Self-Contained Data Protection Scheme Based on CP-ABE

Bo Lang, Runhua Xu, and Yawei Duan

State Key Laboratory of Software Development Environment,
School of Computer Science and Engineering, Beihang University,
100191 Beijing, China
`langbo@buaa.edu.cn`,
`{xurunhua,duanyawei}@nlsde.buaa.edu.cn`

Abstract. Self-protection capabilities of outsourced data become noteworthily important in cloud computing. Ciphertext-Policy Attribute Based Encryption (CP-ABE) can dynamically control the user group of the encrypted data by defining decryption attributes; hence has certain ability of access control. Although there are different schemes of CP-ABE, as far as we know, most of these schemes can only express simple policies with *AND*, *OR* and *threshold* attribute operations, which cannot support traditional access control policies. In order to effectively integrate access control with encryption to build a self-contained data protection mechanism, this paper proposed an Extended CP-ABE(ECP-ABE) scheme based on the existing CP-ABE scheme. The ECP-ABE scheme can express any Attribute Based Access Control (ABAC) policies represented by arithmetic comparison and logical expressions that involve *NOT*, $<, \leq, >, \geq$, [ ], ( ), ( ] and [ ) operators in addition to *AND*, *OR* and *threshold* operators. We prove the Chosen-plaintext Attack (CPA) security of our scheme under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model, and also discuss the experimental results of the efficiency of ECP-ABE.

Keywords: self-contained data protection, ciphertext-policy attribute based encryption (CP-ABE), extended CP-ABE, attribute based access control, cloud computing

1 Introduction

In open computing environment such as cloud computing, the protection mechanism of outsourced data (sometimes just simply called data) attracts much more attentions [1, 2]. These data departs from the control domain of its owner and is stored and managed by unreliable service providers. Hence, the self-protection capabilities of data become very important. Traditionally, access control and encryption are the two basic protection mechanisms for achieving data integrity and confidentiality. Self-contained protection of data means that data itself can ensure its integrity and confidentiality without depending on other parties.

Data encryption is the primary data self-protection means at present. Traditional Public-Key encryption and Identity Based Encryption schemes [3] are designed for one-to-one communication, which means the information encrypted by a public key or identity can only be decrypted by the specific private key. This situation has been changed since Sahai and Waters proposed the Attribute Based Encryption scheme [4], where ciphertexts are not necessarily encrypted to one particular user. Both users private keys and ciphertexts are associated with a set of attributes or a policy over attributes. When the attributes of a users private key can match the attributes of the ciphertext in a certain extent, the user can be able to decrypt the ciphertext. By defining decryption attributes, ABE can dynamically control the user group of the encrypted data.

Goyal et al. further developed this idea and introduced two variants of ABE, namely key-policy attribute based encryption(KP-ABE) and ciphertext-policy attribute based encryption(CP-ABE). In KP-ABE, whose first construction is given by [5], ciphertext is associated with a set of attributes and the secret key is associated with the access tree. A user will be able to decrypt if and only if the attributes in the ciphertext satisfy his access tree. In CP-ABE, the idea is reversed. The ciphertext is associated with the access tree and the secret key is associated with a set of attributes, and the encrypting party determines the decryption policy.

Bethencourt et al[6] gave the initial structure of CP-ABE. We refer to this scheme as BSW07 in this paper. BSW07 is relatively expressive and efficient, but the security argument is based on generic group model, an artificial model which assumes the attacker needs to access an oracle in order to perform any group operation. After that, many researchers have presented different schemes for the less ideal security argument, trying to prove the security based on a well-studied complexity-theoretic problem. And also there are many people worked at improving the efficiency or the flexibility of access policy for the CP-ABE scheme. These schemes mainly support three kinds of access policy structures: AND-gates, tree structure and Linear Secret Share Scheme (LSSS) matrix. Among them, the tree structure and LSSS matrix are relatively flexible, which supports *AND, OR* and *threshold* operation. BSW07 uses bag of bits to express policies containing $<, \leq, >, \geq$. However, this approach is much complex and has poor scalability, and is hard to be used in practical applications. For NOT operator, BSW07 has no solution. To the best of our knowledge, there is no efficient way to express an access policy that contains operators such as $NOT, <, \leq, >$ and $\geq$ in present CP-ABE schemes, which makes CP-ABE only support simple attribute policies.

Access control and encryption are the two key techniques in data-centric protection, and CP-ABE makes it possible to integrate these two techniques seamlessly. However, the limited access policy expression in CP-ABE restricts its access control capability.

Our contribution. In the area of access control, Attribute-based Access Control (ABAC) model [7–9] makes access control decisions based on user attributes. The policies in ABAC are defined as attribute expressions that contain attributes,

constants, and $AND, OR, NOT, <, \leq, >, \geq, [\ ], (\ ), (\ ]$ and $[\ )$ operators, and can express complex access control rules. If the access policy structure of CP-ABE can be enhanced to express complex attribute policies as ABAC, CP-ABE will become an ideal scheme for implementing data self-protection in open computing environments. Following this idea, we proposed the Extended CP-ABE scheme (ECP-ABE). In ECP-ABE, by introducing extended leaf nodes, the access tree of CP-ABE is enhanced to support all kinds of logical and arithmetic comparison operators, including $<, \leq, >, \geq, NOT, [\ ], (\ ), (\ ]$ and $[\ )$. Therefore, ECP-ABE can realize powerful access control as well as encryption, and data processed by ECP-ABE will have strong self-protection capabilities. Our scheme is proven to be chosen plaintext attack(CPA) secure under the decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model.
Organization. The remaining sections are organized as follows. In Section 2, we introduce related work. In Section 3, we review the preliminaries. We present our extended CP-ABE (ECP-ABE) scheme in Section 4, and give an implementation framework of ECP-ABE in Section 5. We then discuss the performance of ECP-ABE from aspects of security and efficiency in Section 6. Finally, we conclude this paper in Section 7.

2 Related Work

BSW07 expresses the access policy by a tree structure which supports *AND, OR* and *threshold* operations. At the same time, the length of the ciphertext and the encryption or decryption time are linearly related with the number of attributes of the access structure tree. However, the security proof of BSW07 is based on generic group model, rather than the standard numerical theoretical assumptions. In addition, as a result of using polynomial interpolation to resume secret during the decryption phase, BSW07 needs greater number of bilinear mapping and exponentiation operation, and costs of these operations are relatively high.

After that many scholars have proposed different schemes [10]. Cheung and Newport first gave the CP-ABE scheme (CN07)[11] under CPA security based on DBDH assumption. However, the scheme only have the AND and NOT operator in the access policy structure, and the ability of policy expression is poor. Moreover, the length of the ciphertext and the key, and the time of encryption or decryption are linearly related with the number of attributes, which lead to the lower efficiency. Goyal et al raised the Bounded Ciphertext Policy Attribute Based Encryption scheme[12] based on DBDH assumption, which supported the *AND,OR* and *threshold* operations.

Nishide gave an Attribute-Based encryption scheme [13] with partially hidden encryptor-specified access structures, which only supported the AND operation and attributes have more than one candidate value. Emura et al first raised the CP-ABE with constant ciphertext length based on Nishide's scheme [14], which improved the efficiency of the algorithm. But it also just supported the AND operation. Ibraimi et al gave an efficient and provable secure CP-ABE scheme [15] based on DBDH assumption using the threshold secret share technology

[16], which supported *AND, OR* and *threshold* operations. Its access structure was an n-tree and the costs of key generation, encryption and decryption are lower than the BSW07 scheme. Waters has used the LSSS matrix to express the access control policy and pointed out that the ability of expression is not lower than the tree structure[17].

In order to support complex Boolean access policies, Junod and Karlov[18] proposed an efficient public-key ABBE scheme allowing arbitrary access policies, which is based on a modification of the Boneh-Gentry-Waters broadcast encryption scheme. Chen et al[19] presented two new CP-ABE schemes, which have both constant-size and constant computation costs for a non-monotone AND gate policy. Jin et al[20] enhanced the attribute-based encryption with attribute hierarchy and obtain a provable secure HABE under tree hierarchy. Attrapadung et al[21, 22] proposed the first KP-ABE schemes allowing for non-monotonic access structures and with constant ciphertext size. Zhiguo et al[23] proposed a hierarchical attribute-set-based encryption (HASBE) scheme which extended the ciphertext-policy attribute-set-based encryption for access control in cloud computing.

From the view of security and expressive ability of access policy, only the W08 and ITHJ09 scheme supported the AND, OR and threshold operation under the theoretical assumptions of the standard numerical. And the computation cost of encryption and decryption of ITHJ09 is lower than W08's. Therefore, we choose ITHJ09 as the basic CP-ABE scheme, and further expand the access policy tree of ITHJ09 to construct an Extended CP-ABE scheme.

3 Preliminaries

3.1 Access Tree

Definition 1. ***(Access Tree[6]).*** *Let τ be a tree representing a kind of* ***Access Structure****[24]. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x < num_x$. When $k_x = 1$, the threshold gate is an* OR *gate and when $k_x = num_x$, it is an* AND *gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.*

We define tree functions over the tree. The function **parent**(x) represents the parent of node x. If x is a leaf node, we define the function **attr**(x) to denote the attribute with the leaf node. As the access tree has an ordering between the children of every node, the function **index**(x) represents the index number of each child node.

Definition 2. ***(Satisfied Access Tree[6]).*** *Let τ be an access tree with root r. Denote by τ_x the subtree of τ rooted at the node x. Thus, τ is the same as τ_r. If a set of attributes γ satisfies the access tree τ_x, we denote it as $\tau_x(\gamma) = 1$. We compute $\tau_x(\gamma)$ recursively as follows. If x is a non-leaf node, evaluate $\tau_{x'}(\gamma)$ for all children x' of node x. $\tau_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\tau_x(\gamma)$ returns 1 if and only if* ***att(x)***$\in \gamma$.

3.2 CP-ABE Algorithms

The ciphertext-policy attribute based encryption scheme consists of four fundamental algorithms [6]: Setup, Encrypt, Key Generation, and Decrypt.

- **Setup** (k). The setup algorithm takes no input other than the security parameter k. It outputs the public parameters PK and a master key MK.
- **Key-Generation** (MK, S). The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK.
- **Encrypt** (PK, M, A). The encryption algorithm takes as input the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt M and produce a ciphertext C_T such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains A.
- **Decrypt** (PK, C_T, SK). The decryption algorithm takes as input the public parameters PK, a ciphertext C_T which contains an access policy A, and a private key SK. If the set S of attributes satisfies the access structure A then the algorithm will decrypt the ciphertext and return a message M, otherwise return the error symbol $\perp$.

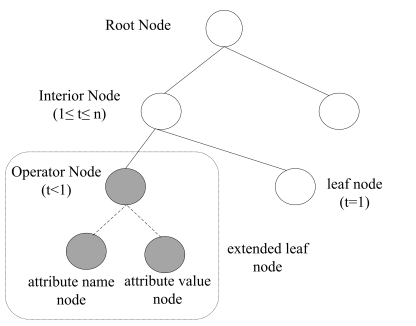
4 ECP-ABE Scheme

The ITHJ09 used Shamir secret sharing technique to support *AND, OR* and *of* (threshold) nodes based on CP-ABE scheme. The access policy tree is n-ary tree. Each node has two attributes: the number of child nodes n and threshold value $t(1 \leq t \leq n)$. When $t = 1$, it's an *OR* gate; when $t = n$, it's an *AND* gate; when $1 < t < n$, it's an *of* gate. The leaf node associates policy properties and its value t is 1. The ECP-ABE scheme we proposed is based on the ITHJ09 scheme and we extend the access tree to make it be able to express the complex policies that contain arithmetic and logical expressions.

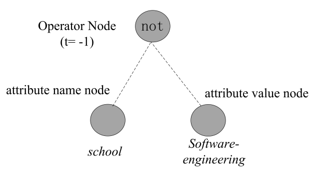
4.1 Extended Leaf Node

The universal attribute set U is published by the Trusted Authority. Each user has his or her attribute set w which is used for key generation and we refer to it as the basic attribute set. In Attribute Based Access Control system, user's access right could be dynamically calculated according to his security character and the resource he applied for. Inspired by this, we extend the leaf node of the access policy tree.

We replace the original leaf node with the ***operator node*** and give it two children, which we refer to as the ***attribute name node*** and the ***attribute value node***, as shown in Fig.1(a). The ***operator node***, the ***attribute name node*** and the ***attribute value node*** compose an ***extended leaf node***, and the attribute expression described by an extended leaf node is called an extended



(a) Extended access tree



(b) Example of extended leaf node

Fig. 1: Extend access tree.

attribute, for instance, the attribute “age>18” is an extended attribute. Meanwhile, the range of threshold value t of the extended leaf node is less than 0 from the original value 1.

The operator node only has the threshold value $t(t < 0)$. Different value of t denotes specific operator, for instance, $t = -1$ for NOT operator, $t = -2$ for $>$ operator. The attribute name/value node denotes the attribute name and the attribute value respectively that are associated with the operator. With this structure, we can express policy attributes using operators of $NOT, <, \leq, >, \geq$ $, [\ ], (\ ), (\ ]$ and $[\ )$. Fig.1(b) is an example of this structure, which express the policy attribute school not software-engineering.

ECP-ABE scheme augments two kinds of operators: comparison operators and logic operators.

- Comparison operators: $<, \leq, >, \geq$.
- Interval operators: $[\ ], (\ ), (\ ], [\ )$.
- Logic operators: *not*.

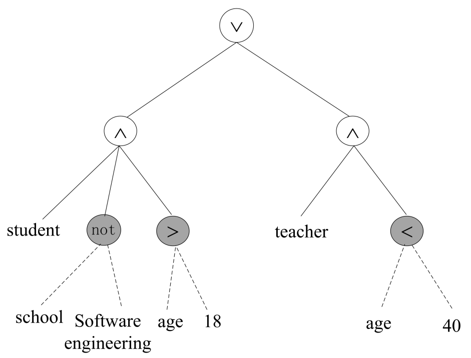
The values of t and the corresponding operator that each value represents are defined in Tab.1.

Table 1: Values of t and its corresponding operator.

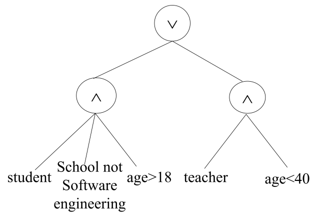
Value t	-1	-2	-3	-4	-5	-6	-7	-8	-9
Operator	*not*	$<$	$>$	$\leq$	$\geq$	[ ]	( )	( ]	[ )

4.2 Transforming an Extended Policy Tree to a Standard Tree

Now we define the extended policy tree as the *extended tree* T^* and the original tree is called the *standard tree* T. An extended tree can be transformed to an



(a) An extended policy tree T^*



(b) The standard policy tree T corresponding to T^* in Fig.2a

Fig. 2: Examples of extended and standard access policy tree.

equivalent standard tree by removing the attribute name/value node, converting the operator node to the standard leaf node and then assigning the attribute expression described by the extended leaf node as an extended attribute to the standard leaf node. The extended tree T^* and the standard tree T express the same access policy. For example, the extended tree in Fig.2a can be transferred into the standard tree in Fig.2b.

The user expresses the access policy using the extended tree and makes it the parameter for the encryption. The encryption algorithm will firstly transform the extended tree to a standard tree, and then encrypts the message using the standard access policy tree. Finally, we attach the extended tree in the ciphertext.

To decrypt the ciphertext, the decryption party needs to apply the secret key by giving PKG his basic attribute set and the extended parts of the access tree. At the PKG side, we use the **attribute verification algorithm** as shown in Alg.1 to verify and transform an extended leaf node. This algorithm will first get user's basic attribute set and then traverse the attribute set to check whether or not the attribute N satisfies the expression *exp(N.O.V)*. If the answer is yes, it returns the string form of *exp(N.O.V)*, i.e. "*attribute name operator attribute value*" which is regarded as an *extended attribute* of the user. Otherwise it will return null.

Here is an example of the transformation.

There is a file F in a campus network system and the file has an access policy: "It can be accessed if and only if the user is a teacher under age of 40 or an older than 18-year-old student who is not in school of software-engineering". So, we can give the policy "$T^* = (student \wedge schoolnotsoftware - engineering \wedge age > 18) \vee (teacher \wedge age < 40)$", and the extended access tree for this policy is shown in Fig.2(a). Fig.2(b) is the standard access tree which converts from the extended tree in Fig.2(a). The encryption party encrypts the file F with T and attaches T^* in the ciphertext.

ALGORITHM 1: Attribute Verification

```
1  Get the expression exp(N.O.V) of the extended leaf node, where N,O and
   V denote the basic attribute name, the operator array and the attribute
   value array respectively;
2  Traverse the basic attribute set A' to find the basic attribute N and its
   value V';
3  Let $O_{size}$ be the size of the array O, $V_{size}$ be the size of the array V;
4  if $O_{size}$ == 1 && $V_{size}$ == 1
5      Let N=V', calculate the boolean expression N.O[1].V[1];
6      if the value of the expression is true
7          Convert exp(N.O.V) to string S=N.O[1].V[1];
8          return S;
9      else
10         return null;
11     end if
12 else if $O_{size}$ == 2 && $V_{size}$ == 2
13     Let N=V', calculate the boolean expression V[1].O[1].N.O[2].V[2];
14     if the value of the expression is true
15         Convert exp(N.O.V) to string S=V[1].O[1].N.O[2].V[2];
16         return S;
17     else
18         return null;
19     end if
20 else
21     return null;
22 end if
```

Suppose user A and user B wants to decrypt the file F. The basic attributes of A is {*student, school=computer science, age=20*}, and the basic attributes of B is {*student, school=computer science, age=17*}. Firstly, Both A and B need to extract the extended parts of T^* from the ciphertext and send them with their basic attribute set to PKG. Then, PKG verifies and generates the new attribute set {*student, school not software-engineering, age<18*} for A, and the new attribute set {*student,school not software-engineering, age=17*} for B. The corresponding private keys are generated using these new attribute sets by PGK concurrently. Obviously, the attribute set of user B doesn't satisfy the access policy, hence user A can decrypt the file F while user B can't.

4.3 Encryption and Decryption Process of ECP-ABE

The encryption party expresses the access policy with an extended tree and the tree in the ciphertext is also in the extended structure. However, when encrypts a message, the encryption algorithm will first transform the extend tree to an equivalent standard tree and encrypt the message using the standard one. So in the encryption phase, we can use the algorithm of ITHJ09 scheme. For ciphertexts that encrypted under different extended access trees, users have to apply for different secret keys, since PKG need to verify and generate extended attributes according to the extended tree and user's basic attributes. Detailed encryption and decryption processes are described as follows.

a. **Initialize**: the system initializes and generates public parameter *pk* and master key *mk*. It gives *pk* to the encryption party. The description of initialization algorithm *Setup (k)* is as follow.
   i. Generate a bilinear group G of prime order p with a generator g and a bilinear map $e : G \times G \to G_T$.
   ii. Generate the attribute set $U = \{a_1, a_2, \ldots, a_m\}$, for some integer m, and random elements $\alpha, t_1, t_2, \ldots, t_m \in Z_p^*$. Let $y = e(g, g)^\alpha, T_j = g^{t_j}(1 \leq j \leq m)$. The public key is $pk = \{e, g, y, T_j(1 \leq j \leq m)\}$, and the master key is $mk = (\alpha, t_j(1 \leq j \leq m))$.
b. **Specify the access policy**: the encryption party specifies access policy, which is expressed by an extended tree T^*.
c. **Encryption**: the encryption party calls the encryption algorithm *Encrypt (m, T^*, pk)* with plaintext m, the extended tree T^* and the public parameter *pk*. The encryption algorithm will first transform T^* to the equivalent standard tree T, and then encrypt m under T using Shamir's secret sharing technique. Finally it returns the ciphertext C_T which contains T^*, such that only users who have the secret key generated from the attributes that satisfy T^* will be able to decrypt the message. The detail description is as follows:
   i. Convert the T^* to the standard tree T;
   ii. Select a random element $s \in Z_p^*$ and compute $c_0 = g^s$ and $c_1 = M \cdot y^s = M \cdot e(g, g)^{\alpha s}$;
   iii. Set the value of the root node of T to be s, mark all child nodes as un-assigned, and mark the root node assigned. Recursively, for each un-assigned non-leaf node, do the following:
   If its child nodes are un-assignedthe secret s is divided using *(t,n)*-Shamir secret sharing technique. The relation of n and t is: if the symbol is *of* then $1 \leq t \leq n$; if the symbol is *AND*, then $t = n$; if the symbol is *OR*, then $t = 1$. To each child node a share secret $s_i = f(i)$ is assigned. Mark this node assigned.The function $f(x)$ is the random polynomial over Z_p:$f(x) = \sum_{j=0}^{t-1} a_j x^j$.
   iv. For each leaf attribute $a_{j,i} \in T$, compute $c_{j,i} = T_j^{s_i}$, where i denote the index of the attribute in the access tree.
   v. Return the ciphertext: $C_T = (T, c_0, c_1, \forall a_{j,i} \in T : c_{j,i})$.
d. **Secret key request**: when a user gets C_T and wants to decrypt, he first needs to analyze the structure of T^* and find the extended parts, then apply for the secret key by giving PKG his basic attribute set w and the extended parts of the access tree.
e. **Secret key generation**: PKG first verify the user's basic attribute. If the basic attributes of the user are authenticated, PKG will extract the attribute name, the attribute value and the operator, and run Alg.1. Attributes in w that satisfy the extended leaf node will be replaced by the returned extended attributes. Finally PKG gets the new attribute set w^* and generates the

secret key sk_{w^*} corresponds to w^* and sends it back to the user. The detailed description is as follows:

i. Select a random value $r \in Z_p^*, d_0 = g^{\alpha - r}$.

ii. For each attribute a_j in w, compute $d_j = g^{rt_j^{-1}}$.

iii. Return the secret key $sk_w = (d_0, \forall a_j \in w : d_j)$.

f. **Decryption**: the user calls the decryption algorithm $Decrypt(C_T, sk_{w^*})$. The algorithm returns message m if the smallest attribute set $w \in w^*$ that corresponds to sk_{w^*} satisfies T. Otherwise it returns an error symbol $\perp$. More details are as follows:
For every attribute $a_j \in w'$, computing:

$$m = \frac{c_1}{e(c_0, d_0) \cdot \prod_{a_j \in w'} e(c_{j,i}, d_j)^{l_i(0)}} \tag{1}$$

$l_i(0)$ is a Lagrange coefficient and can be computed by everyone who knows the index of the attribute in the access tree.

Proof. Correctness.

$$\begin{aligned} m' &= \frac{c_1}{e(c_0, d_0) \cdot \prod_{a_j \in w'} e(c_{j,i}, d_j)^{l_i(0)}} \\ &= \frac{m \cdot e(g,g)^{\alpha s}}{e(g^s, g^{\alpha - r}) \cdot e(T_j^{s_i}, g^{rt_i^{-1}})^{l_i(0)}} \\ &= \frac{m \cdot e(g,g)^{\alpha s}}{e(g^s, g^{\alpha - r}) \cdot \prod_{a_j \in w'} e(g^{t_j s_i}, g^{rt_i^{-1}})^{l_i(0)}} \\ &= \frac{m \cdot e(g,g)^{\alpha s}}{e(g^s, g^{\alpha - r}) \cdot e(g,g)^{rs}} \\ &= \frac{m \cdot e(g,g)^{\alpha s}}{e(g^s, g^{\alpha})} = m \end{aligned} \tag{2}$$

5 ECP-ABE Implementation Framework

The ECP-ABE scheme can be used to protect data which is owned by enterprises and collaborative groups and is stored in a cloud. Building a practical and efficient ECP-ABE implementation mechanism is meaningful for these applications. The typical CP-ABE scheme has three basic components, which are PKG Server, Encryption party and Decryption party. In our proposed ECP-ABE implementation framework, we introduce the Attribute Authority (AA) to relieve the burdens of PKG and avoid the efficiency bottlenecks. Our ECP-ABE implementation framework is shown in Fig.3, and the framework has four main components:

- PKG, which generates ABE private keys for users based on the attributes submitted.
- AA, which authenticates user's identity and verify the user's attribute set.

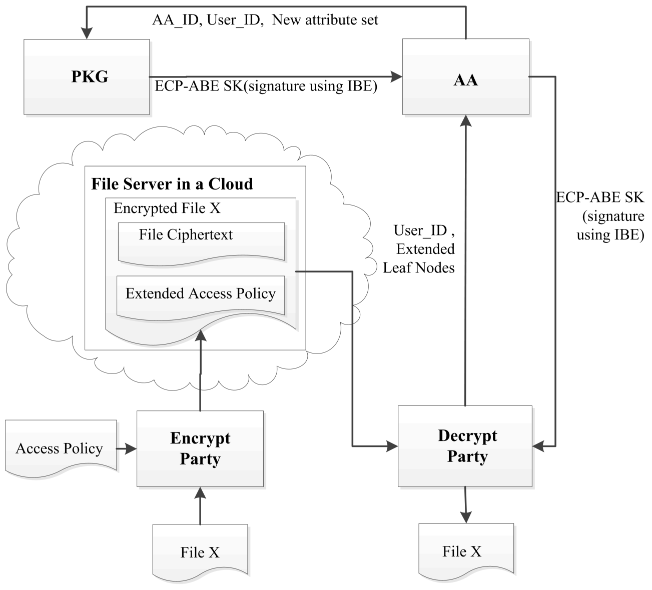
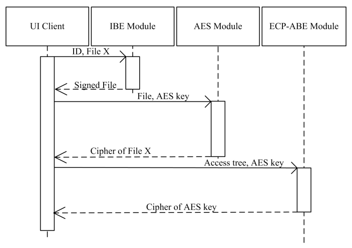


Fig. 3: The implementation framework of ECP-ABE.

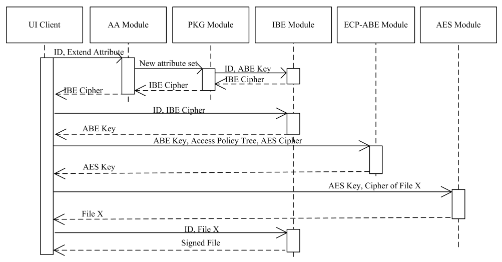
- Encrypt party, which encrypts a file.
- Decrypt party, which decrypts a ciphertext to get the original file.

In order to realize the integral data protection, the framework synthetically uses ECP-ABE, IBE and AES scheme. Among the three schemes, the AES symmetric encryption algorithm is used to encrypt the data file, and the key of AES is encrypted by ECP-ABE, while IBE provides the signature verification for the communications in the process of ECP-ABE private key distribution.

The file encryption process is achieved in the Encrypt Party of the framework, as shown in Fig.4a. Firstly, the user signs the file by IBE module to ensure data



(a) Encryption sequence diagram of ECP-ABE implementation mechanism.



(b) Decryption sequence diagram of ECP-ABE implementation mechanism.

Fig. 4: Encryption/Decryption process in ECP-ABE Implementation framework.

integrity. And then, it uses the AES symmetric encryption algorithm to encrypt the data file. The AES key is encrypted by ECP-ABE module with the extended access policy tree. At last, the Encrypt party puts the signature file, the ECP-ABE ciphertext, the data file ciphertext and the access policy file together and forms a final ciphertext.

The file decryption process is achieved in the Decrypt Party of the framework, as shown in Fig.4b. Firstly, the decrypt party parses the ciphertext and gets every part of the ciphertext. Then, it extracts the extended leaf nodes from the access policy tree and sends them to the AA with the ID of the user. The AA authenticates the user and verifies the extended leaf nodes using user ID and sends the new attribute set to PKG. PKG generates the ECP-ABE private key and sign it using IBE and returns it back. The decrypt party decrypts and verifies the IBE ciphertext, gets the ECP-ABE private key. Then, the AES key can be decrypted from the ECP-ABE ciphertext and the original data file is decrypted. At last, the decrypt party signed the data file using IBE and then compare signature with the original signature to verify if the file has been tampered.

6 ECP-ABE Performance Analysis

6.1 Security

The major contribution of ECP-ABE scheme is the extension of the access tree. The core encryption/decryption algorithm of ECP-ABE is based on ITHJ09 scheme. In ITHJ09 scheme semantic security under chosen-plaintext attack (C-PA) is modeled by IND-sAtt-CPA game. The security model of ECP-ABE will still be based on IND-sAtt-CPA game, but the challenging access tree provided by the adversary in **Init** phase will be an extended tree instead of a standard tree. IND-sAtt-CPA game of ECP-ABE security model is as follows:

- **Init.** The adversary A chooses the challenge access tree T^* and gives it to the challenger, T^* is an extended tree.
- **Setup.** The challenger runs *Setup* to generate(*pk,mk*) and gives the public key *pk* to adversary A. The challenger also transforms T^* to the equivalent standard tree T.
- **Phase1.** Adversary A makes a secret key request to the *Keygen* oracle for any attribute set $w = \{a_j | a_j \in U\}$, with the restriction $a_j \in T^*$ and a_j does not satisfy the policy attribute requirement expressed by the extend part of T^*. The challenger runs Alg.1 to generate extended attribute set w^* and then returns $Keygen(w^*, mk)$.
- **Challenge.** Adversary A sends to the challenger two equal length messages m_0, m_1. The challenger picks a random bit $b \in 0, 1$ and returns $c_b = Encrypt(m_b, T^*, pk)$.
- **Phase2.** Adversary A can continue querying *Keygen* oracle with the same restriction as in **Phase1**.
- **Guess.** Adversary A outputs a guess $b' \in 0, 1$.

The advantage of A winning this game is defined as:

$$\epsilon = |Pr[b' = b] - 1/2|. \tag{3}$$

Definition 3. *We say that the(t, ϵ)-DBDH assumption[4, 25] holds if no t-time algorithm has advantage at least ϵ in solving the DBDH problem in G_0.*

Definition 4. *ECP-ABE scheme is said to be secure against an adaptive chosen-plaintext attack(CPA) in the standard model if any polynomial-time adversary has only a negligible advantage in the above IND-sAtt-CPA game.*

In the above game, adversary A uses an extended tree to challenge instead of a standard tree. We have the following analyse:

- The limitation for the basic attribute set $w = \{a_j | a_j \in U\}$ provided by adversary A in **Phase1** is $a_j \notin T^*$ and a_j does not satisfy the policy attribute requirement expressed by the extended part of T^*. According to this limitation, we can infer that $\forall b_j^* \in w^*, b_j^* \notin T$. So in **Phase1**, changes of access tree will not introduce any new security problem, i.e. the secret key that A gets could not decrypt the ciphertext c_b.
- Although adversary A submits the extended tree T^* in **Init** phase, message m_b is encrypted under standard tree T. Transformation between T^* and T is public. In **Phase1**, Challenge and **Phase2**, adversary A could design the query and challenge against T^*. So the attacking ability of A keeps the same.

Hence, we can conclude that in ECP-ABE scheme the advantage of A in the IND-sAtt-CPA game equals to the advantage of A in ITHJ09 scheme, i.e. in ECP-ABE scheme any polynomial-time adversary has only a negligible advantage in the IND-sAtt-CPA game.

So ECP-ABE scheme is secure against an adaptive CPA in the standard model. Our extension for the access tree will not lower the system security compared with ITHJ09.

6.2 Efficiency

In ITHJ09 scheme, encryption requires $|T| + 1$ exponentiations in G and one exponentiation in G_T and $|T|$ is the number of attributes in the access tree T. Key generation requires $|w| + 1$ exponentiations in G, w is the attribute set the user has. Decryption requires $|w'| + 1$ pairing operations, $|w'|$ multiplications, w' is the set of attributes satisfying the access tree, $w' \in w$.

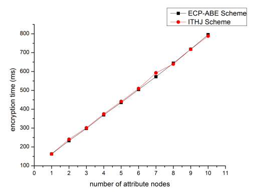
ECP-ABE uses the encryption and decryption algorithms of ITHJ09 scheme, so the calculation expenses and the length of ciphertext are the same as ITHJ09. The ECP-ABE scheme has two main differences compared with ITHJ09 scheme: the ECP-ABE has the conversion from an extended tree to a standard tree during the encryption; it also has the verification and transformation of extended attributes during the key generation. Meanwhile, in ECP-ABE, the attribute set used to generate the private key will be expanded after the extended leaf node transformation. Therefore, the added calculation expense comes from the following two factors:

- The transformation from the extended tree to the standard tree during the encryption phase.
- The verification and transformation of the extended attributes during the key generation phase.

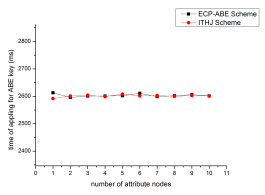
The following experiments illustrate the impact of the above factors on the actual efficiency.

We use two groups of policy files as the test samples. One group only contains policies with the standard attributes which are used as the policies of the ITHJ09 scheme, and the other only contains policies with the extended attributes which are used as the policies of our ECP-ABE scheme. Each group has 10 test policy files to test the efficiency and the number of attribute node varies from 1 to 10. The access tree is a two-tier structure when there are 1-4 attribute nodes, a three-tier structure when there are 5-7 attribute nodes, and a four-tier structure when there are 8-10 attribute nodes.

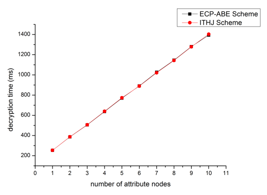
We run three times for each test policy file and get the average cost as the result. Fig.5 is the result of the tests.



(a) The encryption time cost comparison of ECP-ABE scheme and ITHJ09 scheme



(b) The key application time cost comparison of ECP-ABE scheme and ITHJ09 scheme



(c) The decryption time cost comparison of ECP-ABE scheme and ITHJ09 scheme

Fig. 5: The efficiency comparison of ECP-ABE and ITHJ09 scheme.

Discussion: the verification of the extended leaf nodes and the transformation from the extended tree to the standard tree nearly have no effect on the performance during the encryption and key application phase. However, the ECP-ABE scheme has greatly enhanced the access policy expression capability.

7 Conclusion

The paper proposed an ECP-ABE scheme, which introduces the extended leaf nodes into the access policy tree to support access policy formulas involving operators including $NOT, <, \leq, >, \geq, [\ ], (\ ), (\ ]$ and $[\ )$ in addition to *AND, OR* and *threshold* operators. Hence the scheme enhanced access control ability of CP-ABE prominently, and achieved self-contained protection for outsourced data in open computing environments.

ECP-ABE adopts the same implementation mechanism as other CP-ABE schemes. Basing on the experiments analysis, we can see that our scheme has nearly the same expense compared with ITHJ09 scheme, and ECP-ABE scheme is proven chosen plaintext attack(CPA) secure under the decisional Bilinear Diffie-Hellman assumption in the standard model. Hence, ECP-ABE can keep the security and efficiency properties of the CP-ABE scheme which it based on, but prominently improves the access capability of the baseline scheme. Also, the policy extension method used in ECP-ABE is not limited to the ITHJ09 scheme; it can be used on other CP-ABE schemes that utilize tree-based access policy structures.

For future work, it would be interesting to probe other more efficient way to enhance the access control capability of CP-ABE schemes, such as working on other access policy structures like LSSS.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (Grant No.61170088) and Foundation of the State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2013ZX-05).

References

1. Samarati, P., di Vimercati, S.D.C.: Data protection in outsourcing scenarios: Issues and directions. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ACM (2010) 1–14
2. Vimercati, S.D.C.D., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Encryption policies for regulating access to outsourced data. ACM Transactions on Database Systems (TODS) **35**(2) (2010) 12
3. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Advances in cryptology, Springer (1985) 47–53
4. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Advances in Cryptology–EUROCRYPT 2005. Springer (2005) 457–473
5. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security, ACM (2006) 89–98
6. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Security and Privacy, 2007. SP'07. IEEE Symposium on, IEEE (2007) 321–334
7. Hur, J., Noh, D.K.: Attribute-based access control with efficient revocation in data outsourcing systems. Parallel and Distributed Systems, IEEE Transactions on **22**(7) (2011) 1214–1221
8. Lang, B., Foster, I., Siebenlist, F., Ananthakrishnan, R., Freeman, T.: A flexible attribute based access control method for grid computing. Journal of Grid Computing **7**(2) (2009) 169–180
9. Wang, G., Liu, Q., Wu, J.: Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In: Proceedings of the 17th ACM conference on Computer and communications security, ACM (2010) 735–737

10. Su, J.S., Cao, D., Wang, X.F., Sun, Y.P., Hu, Q.L.: Attribute based encryption schemes. Journal of Software **22**(6) (2011) 1299–1315
11. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: Proceedings of the 14th ACM conference on Computer and communications security, ACM (2007) 456–465
12. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Automata, Languages and Programming. Springer (2008) 579–591
13. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Applied cryptography and network security, Springer (2008) 111–129
14. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Information Security Practice and Experience. Springer (2009) 13–23
15. Ibraimi, L., Tang, Q., Hartel, P., Jonker, W.: Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In: Information Security Practice and Experience. Springer (2009) 1–12
16. Shamir, A.: How to share a secret. Communications of the ACM **22**(11) (1979) 612–613
17. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Public Key Cryptography–PKC 2011. Springer (2011) 53–70
18. Junod, P., Karlov, A.: An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In: Proceedings of the tenth annual ACM workshop on Digital rights management, ACM (2010) 13–24
19. Chen, C., Zhang, Z., Feng, D.: Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In: Provable Security. Springer (2011) 84–101
20. Li, J., Wang, Q., Wang, C., Ren, K.: Enhancing attribute-based encryption with attribute hierarchy. Mobile Networks and Applications **16**(5) (2011) 553–561
21. Attrapadung, N., Libert, B., De Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Public Key Cryptography–PKC 2011. Springer (2011) 90–108
22. Attrapadung, N., Herranz, J., Laguillaumie, F., Libert, B., De Panafieu, E., Ràfols, C.: Attribute-based encryption schemes with constant-size ciphertexts. Theoretical Computer Science **422** (2012) 15–38
23. Wan, Z., Liu, J., Deng, R.H.: Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing. Information Forensics and Security, IEEE Transactions on **7**(2) (2012) 743–754
24. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
25. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Advances in Cryptology-EUROCRYPT 2004, Springer (2004) 223–238