*.[4]

Challenge The adversary sends two messages M_0 and M_1. The challenger chooses a bit x, computes the encryption of M_x for attribute set $\mathcal{A}_{C^*}$, and sends this ciphertext C^* to the adversary.

More secret key queries The adversary may make more secret key queries subject to the requirements described in the above.

Guess The adversary outputs a guess x'. The adversary succeed if $x = x'$.

Definition 2. A multi authority attribute scheme is selective attribute sets (SAS) CPA secure if there exists a negligible function v such that, in the above game any $(t-1, n)$-threshold adversary will succeed with probability at most $\frac{1}{2} + v(\kappa)$ (κ denotes the system security parameter).

4. Threshold MA-FIBE scheme without a central authority

4.1. Primitive idea

A large universe construction of FIBE is given by Sahai and Waters [19], and the FIBE construction adopted in this section could be considered as the respective variant construction with large size of public parameters. Although this modified FIBE scheme might be implied in their later proposed ABE scheme, this construction was never clearly stated neither in [10] nor [19]. Different from Chase's scheme, each authority AA_k selects a polynomial $a_{k,0} + a_{k,1}x + \cdots + a_{k,m}x^m$ rather than a pseudo random function to evaluate on a user's *GID*. Only in this way could we manage to use DKG and JZSS protocol to enable all authorities to share a secret without letting any authority to master the system master key. That's why the proposed scheme is able to eliminate the central authority. The system secret key a_0 is generated by executing a DKG protocol and thus a_0 is

[3] Honest means the authority would distribute correct secret keys to the users.

[4] This is feasible using the method discussed in Section 2 of [3].

not known to any authority, and each authority $AA_k,\ k = 1,\ldots,n$ would have the share $a_{k,0}$ about a_0. In order to generate the polynomial $a_{k,0} + a_{k,1}x + \cdots + a_{k,m}x^m,\ AA_k,\ k = 1,\ldots,n$ needs to decide how to pick up the other coefficients $a_{k,j},\ k = 1,\ldots,n,\ j = 1,\ldots,m$. However, the system has to guarantee the ability to decrypt independent of *GID*. In consequence, JZSS protocol is adopted to generate the other coefficients $a_{k,j},\ k = 1,\ldots,n,\ j = 1,\ldots,m$ since $a_{1,j},\ldots,a_{n,j},\ j = 1,\ldots,m$ forms a (t,n) threshold polynomial secret sharing of 0 due to JZSS protocol, and thus each user's *GID* would have nothing to do with the final decryption.

If the users obtain enough secret keys from honest authorities AA_k, then they would be able to compute the respective share $e(g,g_2)^{p_k(0)s} = e(g,g_2)^{(a_{k,0}+a_{k,1}GID+\cdots+a_{k,m}GID^m)s}$ in the proposed scheme. As stated in the above, $(a_{1,0},\ldots,a_{n,0})$ form a random (t,n) threshold polynomial secret sharing of a_0 due to DKG protocol, and $(a_{1,k},\ldots,a_{n,k})_{k=1,\ldots,m}$ form m random (t,n) threshold polynomial secret sharing of 0. When the $t+1$ shares combine to the corresponding secrets (which means the users have enough secret keys from at least $t+1$ honest authorities), *GID* would disappear in the final combining polynomial, and they would have $e(g,g_2)^{a_0 s} = e(g_1,g_2)^s$ to decrypt the ciphertext. In other words, the users are only liberated from *GID* in the final decryption step.

4.2. The proposed scheme

Fix prime order groups $G_1,\ G_2$, bilinear map $e: G_1 \times G_1 \to G_2$, and generator $g \in G_1$. Define the universe of attributes $\mathcal{U} = \{1,2,\ldots,N\}$. Each attribute authority AA_k monitors a set of n_k attributes in this universe, where $N = \sum_{k=1}^{n} n_k$.

1. *Setup*
   (a) Execute DKG protocol twice and JZSS protocol independently m times, the shares of attribute authority $AA_i,\ i = 1,\ldots,n$ obtained from the execution of these two protocols constitute the set of partial secret keys for each authority. In consequence, there are totally $m+2$ secret keys for each authority, as shown Table 1: The secret keys $Sk_i,\ i = 1,\ldots,n$ for the respective AA_i are $a_{i,0},\ldots,a_{i,m},b_{i,m+1}$. The respective public keys of two DKG protocols are $g^{a_0},\ g^{b_0}$, which would be treated as a part of system public keys.According to the DKG protocol, we have $a_0 = \sum_{l=1}^{t+1} a_{k_l,0}\gamma_{k_l},\ b_0 = \sum_{l=1}^{t+1} b_{k_l,m+1}\gamma_{k_l}$. We also have $0 = \sum_{l=1}^{t+1} a_{k_l,j}\gamma_{k_l},\ j = 1,2,\ldots,m$ according to the JZSS protocol.[5]
   (b) Each authority $AA_k,\ k = 1,\ldots,n$ also needs to randomly choose another set of secret keys $t_{k,1},\ldots,t_{k,n_k}$ from Z_q, each of which corresponds to the jth attribute mastered by the authority AA_k in the universe $\mathcal{U}$. The corresponding public keys are $T_{k,1} = g^{t_{k,1}},\ldots,T_{k,n_k} = g^{t_{k,n_k}}$.
   (c) the secret key SK_k for each authority $AA_k,\ k = 1,\ldots,n$ are

   $$a_{k,0}, a_{k,1}, a_{k,2}, \ldots, a_{k,m}, b_{k,m+1}, t_{k,1}, \ldots, t_{k,n_k}$$

   The secret keys of all authorities form the set of secret keys $SK = \{SK_1,\ldots,SK_n\}$.The published public parameters PK are $g,\ g_1 = g^{a_0},\ g_2 = g^{b_0},\ \{T_{k,1},\ldots,T_{k,n_k}\}_{k=1,\ldots,n}$.
2. *SKD* $(SK, GID, \mathcal{A}_u)$The user randomly selects a *GID* from Z_q and presents it to the authority, and the authority checks whether it is valid (according to the method mentioned in Footnote 4), if it is valid then continue, else reject.For $AA_k,\ k = 1,\ldots,n$, the SKD process is shown as follows:
   For each *GID*, the authority would first randomly select $p_k(x)$ (where $p_k(x)$ is a $d_k - 1$ degree polynomial) satisfied with $p_k(0) = a_{k,0} + a_{k,1}GID + \cdots + a_{k,m}GID^m$. The secret keys D_u for u are $D_u = \{D_{k,j}\}_{j\in\mathcal{A}_u^k, k=1,\ldots,n}$, where $D_{k,j} = \{g_2^{p_k(j)/t_{k,j}}\}_{j\in\mathcal{A}_u^k}$.
3. *ENC* $(\mathcal{A}_C, M, PK)$
   Choose a random value $s \in Z_q$. For the attribute set $\mathcal{A}_C = \left\{\mathcal{A}_C^k\right\},\ k \in \{1,\ldots,n\}$, generate the ciphertext $C = \left\{\mathcal{A}_C,\ E = e(g_1,g_2)^s \cdot M \| 0^{l'},\ \left\{E_{k,j} = T_{k,j}^s\right\}_{j\in\mathcal{A}_C^k,\forall k}\right\}$ (k corresponds to the authority AA_k, $\mathcal{A}_C^k$ denotes the attribute set of the ciphertext monitored by authority AA_k. In order to check whether a decryption is valid, prior to encryption, we append M trailing 0s denoted $0^{l'}$ to message $M \in \{0,1\}^l$).
4. *DEC*(C, D_u)
   (1) For $AA_k,\ k = 1,2,\ldots,t+1$ run the following two steps.[6]
      (a) For d_k attributes $i \in \mathcal{A}_C^k \bigcap \mathcal{A}_u$, compute $e(E_{k,i}, D_{k,i}) = e\left(g^{t_{k,j}s}, g_2^{p_k(j)/t_{k,j}}\right) = e(g,g_2)^{p_k(j)s}$.
      (b) Interpolate to find

      $$e(g,g_2)^{p_k(0)s} = e(g,g_2)^{(a_{k,0}+a_{k,1}GID+\cdots+a_{k,m}GID^m)s}$$

   (2) $\prod_{k=1}^{t+1} e(g,g_2)^{p_k(0)s\gamma_k} = e(g,g_2)^{\sum_{k=1}^{t+1}(\gamma_k a_{k,0}+\gamma_k a_{k,1}GID+\cdots+\gamma_k a_{k,m}GID^m)s} = e(g,g_2)^{a_0 s} = e(g_1,g_2)^s$
   (3) $E / e(g_1,g_2)^s = M \| 0^{l'}$

[5] $\gamma_j = \prod_{k\in\{1,2,\ldots,t+1\}, j\neq k} \frac{0-k}{j-k}$ denotes the Lagrange interpolation coefficients for the set $\{1,2,\ldots,t+1\}$. For simplicity of discussion and without lost of generality, the first $t+1$ authorities are assumed to be honest and distribute correct secret keys here.

[6] For ease of exposition and without lost of generality, we assume the first $t+1$ authorities are honest and distribute correct secret keys and the decryptor has enough secret keys from all these honest authorities.

A possible collusion attack and the restriction of m:

The adoption of polynomial $a_{k,0} + a_{k,1}GID + a_{k,2}GID^2 + \cdots + a_{k,m}GID^m$ as the master key while generating the secret key for a user might cause a collusion attack. Assume $m+1$ users who only have qualified secret keys for an attribute set $\mathcal{A}_C^k$ of a ciphertext E (as shown in the **ENC** algorithm), which means $e(g,g_2)^{s\cdot\left(a_{k,0}+a_{k,1}GID+a_{k,2}GID^2+\cdots+a_{k,m}GID^m\right)}$ can be computed by all of them. Consequently, $e(g,g_2)^{s\cdot\left(a_{k,0}+a_{k,1}x+a_{k,2}x^2+\cdots+a_{k,m}x^m\right)}$ can be interpolated for $m+1$ evaluations of $e(g,g_2)^{s\cdot\left(a_{k,0}+a_{k,1}x+a_{k,2}x^2+\cdots+a_{k,m}x^m\right)}$ on different *GID*s are available. Consider another user with identifier GID' which only has eligible secret keys for the other t attribute sets $\mathcal{A}_C^{k_l},\ l=1,\ldots,t$ of the same ciphertext, then they would be able to obtain $e(g,g_2)^{s\cdot\left(a_{k_l,0}+a_{k_l,1}GID'+a_{k_l,2}GID'^2+\cdots+a_{k_l,m}GID'^m\right)},\ l=1,\ldots,t$. If this user colludes with the above $m+1$ users with satisfied keys from $\mathcal{A}_k$, then they will have the ability to interpolate $e(g_1,g_2)^s$. All they have to do is to evaluate $e(g,g_2)^{s\cdot\left(a_{k,0}+a_{k,1}x+a_{k,2}x^2+\cdots+a_{k,m}x^m\right)}$ on GID' and do the rest interpolation, which means all of them would be able to open the message even though none of them is qualified to do this. That's how the possible collusion attack works.

In order to prevent this attack in practice, we have to restrict the number of *GID*s no more than m in the basic constructions. That's why the number of secret key queries is restricted to be equal to m in the security model. As we can see here, m is a crucial parameter for the security of the basic constructions, and there is a tradeoff between the security property and the efficiency of the initialization step. Although the system could accommodate more *GID*s as m grows, more JZSS protocols need to be executed during the initialization while m is bigger, and thus the computation and communication cost increases.

4.3. Security proof

Theorem 1. *The proposed threshold MA-FIBE scheme is SAS CPA secure in the standard model and $(t-1,n)$-threshold adversary model if decisional BDH assumption holds in (G_1,G_2) and the underling DKG protocol and JZSS protocol are secure.*

Proof. See Appendix E. □

5. Threshold MA-ABE without a central authority

Since MA-ABE is actually the generalization of the MA-FIBE, we will show how to employ the method introduced in [10,9] to convert the proposed MA-FIBE scheme into MA-ABE schemes without a central authority in this section. Goyal et al. [10] first introduced the methodology to construct a key policy based ABE construction from a FIBE scheme. After that, how to convert a key policy based ABE scheme to ciphertext policy based ABE scheme is presented in ICALP2008 [9].

In the following section, we will show all these methodologies could be applied to convert the proposed MA-FIBE construction into the counterpart MA-ABE construction without a central authority. The major difference between the proposed MA-FIBE and MA-ABE schemes lies in the SKD algorithm and the rest algorithms. At first, two key policy based MA-ABE schemes without a central authority will be given in this section. The first scheme corresponds to the construction for access trees in [10] which we denote as key policy construction for small universe here, and the other one corresponds to the key policy large universe construction. Then, two ciphertext policy based MA-ABE schemes without a central authority are also briefly introduced in this section.

5.1. Key policy construction for small universe

1. *Setup* The setup process is the same to that of the MA-FIBE scheme.
2. *SKD* $(SK, GID, \mathcal{A}_u)$
   The selection of *GID* is the same to that of MA-FIBE. For each *GID*, set *MK* as $MK = y_{k,u} = a_{k,0} + a_{k,1}GID + \cdots + a_{k,m}GID^m$. Employ the **Key Generation**(τ_k, MK) algorithm as shown in Appendix C (τ_k is the access tree of AA_k). For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Choose a polynomial $q_r(x)$ for the root node of the access tree τ_k of authority AA_k such that $q_r(0) = MK$, then select d_r other points of the polynomial q_r randomly to define it completely. For any other node x, set $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ and choose d_x other points randomly to completely define q_x. Once the polynomials have been decided for the leaf node x, the authority could distribute the secret key for each user which is $D_{k,x} = \left\{g_2^{q_x(0)/t_{kj}}\right\}_{j\in\mathcal{A}_u^k}$, here j denotes the respective attribute for the leaf x.
   The secret keys D_u for u are $\{D_{k,x}\}_{j\in\mathcal{A}_u^k,\ k=1,\ldots,n}$.
3. *ENC* $(\mathcal{A}_C, M, PK)$
   The algorithm is same to that of the MA-FIBE scheme.
4. *DEC* (C, D_u)
   (1) For $AA_k,\ k=1,2,\ldots,t+1$ run the following two steps.[7]
   (a) For each attributes $j\in\mathcal{A}_C^k\bigcap\mathcal{A}_u^k$, compute $e(E_{k,j},D_{k,x}) = e(g^{t_{kj}s}, g_2^{q_x(0)/t_{kj}}) = e(g,g_2)^{q_x(0)s}$.

[7] We make the same assumption to Footnote 6.

(b) The left steps run the first decryption algorithm in [10], i.e. run **DecryptNode** algorithm on the root of the access tree τ_k (as shown in Appendix D). At each level of the tree, if the decryptor has successfully computed **DecryptNode** for sufficient children, then it combines the results to obtain $e(g,g_2)^{q_x(0)s}$ for the parent node x. Finally, if the decryptor has the attributes set which satisfies the tree τ_k, then it can compute $e(g,g_2)^{q_r(0)s} = e(g,g_2)^{y_{k,u}s} = e(g,g_2)^{(a_{k,0}+a_{k,1}GID+\cdots+a_{k,m}GID^m)s}$.

(2) $\prod_{k=1}^{t+1} e(g,g_2)^{y_{k,u}s\gamma_k} = e(g,g_2)^{\sum_{k=1}^{t+1}(\gamma_k a_{k,0}+\gamma_k a_{k,1}GID+\cdots+\gamma_k a_{k,m}GID^m)s} = e(g,g_2)^{a_0 s} = e(g_1,g_2)^s$

(3) $E/e(g_1,g_2)^s = M||0^l$

5.2. Key policy construction for large universe

1. *Setup*
   (a) This step is the same to that of MA-FIBE scheme.
   (b) Each authority $AA_k,\ k=1,\ldots,n$ also needs to randomly choose $t_{k,1},\ldots,t_{k,n_k+1}$ from G_1, then define a function $T_k(x) = g_2^{x^{n_k}} \prod_{i=1}^{n_k+1} t_{k,n_k+1}^{\Delta_{i,\{1,2,\ldots,n_k+1\}}(x)}$, $t_{k,1},\ldots,t_{k,n_k+1}$ would be published as part of public keys.
   (c) the secret key SK_k for each authority $AA_k,\ k=1,\ldots,n$ are

   $a_{k,0}, a_{k,1}, a_{k,2}, \ldots, a_{k,m}, b_{k,m+1}$

   The secret keys of all authorities form the set of secret keys $SK = \{SK_1,\ldots,SK_n\}$.

   The published public parameters PK are $g,\ g_1 = g^{a_0},\ g_2 = g^{b_0},\ \{t_{k,1},\ldots,t_{k,n_k+1}\}_{k=1,\ldots,n}$.
2. *SKD* $(SK, GID, \mathfrak{A}_u)$
   The selection of GID is the same to that of MA-FIBE.
   For $AA_k,\ k=1,\ldots,n$, the SKD process is shown as follows:
   (a) For each GID, set MK as $MK = y_{k,u} = a_{k,0} + a_{k,1}GID + \cdots + a_{k,m}GID^m$. Apply **Key Generation**(τ_k, MK) algorithm in the similar way to that in the SKD algorithm of the first MA-ABE scheme, the polynomials for the leaf node x would be determined, the authority could distribute the secret key $D_{k,x} = \left\{g_2^{q_x(0)} T_k(j)^{r_{k,x}}\right\}_{j\in\mathfrak{A}_u^k}$ for each user u. we have $j = att(x)$ here, which denotes the corresponding attribute for the leaf x, and $r_{k,x}$ is a random number selected by the authority AA_k from Z_q.
   (b) The authority also computes $R_{k,x} = \{g^{r_{k,x}}\}_{j\in\mathfrak{A}_u^k}$.
   (c) The above secret keys constitute the corresponding secret keys D_u for the user,

   $D_u = \{D_{k,x}, R_{k,x}\}_{j\in\mathfrak{A}_u^k and j=\mathrm{att}(x)}, \quad k=1,\ldots,n$

3. *ENC* $(\mathfrak{A}_C, M, PK)$
   Choose a random value $s\in Z_q$. For the attribute set $\mathfrak{A}_C = \left\{\mathfrak{A}_C^k\right\}_{k\in\{1,\ldots,n\}}$, generate the ciphertext $C = \{\mathfrak{A}_C,\ E = e(g_1,g_2)^s \cdot M||0^{l'},\ E' = g^s,\ \{E_{k,j} = T_k(j)^s\}_{j\in\mathfrak{A}_C^k,\ \forall k}\}$.

4. *DEC* (C, D_u)
   For $AA_k,\ k=1,2,\ldots,t+1$
   For each k, for each attribute $j\in\mathfrak{A}_C^k\bigcap\mathfrak{A}_u^k$, compute $\frac{e(D_{k,x},E')}{e(R_{k,x},E_{k,j})} = \frac{e\left(g_2^{q_x(0)}T_k(j)^{r_{k,x}},g^s\right)}{e\left(g^{r_{k,x}},T_k(j)^s\right)} = e(g,g_2)^{q_x(0)s}$.

   The left steps are the same to that of the first MA-ABE scheme given in the above section.

5.3. Ciphertext policy construction for small universe

Let G_1 be a bilinear group of prime order q, and let g be a generator of G_1. In addition, let $e: G_1\times G_1\to G_2$ denote the bilinear map. A security parameter κ will determine the size of groups. We also define the Lagrange coefficient $\Delta_{i,S}$ for $i\in Z_q$ and a set S of elements in Z_q : $\Delta_{i,S}(x) = \prod_{j\in S, j\neq i}\frac{x-j}{i-j}$. We will associate each attribute with a unique element in Z_q^*. Our construction follows:

Setup (d, num) This algorithm takes as input two system parameters, namely, (a) the maximum tree depth d, and (b) the maximum node cardinality num. The algorithm proceeds as follows. Define the universe of real attribute $\mathcal{U} = \{1,\ldots,n*n\}$, and kth authority $AA_k,\ k\in\{1,2,\ldots,n\}$ monitors a set $\mathcal{U}_k$ of n attributes $\mathcal{U}_k = \{(k-1)*n+1,\ldots,(k-1)*n+n\}$. A $(num-1)$-sized universe of dummy attributes $\mathcal{U}^* = \{n*n+1,\ldots,n*n+num-1\}$ is also defined. Next, define a (d, num)-universal access tree $\mathcal{T}_k$ for each authority AA_k as explained in Section 2. In the sequel, d, num, $\mathcal{U}_k$, $\mathcal{U}^*$ $\mathcal{T}_k$ will be assumed as implicit input to all the procedures.

1. Execute DKG protocol twice and JZSS protocol m times independently. The shares of each authority AA_k from the execution of DKG and JZSS protocol form the authority's partial secret keys $Sk_k = \{a_{k,0}, a_{k,1}, a_{k,2}, \ldots, a_{k,m}, b_{k,m+1}\}$ as shown Table 1. According to the DKG protocol and JZSS protocol, we have $0 = \sum_{l=1}^{t+1} a_{k_l,j}\Delta_{k_l,S}(0),\ j\in\{1,\ldots,m\},\ a_0 = \sum_{l=1}^{t+1} a_{k_l,0}\Delta_{k_l,S}(0)$, and $b_0 = \sum_{l=1}^{t+1} b_{k_l,m+1}\Delta_{k_l,S}(0)$. S denotes $\{k_1,\ldots,k_{t+1}\}$ here.

2. Now, for each real attribute $j \in \mathcal{U}_k$, choose a set of $|\Psi_{\mathcal{T}_k}|$ numbers $\{t_{j,x}\}_{x\in\Psi_{\mathcal{T}_k}}$ uniformly at random from Z_q. Furthermore, for each dummy attribute $j \in \mathcal{U}^*$, choose a set of $|\Phi_{\mathcal{T}_k}|$ numbers $\{t^*_{j,x}\}_{x\in\Phi_{\mathcal{T}_k}}$ uniformly at random from Z_q.
3. Finally, the public keys PK are:

$$g,\quad g_1 = g^{a_0},\quad g_2 = g^{b_0},\quad \left\{\left\{T_{j,x} = g^{t_{j,x}}\right\}_{j\in\mathcal{U}_k, x\in\Psi_{\mathcal{T}_k}}, \left\{T^*_{j,x} = g^{t^*_{j,x}}\right\}_{j\in\mathcal{U}^*, x\in\Phi_{\mathcal{T}k}}\right\}_{k=1}^{n}$$

The secret key SK_k for each authority AA_k, $k \in \{1,\ldots,n\}$ contains:

$$\{t_{j,x}\}_{j\in\mathcal{U}_k, x\in\Psi_{\mathcal{T}_k}}, \{t^*_{j,x}\}_{j\in\mathcal{U}^*, x\in\Phi_{\mathcal{T}_k}}, Sk_k = \{a_{k,0}, a_{k,1}, a_{k,2}, \ldots, a_{k,m}, b_{k,m+1}\}$$

The set of the above secret keys $\{SK_k\}_{k=1}^n$ constitutes the secret key SK.

Key generation $(SK = \{SK_k\}_{k=1}^n, GID, \gamma)$ For each user with an identifier GID and an attribute set $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ (γ_k denotes the user's satisfied attribute set which is monitored by authority $\mathcal{A}_k$), authority $\mathcal{A}_k$ chooses a random polynomial q_x for each non-leaf node x in the universal access tree $\mathcal{T}_k$. These random polynomials are chosen in following top-to-down manner, starting from the root r. For each x, set the degree c_x of the polynomial q_x to be one less than the threshold value, i.e. $c_x = num - 1$. Now, for root node r, set $q_r(0) = a_{k,0} + a_{k,1}GID + a_{k,2}GID^2 + \cdots + a_{k,m}GID^m$ and choose c_r other points of the polynomial q_r randomly to define it completely. For any other non-leaf node x, set $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ and choose c_x other points randomly to completely define q_x. Once the polynomials have been decided, give the following secret values to the user:

$$D_k = \left(\left\{D_{j,x} = g_2^{\frac{q_x(j)}{t_{j,x}}}\right\}_{j\in\gamma_k, x\in\Psi_{\mathcal{T}k}}, \left\{D^*_{j,x} = g_2^{\frac{q_x(j)}{t^*_{j,x}}}\right\}_{j\in\mathcal{U}^*, x\in\Phi_{\mathcal{T}_k}}\right)$$

The set of above secret values $\{D_k\}_{k=1}^n$ forms the decryption key D.

Encryption $(M, PK, \mathcal{T}')$ This algorithm encrypts a message $M \in G_2$ with an access tree $\mathcal{T}'$. As stated in Section 2, the threshold value of the root node of $\mathcal{T}'$ is $t+1$, and the root node has n sub access trees $\{\mathcal{T}'_k\}_{k=1}^n$, each of which is a (d, num)-bounded tree. For each sub access tree $\mathcal{T}'_k$, assign real attributes from $\mathcal{U}_k$ to the leaf nodes. The root node of sub access tree $\mathcal{T}'_k$ is denoted as r'_k.

Now, to be able to encrypt the message M with the sub access tree $\mathcal{T}'_k$, the encrypter first converts it to normal form (if required). Then, a map $\text{map}(\cdot)$ would be defined to map the access tree $\mathcal{T}'_k$ to the universal access tree $\mathcal{T}_k$ as explained in Section 2.3. Finally, for each non-leaf node x in $\mathcal{T}'_k$, choose an arbitrary $(num - k_x)$-sized set ω_x of dummy child nodes of $\text{map}(x)$ in $\mathcal{T}_k$.

Let $f(j,x)$ be a boolean function such that $f(j,x) = 1$ if a real attribute $j \in \mathcal{U}_k$ is associated with a leaf child of node $x \in \Psi_{\mathcal{T}'_k}$, and 0 otherwise. Now, choose a random value $s \in Z_q$ and publish the ciphertext E as:

$$\mathcal{T}', E' = M\cdot e(g_1,g_2)^s, \left\{E_k = \left(\left\{E_{j,x} = T^s_{j,\text{map}(x)}\right\}_{x\in\Psi'_{\mathcal{T}k}, j\in\mathcal{U}_k: f(j,x)=1}, \left\{E^*_{j,x} = T^{*s}_{j,\text{map}(x)}\right\}_{j=\text{att}(z): z\in\omega_x, x\in\Phi_{\mathcal{T}'_k}}\right)\right\}_{k=1}^{n}$$

Decryption $(E, D = \{D_k\}_{k=1}^n)$ For each authority AA_k, run the $\text{DecryptNode}(E_k, D_k, r'_k)$ algorithm as in [9], i.e. run this algorithm on the root of the respective sub access tree $\mathcal{T}'_k$. For each leaf node x in $\mathcal{T}'_k$, let $j = \text{att}(x)$ and w be the parent of x. If $j \in \gamma_k$, this algorithm computes $e(D_{j,\text{map}(w)}, E_{j,w}) = e\left(g_2^{\frac{q_{\text{map}(w)}(j)}{t_{j,\text{map}(w)}}}, g^{s\cdot t_{j,\text{map}(w)}}\right) = e(g,g_2)^{s\cdot q_{\text{map}(w)}(j)}$. At the other levels of $\mathcal{T}'_k$, if we have successfully computed DecryptNode for at least k_x non-leaf children, then we could use DecrytDummy algorithm from [9] to compute on the other $num - k_x$ dummy nodes chosen by the encrypter to fix $e(g,g_2)^{s\cdot q_{\text{map}(x)}(0)}$ for parent node x.

Finally, if $\mathcal{T}'_k(\gamma_k) = 1$, then the $\text{DecryptNode}(E_k, D_k, r'_k)$ algorithm would output $e(g,g_2)^{s\cdot q_{\text{map}(r'_k)}(0)} = e(g,g_2)^{s\cdot q_{r'_k}(0)} = e(g,g_2)^{s\cdot(a_{k,0}+a_{k,1}GID+\cdots+a_{k,m}GID^m)}$. Otherwise the algorithm returns $\perp$.

For the first $t+1$ sub access trees $\{\mathcal{T}'_k\}_{k=1}^{t+1}$,[8] compute $\prod_{k=1}^{t+1} e(g,g_2)^{q_{r'_k}(0)s\Delta_{k,S}(0)} = e(g,g_2)^{\sum_{k=1}^{t+1}(a_{k,0}+a_{k,1}GID+\cdots+a_{k,m}GID^m)s\Delta_{k,S}(0)} = e(g,g_2)^{a_0 s} = e(g_1,g_2)^s$. S denotes the set $\{1,2,\ldots,t+1\}$ here. At last, compute $M = \frac{E'}{e(g_1,g_2)^s}$.

5.4. Ciphertext policy construction for large universe

For space limitation, the multi authority bounded ciphertext policy ABE scheme for large universe (MA-BCP-ABE-LU) would only be briefly introduced here. Large universe allows us to use arbitrary strings as attributes in the system. In consequence, there's no need to define the universe of real attributes. The **Setup** algorithm would need to fix the maximum number of leaf child nodes of a node in an access tree we can encrypt under, denoted as v. Define a (d, num) universal access tree T and a $(num - 1)$ sized universe of dummy attributes. Set $w = |\Psi_T| * v + |\Phi_T| * (num - 1)$. Step 1 would be run in the

[8] For ease of exposition and without lost of generality, we assume the first $t+1$ authorities to be honest to distribute correct secret keys for the users and the users obtain satisfied keys for the respective policies with the ciphertext.

same way as in the **Setup** algorithm of the small universe construction. In step 2, each authority $\mathcal{A}_k$ selects uniformly $t_{k,1}, t_{k,2}, \ldots, t_{k,w+1}$ from G_1 at random to define $F_k(X) = g_2^{X^w} \prod_{i=1}^{w+1} t_{k,i}^{\Delta_{i,W}(X)}$. W denotes the set $\{1, 2, \ldots, w+1\}$ here. As a result, the secret key SK_k for authority $\mathcal{A}_k$, $k \in \{1, \ldots, n\}$ is: $a_{k,0}, a_{k,1}, a_{k,2}, \ldots, a_{k,m}, b_{k,m+1}$. The public keys PK are: g, $g_1 = g^{a_0}$, $g_2 = g^{b_0}$, $\{t_{k,1}, t_{k,2}, \ldots, t_{k,w+1}\}_{k=1}^n$. Note that the choice of w is to make sure that we can deal with the dummy attributes in the same way as we deal with the real attributes using the random function $F_k(X)$. The universe of dummy attributes is defined to make the choice of dummy attributes easier in the **Key Distribution** and the **Encryption** algorithms.

The rest algorithms are basically the combination of bounded ciphertext policy ABE scheme with the trick presented by GPSW06 except the interpolation step in the decryption algorithm.

5.5. Analysis of these MA-ABE Schemes

The above proposed MA-ABE schemes can also be proven SAS CPA secure under the decisional BDH assumption as stated in Theorem 1. The proofs are straightforward and neglected here. The major steps follow the proofs of Theorem 1 concerning the basic MA-FIBE scheme as shown in Appendix E. Since the ABE scheme rather than the FIBE scheme is adopted here, then the respective reduction about the part of ABE scheme would follow the proof shown in [10,9].

6. Extension

6.1. Proactive multi authority attribute scheme: loosing the restriction of m

Why proactive The above mentioned MA-ABE schemes are only suitable for attribute based system with finite users of relatively static attributes since the proposed constructions could only accommodate at most m identifiers. However, the computation cost of each authority during the setup might be too huge to tolerate if m is such a large number when we apply these schemes to the systems accommodating large number of users characterized with frequently changed attributes. But, if we could separate this heavy task into several parts, then the authorities only need to accomplish a small proportion of the whole task one time and thus the computation cost would be acceptable to these authorities.

A straightforward solution is to simply re-initialize the whole system periodically. But, this might lead to certain inconvenience for the users. To simply update the secret keys and public keys of the authorities each period implies that the secret keys obtained from the updated system cannot be used to decrypt all the ciphertext generated before the renewal. All these *old* secret keys would be obsolete after the renewal, and the encrypter needs to regenerate their ciphertext using the updated public parameters in order to fit in the new system.

Proactive secret sharing (PSS), which was first introduced by Herzberg etc. [12] in 1995, provides a useful tool to construct a proactive attribute based system in which the authorities' secret keys could be updated periodically without any modification to the authorities' respective public keys. Thus, the inconvenience due to the periodical renewal would be reduced to some extent.

What's PSS protocol We outline the properties of PSS protocol in Appendix B. There are two vital phases of the PSS protocol: **PSS.Init** and **PSS.Update**. During **PSS.Init** phase, the DKG protocol would be executed to generate a share σ_i' of a uniformly distributed secret σ for each player and publish g^σ as the public key. The JZSS protocol would be executed during **PSS.Update** phase to randomize each player's share by adding a new share σ_i to σ_i'. Since σ_i, $i = 1, \ldots, n$ form a (t,n)-secret sharing of 0, then the resultant shares $\sigma_i' + \sigma_i$, $i = 1, \ldots, n$ still form a secret sharing of σ due to the linearity of polynomial secret sharing. Previous shares would be obsolete and safely erased. Informally, we could consider the PSS protocol as a *chimera* of the DKG protocol and the JZSS protocol.

The security of the PSS protocol is proven under the threshold mobile adversary model as shown in [11]. The PSS protocol is able to provide semantic security relative to any auxiliary information as shown in Theorem 3 of Appendix B. The so-called (t,n)-threshold mobile adversary, modeled by a probabilistic polynomial time algorithm, who can *statically*, i.e. at the beginning of the life time of the scheme, schedule up to $t < n/2$ arbitrarily corrupted authorities, independently for each period. The computationally bounded adversary is assumed to attack the system for only polynomially-many time periods.

The PSS protocol is used for constructing a proactive public key system [11,5,18] after its invention. An additional step of proactive public key system compared with the PSS protocol is a function computation (denoted as *FunctionCom*) phase. Time is divided into time periods which are determined by the common global clock in the proactive system. Each time period consists of a short *Update* phase (or *Init* phase at the beginning of the system), and a *FunctionCom* phase, in which the servers perform the intended secret key operation using their current sharing of σ. Informally, we could consider the proactive public key system introduced in [11] as a *chimera* of the PSS protocol and a certain function computation operation.

How to be proactive We adopt two DKG protocols and m JZSS protocols to generate the secret keys $\{SK_i\}_{i=1}^n$ and the respective public key (as shown in Table 1.) in the **Setup** algorithm of the basic constructions. The primitive idea of constructing the PSS protocol could be borrowed to design a proactive large universe MA-ABE (P-MA-ABE-LU) scheme, which means $m+2$ JZSS protocols would be executed independently during the *Update* phase to renewal the secret keys $SK_i = \{a_{i,0}, a_{i,1}, a_{i,2}, \ldots, a_{i,m}, b_{i,m+1}\}\}_{i=1}^n$ in the similar way to that of the PSS protocol. In other words, we would run the JZSS protocols $m+2$ times to generate $\{\sigma_{i,0}, \sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,m}, \sigma_{i,m+1}\}_{i=1}^n$ and add these shares to the original secret keys to obtain the updated secret keys. Note that, different from the common proactive public key system which employs only one PSS

protocol in a straightforward way, our P-MA-ABE-LU scheme uses two PSS protocols and m variants of the PSS protocol. The variant PSS protocol differs from the original protocol in the sense that we run the JZSS protocol instead of the DKG protocol during the **PSS.Init** phase. For ease of exposition, we denote the original PSS protocol as DKG-PSS protocol and the variant protocol as JZSS-PSS protocol. The security of JZSS-PSS protocol can be proven under $(t-1,n)$-threshold mobile adversary model as shown in Theorem 3 of Appendix B. The secret keys SK_i of the honest authorities are information theoretically secure to the mobile adversary who corrupts at most $t-1$ authorities during the execution of the JZSS-PSS protocol. Informally, the proposed P-MA-ABE-LU scheme could be considered as a *chimera* of two DKG-PSS protocols and m JZSS-PSS protocols and function computation operations.

The P-MA-ABE-LU scheme also could be divided into three phases: *Init*, *Update*, and *FuctionCom* phase as in the traditional proactive public key system. The *Init* phase executes the **Setup** algorithm of the underlying MA-ABE scheme, and the *FuctionCom* phase contains the rest three algorithms of the MA-ABE scheme: **SKD**, **ENC**, and **DEC** algorithm. A counter initialized as 0 would be given to the authorities at the beginning (or after an *Update* phase), the counter would increase by one if a valid *GID* is used. The system would enter the *Update* phase and execute P-MA-ABE-LU.**Update** algorithm when m *GID*s are used up in the **SKD** algorithm. The P-MA-ABE-LU.**Update** algorithm is described in Table 2 of Appendix F.

The P-MA-ABE-LU.**Update** algorithm updates secret keys $\{a_{k,0},\ a_{k,1},\ a_{k,1},\ \ldots,\ a_{k,m},\ b_{k,m+1}\}_{k=1}^{n}$ and all the public keys remain unchanged. As stated in Section 4.2, more than m *GID*s might endanger the system. Intuitively, the information obtained by the adversary during one period related to the random function $e(g,g_2)^{s\cdot\left(a_{k,0}+a_{k,1}x+a_{k,2}x^2+\cdots+a_{k,m}x^m\right)}$ would be useless for the attack to this function during the other periods since all the coefficients of $a_{k,0}+a_{k,1}x+a_{k,2}x^2+\cdots+a_{k,m}x^m$ would be updated before m *GID*s are used up (which means the adversary does not have the ability to launch the collusion attack in each period) in the proactive system. In consequence, the collusion attack would not be feasible in any period of the P-MA-ABE-LU scheme.

A secure proactive P-MA-ABE-LU scheme is defined as follows:

Definition 3. We call a 5-tuple of algorithms **Setup**, P-MA-ABE-LU.**Update**, **SKD, ENC, DEC** a SAS CPA secure *P-MA-ABE-LU* scheme if no polynomial time $(t-1,n)$-threshold mobile adversary, after running **Setup** algorithm once, and **SKD** algorithm, **ENC** algorithm on input messages which the adversary adaptively chose, and P-MA-ABE-LU.**Update** algorithm polynomially-many times, can win the SAS game[9] (as described in Section 3) with non negligible advantage.

The following theorem can be proven.

Theorem 2. *Under DBDH assumption, if the following conditions hold:*

- *the underlying JZSS-PSS protocol and DKG-PSS protocol are secure (as described in* Theorem 3*).*
- *(***Setup, SKD, ENC,DEC***) is a SAS CPA secure multi authority large universe attribute based encryption scheme.*

then the resultant P-MA-ABE-LU scheme **Setup** *, P-MA-ABE-LU.***Update***,* **SKD, ENC, DEC** *is a SAS CPA secure proactive scheme.*

The proof of this theorem is trivial and neglected here. There are two advantages resulting from the proactive property: firstly, large number of *GID*s could be adopted in the proactive system, while no more than m *GID*s is allowed to be used in the basic construction. Since the public keys remain unchanged through different periods, then the secret keys obtained from the old system could still be used for decrypting in the updated system, although the old secret keys could not be mixed with the newly-obtained secret keys to decrypt since they correspond to different polynomial evaluations. In other words, if a user tries to update his(her) secret keys, then he(she) has to reobtain the secret keys from all the authorities. The ciphertext generated in the old system are still *decipherable* and no modification needs to be done to them for the public parameters remain unchanged. Furthermore, there's another bonus effect due to the proactive property. Because P-MA-ABE-LU scheme is proven secure under the mobile adversary model and the mobile adversary is required to successfully attack the system during a shortened period (because the secret keys of the authorities are changed each period) rather than the whole lifetime as in the basic construction, then the difficulty for the adversary to attack the system increases. As a result, the security of the system is enhanced.

6.2. Some other extensions

Most extensions mentioned in Chase's scheme such as **Changing** d_k, **Leaving out authorities**, could be realized by applying the method used in Chase's scheme directly to our proposed schemes. Our proposed scheme could also be considered as a non trivial realization of an extension named **More complicated functions of the authorities** in Chase's paper since the central authority is removed in the proposed constructions. However, there seems no convenient way to realize another extension **Adding attribute authorities** in our proposed constructions. It seems that all the authorities, no matter old or newly-jointed authorities, need to get together to renew the whole system. This kind of trivial re-initialization implies certain inconvenience for the users as mentioned in Section 4.2. Thus, how to add new authorities to the proposed constructions without causing too much trouble for the users would be left as an open problem.

[9] The targeted attribute set is chosen at the beginning of the proactive scheme here.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 60972034, 60970110 and 60773086.

Appendix A. DKG protocol

The following DKG protocol outputs a public key g^x, and the players form a (t,n) secret sharing of the corresponding secret x.

Generating x:

Each player P_i performs a Pederson-VSS of a random value z_i as a dealer:

(A) P_i chooses two random polynomials $f_i(z),\ f_i'(z)$ over Z_q of degree t:

$$f_i(z) = a_{i0} + a_{i1}z + \cdots + a_{it}z^t$$
$$f_i'(z) = b_{i0} + b_{i1}z + \cdots + b_{it}z^t$$

Let $z_i = a_{i0} = f_i(0)$. P_i broadcasts $C_{ik} = g^{a_{ik}}h^{b_{ik}} \bmod p$ for $k = 0,\ldots,t$. P_i computes the shares $s_{ij} = f_i(j) \bmod q,\ s_{ij}' = f_i'(j) \bmod q$ for $j = 1,\ldots,n$ and sends $s_{ij},\ s_{ij}'$ to player P_j.

(B) Each player P_j verifies the shares he received from the other players. For each $i = 1,\ldots,n$, P_j checks if

$$g^{s_{ij}}h^{s_{ij}'} = \prod_{k=0}^{t}(C_{ik})^{j^k} \bmod p \tag{2}$$

If the check fails for an index i, P_j broadcasts a complaint against P_i.

(C) Each player P_i who, as a dealer, received a complaint from player P_j broadcasts the values $s_{ij},\ s_{ij}'$ that satisfy Eq. (2).

(D) Each player marks as *disqualified* any player either received more than t complaints in Step 1B, or answered to a complaint in step 1C with values that falsify Eq. (2).

Each player then builds the set of non-disqualified players *QUAL*. (All honest players build the same *QUAL* set and hence, for simplicity, a unique global name is used to denote this set).

The distributed secret value x is not explicitly computed by any party, but it is equal to $x = \sum_{i\in QUAL} z_i \bmod q$. Each player P_i sets his share of the secret as $x_i = \sum_{i\in QUAL} s_{ij} \bmod q$ and $x_i' = \sum_{i\in QUAL} s_{ij}' \bmod q$.

Extracting $y = g^x \bmod p$:

Each player $i \in QUAL$ exposes $y_i = g^{z_i} \bmod p$ via Feldman VSS:

Each player $P_i,\ i \in QUAL$, broadcasts $A_{ik} = g^{a_{ik}} \bmod p$ for $k = 0,\ldots,t$.

Each player P_j verifies the values broadcast by the other players in *QUAL*, namely, for each $i \in QUAL,\ P_j$ checks if

$$g^{s_{ij}} = \prod_{k=0}^{t}(A_{ik})^{j^k} \bmod p \tag{3}$$

If the check fails for an index i, P_j complaints against P_i by broadcasting the values $s_{ij},\ s_{ij}'$ that satisfy Eq. (2) but not satisfy Eq. (3).

For player P_i who receives at least one valid complaint, i.e. values that satisfy Eq. (2) and not Eq. (3), the other players run the reconstruction phase of Pederson-VSS to compute $z_i,\ f_i(z),\ A_{ik}$ for $k = 0,\ldots,t$ in the clear. For all players in *QUAL*, set $y_i = A_{i0} = g^{z_i} \bmod p$. Compute $y = \prod_{i\in QUAL} y_i \bmod p$.

Appendix B. The properties of PSS protocol

According to [11], the proactive secret sharing (PSS) protocol could be mainly divided into three algorithms as follows:

1. *PSS.Init*
   This algorithm is performed by all the authorities, executing the DKG protocol. Having a security parameter as an input, this algorithms outputs the description of the space over which the secret sharing will be performed, namely it creates a pair p, q of big primes s.t. $q|(p-1)$, and an element g of order q in Z_p^*. Let $\mathcal{S}$ be the space $\mathcal{S} = (Z_q)^n \times \left(Z_p^*\right)^{(n+1)}$. For every $x \in Z_q$, we will denote by $\mathcal{S}_x$ a probability distribution in $\mathcal{S}$ of elements $I = (\{x_1,\ldots,x_n\},\{y_1,\ldots,y_n\},y)$ s.t. $\{x_1,\ldots,x_n\}$ form a random $(\lfloor\frac{n}{2}\rfloor,n)$-threshold polynomial secret sharing of a number x s.t. $y = g^x$, and $y_i = g^{x_i}$ for every $i \in \{1,\ldots,n\}$. This algorithm picks out a secret x uniformly from Z_q, and gives each player $P_i, i \in \{1,\ldots,n\}$ a random share $x_i \in Z_q$ as its private output, and $g^{x_1},\ldots,g^{x_n},g^x$ as a public output. I_0 are used to denote the private and public output $I_0 = (Pri_0, Pub_0) = (\{x_1,\ldots,x_n\},\{y_1 = g^{x_1},\ldots,y_n = g^{x_n}, y = g^x\})$.
2. *PSS.Reconstruct*
   For every $I \in \mathcal{S}_x$, **Reconstruct**$(I) = x$, even if up to t servers are compromised by the adversary.

3. *PSS.Update*
   This algorithm is performed by all the authorities, executing the protocol JZSS to renew the share held by each authority at the beginning of each time period. On input $I_{i-1} = (Pri_{i-1}, Pub_{i-1})$, even in the presence of an adversary who compromises up to t servers in both the $i-1$st and the ith time periods, protocol **Update** outputs I_i, which is picked anew at random from the probability distribution $\mathcal{S}_x$. Note that, **Reconstruct**(Pri_i, Pub_i) outputs x for every time period i even in the presence a mobile adversary corrupting up to t severs in each time period. This ensures the robustness of the PSS protocol.

The semantic security [11] is defined in the following theorem:

Theorem 3. *We call a DKG-PSS protocol (JZSS-PSS protocol)* **DKG-PSS.Init** *and* **DKG-PSS.Update (JZSS-PSS.Init** *and* **JZSS-PSS.Update** *for JZSS-PSS protocol) secure relative to any auxiliary information if For every prior knowledge κ, no computationally bounded machine on input κ and the complete view of a $(t-1,n)$-threshold mobile adversary who corrupts up to $t-1$ servers every time period, can compute anything more (with non negligible probability) than a polynomial time machine which takes as inputs κ, the public information $p,\ q,\ g,\ g^x$ $(p,q,g$, the secret 0 for JZSS-PSS protocol), and the secret shares of up to $t-1$ servers in each time period.*

Note that the security of DKG-PSS protocol could be proven under the (t,n) threshold mobile adversary, and thus it is secure under the $(t-1,n)$ threshold mobile adversary. The proof of this theorem related to JZSS-PSS protocol is similar to that of DKG-PSS protocol and neglected here.

Appendix C. KeyGeneration(τ, MK)

The algorithm outputs a key that enables the user to decrypt a message encrypted under a set of attributes γ if and only if $\tau(\gamma) = 1$. The algorithm proceeds as follows. First choose a polynomial q_x for each node x (including the leaves) in the tree τ. These polynomials are chosen in the following way in a top-down manner, starting from the root node r. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Now, for the root node r, set $q_r(0) = y$ and d_r other points of the polynomial q_r randomly to define it completely. For any other node x, set $q_x(0) = q_{\mathrm{parent}(x)}(\mathrm{index}(x))$ and choose d_x other points randomly to completely define q_x. Once the polynomials have been decided, for each leaf node x, we give the following secret value to the user: $D_x = g^{q_x(0)/t_i}$ where $i = \mathrm{att}(x)$: The set of above secret values is the decryption key D.

Appendix D. DecryptNode(E, D)

It is a recursive algorithm that takes as input the ciphertext $E = \left(\gamma, E', \{E_i\}_{i\in\gamma}\right)$, the private key D (we assume the access tree τ is embedded in the private key), and a node x in the tree. It outputs a group element of G_2 or $\perp$.

Let $i = \mathrm{att}(x)$. If the node x is a leaf node then:

$$DecryptNode(E,D,x) = \begin{cases} e(D_x, E_i) \\ = e\left(g^{q_x(0)/t_i}, g^{s\cdot t_i}\right) \\ = e(g,g)^{s\cdot q_x(0)} & \text{if } i \in \gamma \\ \perp & \text{otherwise} \end{cases} \tag{4}$$

Consider the recursive case when x is a non-leaf node. The algorithm DecryptNode(E,D,x) then proceeds as follows: for all nodes z that are children of x, it calls DecryptNode(E,D,z) and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, we compute:

$$\begin{aligned} F_x &= \prod_{z\in S_x} F_z^{\triangle_{i,S'_x}(0)}, \quad \text{where } i = \mathrm{index}(z), \quad S'_x = \{\mathrm{index}(z) : z \in S_x\} = \prod_{z\in S_x}\left(e(g,g)^{s\cdot q_z(0)}\right)^{\triangle_{i,S'_x}(0)} \\ &= \prod_{z\in S_x}\left(e(g,g)^{s\cdot q_{\mathrm{parent}(z)}(\mathrm{index}(z))}\right)^{\triangle_{i,S'_x}(0)} \text{ (by construction)} = \prod_{z\in S_x} e(g,g)^{s\cdot q_x(i)\cdot\triangle_{i,S'_x}(0)} \text{(using polynomial interpolation)} \end{aligned} \tag{5}$$

Appendix E. Security proof of Theorem 1

Proof. We will show that if there exists a $(t-1,n)$-threshold adversary that could break the security property of our proposed scheme (as defined in Definition 2), then we could use this adversary to break the DBDH assumption or to violate the secrecy of the underlying DKG and JZSS protocol. Without lost of generality, the adversary is assumed to control $t-1$ authorities.

Since we assume the adversary might control $t-1$ authorities, then it already has $t-1$ shares corresponding to the secret $e(g_1,g_2)^s$ here. We could hide our hard problem in the share the adversary is about to get from the rest honest authorities. The adversary would not have enough secret keys to generate the corresponding share from these authorities, which means the authorities would give it insufficient secret keys such that $\left|\mathcal{A}^k_{C_*} \bigcap \mathcal{A}^k_u\right| < d_k$.

Our reduction follows that in [3,6], and behaves as follows:

Given a DBDH tuple $A = g^a,\ B = g^b,\ C = g^c$ and $Z = e(g,g)^{abc}$ or $e(g,g)^R$ for a random number $R \in Z_q$.

Receive the targeted attribute set $\mathcal{A}_{C_*}$ and a list of corrupted authorities $\mathcal{B}$ from adversary. Without lost of generality, the corrupted authorities controlled by the adversary are assumed to be the first $t-1$ authorities $\mathcal{B} = \{AA_1, \ldots, AA_{t-1}\}$. The honest authorities controlled by the simulator are the last $n-t+1$ authorities $\mathcal{G} = \{AA_t, \ldots, AA_n\}$. □

Setup: A, B(implicitly set $a = a_0, b = b_0, A = g_1, B = g_2$) are given as part of public parameters PK.

The simulator simulates the DKG protocol twice and JZSS protocol m times independently, the respective inputs of these protocols are A and B (see [7] for the simulation of DKG protocol and JZSS protocol). The adversary has $a_{k,0}, a_{k,1}, \ldots, a_{k,m}, b_{k,m+1}, k \in \{1, \ldots, t-1\}$ for $\{AA_1, \ldots, AA_{t-1}\}$ from the simulation of these two protocols. According to the above simulation, the simulator knows all the secret keys $\{a_{k,l}, k \in \mathcal{B}, l = 0, 1, \ldots, m\}$. That means, the simulator is able to compute all the $z_{k,u} = p_k(0) = a_{k,0} + a_{k,1}GID + \cdots + a_{k,m}GID^m,\ k = 1, 2, \ldots, t-1$.

The adversary should also be given the following information:

The PK of the authority in $\mathcal{G}$: Choose randomly $\beta_{k,j}$ from Z_q; $PK : \left\{T_{k,j} = g^{\beta_{kj}}\right\}_{j \in \mathcal{U} \bigcap \mathcal{A}^k_{C_*}},\ \left\{T_{kj} = B^{\beta_{kj}}\right\}_{j \in \mathcal{A}^k - \mathcal{A}^k_{C_*}}$ ($\mathcal{A}^k$ denotes the attribute set mastered by AA_k).

The rest secret keys of the authority in $\mathcal{B}$: Choose randomly $t_{k,j}$ from Z_q, $SK : \{t_{kj}\}$.

Secret Key Queries: Secret key queries for user u to most honest Attribute Authorities $AA_{\widehat{K}} \in \{AA_t, \ldots, AA_n\}$ should satisfy the requirement $\left|\mathcal{A}^{\widehat{K}}_{C_*} \bigcap \mathcal{A}^{\widehat{K}}_u\right| < d_{\widehat{K}}$. The simulation still allows the adversary to obtain enough secret keys from one honest authority $AA_{\overline{K}}$ since the adversary has to obtain at least $t+1$ qualified sets of secret keys from these authorities in order to decrypt in reality, and the simulation behaves as follows.

For the authority $AA_{\overline{K}}$, choose a random $z_{\overline{k},u} \in Z_q$, and set $p_{\overline{K}}(0) = z_{\overline{k},u}$. Choose a random polynomial such that $\rho(0) = z_{\overline{k},u}$, and we implicitly set $p(i) = \rho(i)$. For $i \in \mathcal{A}^{\overline{K}}_{C_*},\ t_{\overline{K},i} = \beta_{\overline{K},i}$, so $D_{\overline{K},i} = B^{\rho(i)/\beta_{\overline{K},i}} = B^{p(i)/t_{\overline{K},i}}$. For any other attributes $i \notin \mathcal{A}^{\overline{K}}_{C_*}$ we have $t_{\widehat{K},i} = b\beta_{\overline{K},i}$, so $D_{\overline{K},i} = g^{\rho(i)/\beta_{\overline{K},i}} = B^{\rho(i)/b\beta_{\overline{K},i}} = B^{p(i)/t_{\overline{K},i}}$.

For the authority $AA_{\widehat{K}}$, set $p_{\widehat{K}}(0) = \frac{a - \sum_{k \in \{1,2,\ldots,t-1,\overline{K}\}} (z_{k,u}\gamma_k)}{\gamma_{\widehat{K}}}$, $\gamma_{\widehat{K}}$ is the Lagrange coefficient corresponding to the set $\{1, 2, \ldots, t-1, \overline{K}, \widehat{K}\}$. For $i \in \mathcal{A}^{\widehat{K}}_{C_*}$, choose $d_{\widehat{K}} - 1$ random points v_i, set $p_{\widehat{K}}(i) = v_i$. For these attributes, $t_{\widehat{K},i} = \beta_{\widehat{K},i}$, so $D_{\widehat{K},i} = B^{p_{\widehat{K}}(i)/\beta_{\widehat{K},i}}$. Thus, $p_{\widehat{K}}(x)$ can be interpolated by the above mentioned $d_{\widehat{K}}$ points.

For any other attributes $i \notin \mathcal{A}^{\widehat{K}}_{C_*}$, we have implicitly defined $p_{\widehat{K}}(i) = \Delta_0(i)(p_{\widehat{K}}(0)) + \sum \Delta_j(i)v_j,\ \Delta_j(i)$ denotes the Lagrange coefficient of $p_{\widehat{K}}(j)$ in the computation of $p_{\widehat{K}}(i)$. For these attributes $t_{\widehat{K},i} = b\beta_{\widehat{K},i},\ i \notin \mathcal{A}^{\widehat{K}}_{C_*}$, then

$$D_{\widehat{K},i} = B^{p_{\widehat{K}}(i)/t_{\widehat{K},i}} = g^{p_{\widehat{K}}(i)/\beta_{\widehat{K},i}} = g^{\frac{\Delta_0(i)\frac{a - \sum_{k \in \{1,2,\ldots,t-1,\widehat{K},K\}} (z_{k,u}\gamma_k)}{\gamma_{\widehat{K}}} + \sum \Delta_j(i)v_j}{\beta_{\widehat{K},i}}} = g^{\frac{\Delta_0(i)a - \sum_{k \in \{1,2,\ldots,t-1,\widehat{K},K\}} (\Delta_0(i)z_{k,u}\gamma_k) + \sum \Delta_j(i)v_j\gamma_{\widehat{K}}}{\beta_{\widehat{K},i}\gamma_{\widehat{K}}}}$$

$$= A^{\frac{\Delta_0(i)}{\beta_{\widehat{K},i}\gamma_{\widehat{K}}}} \cdot g^{\frac{-\sum_{k \in \{1,2,\ldots,t-1,\widehat{K},K\}} (\Delta_0(i)z_{k,u}\gamma_k) + \sum \Delta_j(i)v_j\gamma_{\widehat{K}}}{\beta_{\widehat{K},i}\gamma_{\widehat{K}}}}$$

Challenge: Receive $M_0,\ M_1$ and pick a random $x \in \{0, 1\}$. Challenge Ciphertext: $Z \cdot M_x,\ \{C^\beta_{k,i}\}_{i \in \mathcal{A}^{k*}_C}$.

Guess: Receive a guess x' and if $x = x'$ guess $e(g,g)^{abc}$ otherwise guess $e(g,g)^R$.

Since the adversary should have non negligible advantage ϵ to guess correctly M_x when $Z = e(g,g)^{abc}$, and no better than $\frac{1}{2}$ probability to guess correctly when $z = e(g,g)^R$. Thus, if the adversary guess correctly, we guess that $Z = e(g,g)^{abc}$, otherwise we guess that $Z = e(g,g)^R$. In consequence, an adversary breaking the MA-FIBE scheme implies an algorithm breaking the DBDH assumption with non negligible advantage $\frac{\epsilon}{2}$ or violating the secrecy of the underlining DKG or JZSS protocol. It can be concluded that the proposed scheme is CPA secure under the selective set model.

Appendix F. P-MA-ABE-LU.Update algorithm

Tables 1 and 2.

Table 1
Authorities secret keys.

	DKG	JZSS	⋯	JZSS	DKG
Sk_1	$a_{1,0}$	$a_{1,1}$	⋯	$a_{1,m}$	$b_{1,m+1}$
Sk_2	$a_{2,0}$	$a_{2,1}$	⋯	$a_{2,m}$	$b_{2,m+1}$
⋯	⋯	⋯	⋯	⋯	⋯
Sk_n	$a_{n,0}$	$a_{n,1}$	⋯	$a_{n,m}$	$b_{n,m+1}$
Public keys	$g_1 = g^{a_0}$				$g_2 = g^{b_0}$

Table 2
P-MA-ABE-LU.**Update** algorithm.

Input:
The public keys PK: $g,\ g_1 = g^{a_0},\ g_2 = g^{b_0},\ \{t_{k,1}, t_{k,2}, \ldots, t_{k,w+1}\}_{k=1}^{n}$
The secret key $SK = \{SK_k\}_{k=1,\ldots,n}$
where $SK_k = (a_{k,0}, a_{k,1}, a_{k,1}, \ldots, a_{k,m}, b_{k,m+1})$
Output:
The public keys PK' : $g,\ g_1 = g^{a_0},\ g_2 = g^{b_0},\ \{t_{k,1}, t_{k,2}, \ldots, t_{k,w+1}\}_{k=1}^{n}$
The secret key $SK' = \{SK'_k\}_{k=1,\ldots,n}$
where $SK'_k = \left(a'_{k,0}, a'_{k,1}, a'_{k,1}, \ldots, a'_{k,m}, b'_{k,m+1}\right)$

P-MA-ABE-LU.**Update**(PK, SK):
FOR $i = 0$ to $m + 1$
Run JZSS protocol and output $\sigma_{k,i}, k \in \{1, \ldots, n\}$ for authority $\mathcal{A}_k, k \in \{1, \ldots, n\}$.
FOR $k = 1$ to n
if $0 \leqslant i \leqslant m$ then $a\prime_{k,i} \leftarrow a_{k,i} + \sigma_{k,i}$, and safely erase $\sigma_{k,i}$ and $a_{k,i}$;
if $i = m + 1$ then $b'_{k,i} \leftarrow b_{k,i} + \sigma_{k,i}$, and safely erase $\sigma_{k,i}$ and $b_{k,i}$;
END FOR.
END FOR.

References

[1] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE Symposium on Security and Privacy, 2007, pp. 321–334.
[2] F. Cao, Z. Cao, A secure identity-based proxy multi-signature scheme, Inform. Sci. 179 (3) (2009) 292–302.
[3] M. Chase, Multi-authority attribute based encryption, in: TCC, 2007, pp. 515–534.
[4] L. Cheung, C. Newport, Provably secure ciphertext policy abe, in: ACM Conference on Computer and Communications Security, 2007, pp. 456–465.
[5] Y. Frankel, P. Gemmell, P.D. MacKenzie, M. Yung, Optimal resilience proactive public-key cryptosystems, in: FOCS, 1997, pp. 384–393.
[6] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Robust threshold dss signatures, in: EUROCRYPT, 1996, pp. 354–371.
[7] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Robust threshold dss signatures, Inform. Comput. 164 (1) (2001) 54–84.
[8] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure distributed key generation for discrete-log based cryptosystems, J. Cryptol. 20 (1) (2007) 51–83.
[9] V. Goyal, A.J. 0002, O. Pandey, A. Sahai, Bounded ciphertext policy attribute based encryption, in: ICALP, vol. 2, 2008, pp. 579–591.
[10] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: ACM Conference on Computer and Communications Security, 2006, pp. 89–98.
[11] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, Proactive public key and signature systems, in: ACM Conference on Computer and Communications Security, 1997, pp. 100–110.
[12] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, Proactive secret sharing or: how to cope with perpetual leakage, in: CRYPTO, 1995, pp. 339–352.
[13] J.O. Kwon, I.R. Jeong, D.H. Lee, A forward-secure e-mail protocol without certificated public keys, Inform. Sci. 179 (24) (2009) 4227–4231.
[14] M. Cerecedo, T. Matsumoto, H. Imai, Efficient and secure multiparty generation of digital signatures based on discrete logarithms, in: IEICE Transactions on Fundamentals, 1993, pp. 532–545.
[15] J. Nam, Y. Lee, S. Kim, D. Won, Security weakness in a three-party pairing-based protocol for password authenticated key exchange, Inform. Sci. 177 (6) (2007) 1364–1375.
[16] R. Ostrovsky, A. Sahai, B. Waters, Attribute-based encryption with non-monotonic access structures, in: ACM Conference on Computer and Communications Security, 2007, pp. 195–203.
[17] M. Pirretti, P. Traynor, P. McDaniel, B. Waters. Secure attribute-based systems, in: ACM Conference on Computer and Communications Security, 2006, pp. 99–112.
[18] T. Rabin, A simplified approach to threshold and proactive rsa, in: CRYPTO, 1998, pp. 89–104.
[19] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: EUROCRYPT, 2005, pp. 457–473.
[20] A. Shamir, Identity-based cryptosystems and signature schemes, in: CRYPTO, 1984, pp. 47–53.
[21] S. Wang, Z. Cao, K.-K.R. Choo, L. Wang, An improved identity-based key agreement protocol and its security proof, Inform. Sci. 179 (3) (2009) 307–318.
[22] J. Zhang, J. Mao, A novel id-based designated verifier signature scheme, Inform. Sci. 178 (3) (2008) 766–773.