Provably Secure and Efficient Bounded Ciphertext Policy Attribute Based Encryption

Xiaohui Liang
Department of Computer Science & Engineering
Shanghai Jiao Tong University
liangxh1207@gmail.com

Zhenfu Cao
Department of Computer Science & Engineering
Shanghai Jiao Tong University
zfcao@cs.sjtu.edu.cn

Huang Lin
Department of Computer Science & Engineering
Shanghai Jiao Tong University
faustlin@sjtu.edu.cn

Dongsheng Xing
Department of Computer Science & Engineering
Shanghai Jiao Tong University
dsdsdds@sjtu.edu.cn

ABSTRACT

Ciphertext policy attribute based encryption (CPABE) allows a sender to distribute messages based on an access policy which can be expressed as a boolean function consisting of (OR, AND) gates between attributes. A receiver whose secret key is associated with those attributes could only decrypt a ciphertext successfully if and only if his attributes satisfy the ciphertext's access policy. Fine-grained access control, a new concept mentioned by GPSW in CCS'06 can realize a more delicate access policy which could be represented as an access tree with threshold gates connecting attributes.

In ICALP'08, Goyal *et al.* design a bounded CPABE (denoted as GJPS) with fine-grained access policy which can be proven secure under a number-theoretic assumption. In this paper, we improve their scheme by providing faster encryption/decryption algorithm and shortened ciphertext size. Moreover, we use one-time signature technique to obtain a chosen ciphertext secure extension and give its complete security proof in the standard model under traditional Decisional Bilinear Diffie-Hellman (DBDH) assumption and strong existential unforgeability of one-time signature scheme.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Theory

ASIACCS '09, March 10-12, 2009, Sydney, NSW, Australia.

Keywords

Access Control, Attribute Based Encryption, Public Key Cryptography

1. INTRODUCTION

To distribute a message to a specific set of users, a trivial method is to encrypt it under each user's public key or identity in traditional cryptosystem [2, 3, 4, 7, 8, 9]. As expected, ciphertext size and computational cost of encryption/decryption algorithms are linearly with the number of receivers. Therefore, it is less attractive or even intolerable when the number of receivers is large. Indeed, in most cases, the qualified receivers share some common attributes, such as working location, gender or age range. Due to this reason, recently, ciphertext policy attribute based encryption (CPABE) scheme was introduced as an efficient method to solve the above problem.

In a CPABE scheme, each user is identified by an attribute set γ and receives the secret keys corresponding to those attributes from the authority. The sender who aims to distribute a message will construct an access policy by connecting the attributes with OR, AND, threshold gates. For example, "(AGE>25) AND (CS)" represents the restrictions that a qualified users should at least obtain the secret keys representing age more than 25 and the ownership of a computer science degree; three attributes (DB,OS,DM) connecting by a "two out of three threshold gate" restricts the decryption only to be successful when a user at least registers two courses from the list (Database, Operating System, Discrete Mathematics). As in Figure 1, a sender would save the ciphertext encrypted under the access policy in the server S_1. The users U_2 and U_3 have the access right to the message since they both have two attributes from the attribute set, while U_1 can not since he(she) only has one attribute. This is an efficient and convenient approach for a user to broadcast his message to the others in practice. Therefore, we mainly focus on developing a more efficient CPABE scheme.

Considering the security proof of ABE scheme, more flexible access policy adopted on the sender's side makes the simulation more difficult. Thus, restricting the size of access policy is necessary for designing a CPABE which reduces the

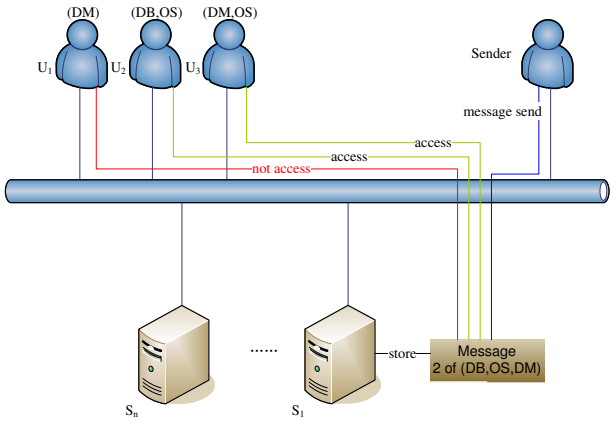

Figure 1: A sample for CPABE system

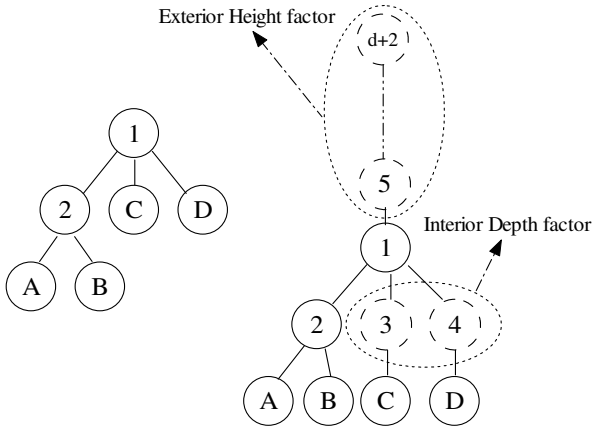

Figure 2: A conversion from (d, c)-bounded access tree $\mathcal{T}$ to its (d, c)-normal form $\mathcal{T}_n$

security to a number theoretic assumption. Goyal *et al.* introduced a new bounded CPABE (BCPABE), in which the encryption access tree must be limited in two conditions: the maximum height and maximum cardinality of each non-leaf node. For example, the access tree (to the left of Figure 2) with height 2 has the maximum cardinality 3.

In GJPS, a bounded ciphertext policy attribute based encryption scheme secure in the standard model was proposed to support a bounded size access tree with threshold gates as its non-leaf nodes. At the beginning, the system manager pre-sets two bounds (d, c) and a unique (d, c)-universal access tree $\mathcal{T}_u$ (Figure 3). After that, the system manager will publish the public parameter and generate users' secret keys according to this universal access tree. If a sender wants to distribute his message, it requires him to first convert a (d, c)-bounded access tree $\mathcal{T}$ (to the left in Figure 2) into its (d, c)-normal form[1] $\mathcal{T}_n$ (to the right in Figure 2) and then to complete encryption procedure according to a map constructed from a (d, c)-normal form access tree $\mathcal{T}_n$ to the (d, c)-universal access tree $\mathcal{T}_u$.

In the (d, c)-universal access tree $\mathcal{T}_u$, only leaf nodes of depth d is associated with real attributes. This setting leads to the fact that users have to construct a map from $\mathcal{T}_n$ to $\mathcal{T}_u$ in order to ensure that the leaf nodes in $\mathcal{T}_n$ share same real attributes with their corresponding leaf nodes in $\mathcal{T}_u$. However, this conversion of normal form actually expands the original tree $\mathcal{T}$ with a great many non-leaf nodes which lead a boost on computational cost. We conclude that the expanded size is mainly due to two factors called exterior height factor and interior depth factor (refer to Figure 2). Exterior height factor correlates with the height h of a (d, c)-bounded access tree $\mathcal{T}$. $d - h$ nodes must be added in order to expand the height of its normal form to d. Goyal *et al.* provide a method to eliminate this factor by constructing multiple parallel schemes with different-sized universal access trees, though it is very inefficient. Interior depth factor is the relative depth between leaf nodes in $\mathcal{T}$. Non-leaf nodes must be added in order to make the leaf nodes all at the same depth (which is the deepest level). In order to eliminate both factors, we neglect the interim step, i.e., converting the access tree to norm form, and directly map the (d, c)-bounded access tree selected by the sender to the (d, c)-universal tree. In other words, the redundant steps which pull all the leaf nodes to the deepest level by adding non-leaf nodes are eliminated and thus the computation cost is reduced.

[1]the definition of normal form can be looked up in Section 2.4

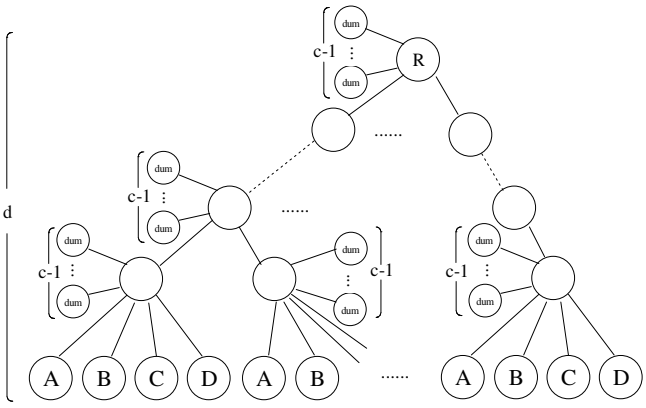

Figure 3: A (d, c)-universal access tree $\mathcal{T}_u$ in GJPS

As mentioned in GJPS, to construct a more efficient BCPABE scheme based on number-theoretic assumptions is an important open problem and we consider this paper as an affirmative answer to this problem.

This paper presents a bounded ciphertext policy attribute encryption scheme $\mathtt{BCP}_1$ which is more efficient than the previous works in GJPS. The security of $\mathtt{BCP}_1$ can be reduced to Decisional Bilinear Diffie-Hellman assumption in the standard model. Different from GJPS, the computational costs of encryption and decryption in our scheme are largely reduced since we eliminate redundant steps, meanwhile the ciphertext size is shorter. Nevertheless, as a tradeoff, we demonstrate that the spacial cost such as the size of public parameter and secret keys, increase but remain to be less than twice of the counterpart of GJPS. Furthermore,

we propose a provably secure BCPABE scheme $\mathtt{BCP}_2$ in the standard model under chosen ciphertext secure notion by adopting one-time signature technique. There are two approaches which could be used for extending $\mathtt{BCP}_1$ to $\mathtt{BCP}_2$. These two methods make a tradeoff between ciphertext size and the size of public/secret parameters.

To better understand our techniques, we illustrate a concrete example to explain the primitive idea we use to reduce the computational cost in encryption/decryption algorithms.

Consider a BCPABE scheme setup with bounds (d, c)[2] and an encryption under a (d, c)-bounded access tree $\mathcal{T}$ shown to the left of Figure 2. The threshold values of Node 1 and Node 2 are both set to be 2, and Nodes A, B, C, D represent four different real attributes. Now, we will show how the encryption algorithm of GJPS differs from that of ours.

In GJPS, to encrypt under $\mathcal{T}$, a user would first convert $\mathcal{T}$ to its normal form access tree $\mathcal{T}_n$ (shown to the right Figure 2). The threshold values of Node 3, Node 4, Node 5, $\cdots$, Node $d+2$ in $\mathcal{T}_n$ are all set to be 1. Assume the computational cost of one node is T, the total cost reaches $(4+c-2+c-2+d\times(c-1))\cdot T$ where exterior factor takes $(d-2)\cdot(c-1)\cdot T$ and interior factor takes $2\cdot(c-1)\cdot T$. Likewise, if a user has a secret key associated with attributes (C, D), he will at least expend $(1+2+d-2)\times c = (d+1)\times c$ paring computation. It is obvious that the larger the two initial parameters (d, c) are set to be, the more cost a user will spend on encryption and decryption.

In contrast, our scheme defines a map from $\mathcal{T}$ to $\mathcal{T}_u$ and the total cost on encryption takes $(4+c-2+c-2)\cdot T$. The user with attributes (C, D) will expend only c paring computation. Therefore, ours saves $d\times(c-1)\cdot T$ on encryption and at least dc times paring computation on decryption. The cost is even irrelevant with the initial parameter d. The comparison of general case can be found in Table 1 in Section 6.

Related Works. The concept of attribute based encryption (ABE) was introduced by Sahai and Waters [13]. In their scheme, the secret key is associated with an attribute set, and the ciphertext is also associated with another set of attributes. The decryption is only successful while these two sets overlap more than a preset threshold. Later, Goyal *et al.* [11] further separated ABE into two categories: ciphertext policy (CP) ABE and key policy (KP) ABE. While key policy ABE (the user's secret key represents an access policy of attributes) is well developed by the subsequent research [11, 12], how to design an efficient and secure ciphertext policy (the ciphertext represents an access policy chosen by the sender) ABE remains an open problem [1, 6, 10].

The first ciphertext policy attribute based encryption scheme was proposed by Bethencourt et al [1]. It allows users to encrypt a message under an expression consisting of threshold gates between attributes (called a fine-grained access structure). However, it only has a security argument in the generic group model and the random oracle model. After then, Cheung and Newport [6] gave a provably secure ciphertext policy ABE in the standard model. Their scheme supports an access policy with "AND" gate on positive and negative attributes but can not resist collusion attack while extending to threshold gate. Recently, a bounded ciphertext policy attribute based encryption scheme, supporting fine-grained access policy, was proposed in GJPS.

Waters[14] presented several CPABE schemes. The construction of the first scheme is very elegant, and the security can be reduced to decisional q-Bilinear-Diffie-Hellman-Exponent(BDHE) problem. The ciphertext size only linearly increases with the attributes presented in the access structure. Another scheme, based on DBDH assumption, is less efficient, due to the ciphertext size restricted by the length and width of matrix which is dependent on the size of the access structure.

Road map. The rest of this paper is organized as follows. In the next section, some preliminaries including bilinear map, complexity assumption, one-time signature technique and definitions of access tree are introduced. In Section 3, the definition of BCPABE and the security model for BCPABE are described. We propose our schemes in Section 4&5, and formally prove their security in the standard model, respectively. The comparisons between ours and others are summarized and given in Section 6. Finally, we conclude the paper with some interesting problems in Section 7.

2. PRELIMINARIES

2.1 Bilinear Pairing

We briefly review some notions about bilinear pairing [1, 4, 6, 13].

Consider two finite cyclic groups G and G_T having the same prime order p. It is clear that the respective group operation is efficiently computable. Assume that there exists an efficiently computable mapping $e : G \times G \to G_T$, called a bilinear map or pairing, with the following properties.

- Bilinear: For any $g, h \in G$, and $a, b \in Z_p$, we have $e(g^a, h^b) = e(g, h)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1$

Note that, $e(*, *)$ is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.2 Complexity Assumption

The security of identity based encryption and attribute based encryption schemes employing bilinear pairings is based on the assumption that some problems are hard to solve. In the following, we introduce Decisional Bilinear Diffie-Hellman (DBDH) problem and its corresponding assumption.

Decisional Bilinear Diffie-Hellman Problem. We say that an algorithm $\mathcal{S}$ is a ε'-solver of the DBDH problem if it distinguishes with probability at least $1/2+\varepsilon'$ between the two following probability distributions:

$\mathcal{D}_{bdh} = (g, g^a, g^b, g^c, e(g, g)^{abc})$, where a, b, c are chosen randomly in Z_p,

$\mathcal{D}_{rand} = (g, g^a, g^b, g^c, Z)$, where a, b, c are chosen randomly in Z_p and Z is chosen randomly in G_T.

Definition 1. *We say that the DBDH assumption holds in G and G_T if no any probabilistic polynomial-time ε'-solver of the DBDH problem for non-negligible value ε'.*

[2] $d \geq 2, c \geq 3$

2.3 One-Time Signature

A one-time signature scheme ots [5, 6] consists of three algorithms (ots.**KGen**, ots.**Sig**, ots.**Ver**). ots. **KGen**$(1^k) \rightarrow (sk, vk)$ is the key generation algorithm, which outputs a secret key sk and a public verification key vk. ots.**Sig**$(sk, m) \rightarrow \sigma$ is the sign algorithm which takes the secret key sk and a message m as its input, and outputs a signature σ. Finally, the verification algorithm ots.**Ver**$(\sigma, m, vk) \rightarrow 0$ or 1 takes the signature σ, a message m and a public verification key vk as its input, and outputs 1 if the signature is valid; 0 otherwise.

Concerning the security issue, an adversary first receives a public verification key vk generated from ots.**KGen**(1^k). He then makes at most one signature query for message m of his choice, and obtains as an answer the valid signature ots.**Sig**$(sk, m) \rightarrow \sigma$. Finally, he outputs a pair (m', σ'). We say that the adversary succeeds if $(m', \sigma') \neq (m, \sigma)$ and ots.**Ver**$(\sigma', m', vk) \rightarrow 1$.

A one-time signature scheme ots is ε_{ots}-secure if any polynomial-time adversary against ots has a success probability bounded by ε_{ots}.

2.4 Access Tree

In this section, several definitions and notions which are necessary for understanding the rest of this paper are given.

Attribute Set. n real attributes, indexed from 1 to n. Any attribute set $\gamma \subseteq \{1, \cdots, n\}$. $c-1$ dummy attributes, indexed from $n+1$ to $n+c-1$.

Access Tree. Let $\mathcal{T}$ represent an access tree with its root r. Each non-leaf node x can be seen as a threshold gate with threshold value k_x. If x has c_x child nodes, we certainly demand $0 < k_x \leq c_x$. If x is a leaf node of the access tree, it is associated with one single attribute, denoted as $\text{att}(x)$. We fix the root of an access tree to be at depth 0. Let $\Sigma_{\mathcal{T}}$ denote the set of all the non-leaf nodes, and $\Theta_{\mathcal{T}}$ denote all the leaf nodes. Let $\text{p}(x)$ denote the parent of node x. For each non-leaf node x, we define an order among x's child nodes, that is, every child node z is numbered from 1 to c_x. $\text{index}(z)$ returns such number associated with node z. For simplicity, if z is a leaf node, we let $\text{att}(z) = \text{index}(z)$.

Satisfying an Access Tree. Let $\mathcal{T}$ be an access tree with root r. $\mathcal{T}_x$ is a subtree rooted at a node x in $\mathcal{T}$. We say that, if an attribute set γ satisfies the access tree $\mathcal{T}_x$, we denote $\mathcal{T}_x(\gamma) = 1$. If x is a non-leaf node, evaluate $\mathcal{T}_z(\gamma)$ for all children z of node x. $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children of x return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma) = 1$ iff $\text{att}(x) \in \gamma$. If an access tree rooted at x is a γ-satisfied tree, then we call the node x as a γ-satisfied node. Suppose $\mathcal{T}$ is a γ-satisfied tree, we call a subtree of $\mathcal{T}$ with root r as a γ-satisfied non-redundant tree, if the cardinality of each non-leaf node x is equal to x's threshold value in $\mathcal{T}$ and every node is a satisfied node. Let $\hat{\mathcal{T}}$ denote the γ-satisfied non-redundant tree with minimum non-leaf nodes.[3]

Universal Access Tree. Here, we describe a universal access tree (Figure 4) with two input parameters d and c. First, we define a complete c-ary tree $\mathcal{T}'$ of height $d-1$, where each node has a threshold value c. Next, $c-1$ new leaf nodes named "dummy nodes" representing $c-1$ dummy attributes and n new leaf nodes named "real nodes" representing n real attributes are attached to each node in $\mathcal{T}'$. The resultant tree $\mathcal{T}_u$ is called a (d, c)-universal access tree. Let each node x except root has an index related with its parent node and $\text{att}(x) = \text{index}(x)$ if x is a leaf node of a universal access tree. Here, for all the child nodes of one non-leaf node x in $\mathcal{T}$, real nodes and dummy nodes will take indexes from $\{1, \cdots, n\}$ and $\{n+1, \cdots, n+c-1\}$ respectively, and other non-leaf nodes will be indexed from $\{n+c, \cdots, n+2c-1\}$.

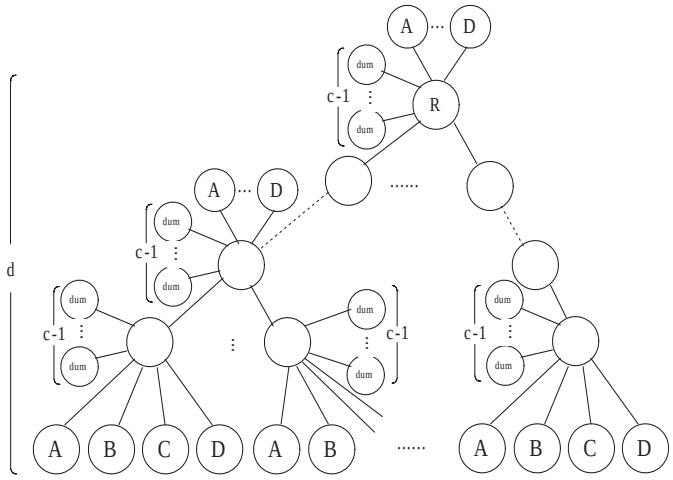

Figure 4: Modified Universal Tree

Bounded Access Tree. We called that $\mathcal{T}$ a (d, c)-bounded access tree if its height $d' \leq d$, each non-leaf node x in $\mathcal{T}$ has at most c non-leaf child nodes.

Normal Form. Consider a (d, c)-bounded access tree $\mathcal{T}_n$. We say that $\mathcal{T}_n$ exhibits the (d, c)-normal form if (a) its height $d' = d$, and (b) all the leaf nodes are at depth d. Any (d, c)-bounded access tree $\mathcal{T}$ can be converted to its normal form without modifying its satisfying logic. (This is a special technique used in GJPS.)

Map between Access Trees. The map, constructed from a (d, c)-bounded access tree $\mathcal{T}$ to (d, c)-universal access tree $\mathcal{T}_u$, is defined in the following way in a top-down manner. First, the root of $\mathcal{T}$ is mapped to the root of $\mathcal{T}_u$. Now, suppose that x' in $\mathcal{T}$ is mapped to x in $\mathcal{T}_u$. Let $z'_1, \cdots, z'_{c_{x'}}$ be the child nodes of x'. For $i \in \{1, \cdots, c_{x'}\}$, if z'_i is a leaf node $\text{att}(z'_i) \in \mathcal{N}$, set x's child node z such that $z = \text{map}(z'_i)$ where $\text{index}(z) = \text{index}(z'_i)$; if z'_i is a non-leaf node, set x's child node z such that $z = \text{map}(z'_i)$ where $\text{index}(z) = \text{index}(z'_i) + n + c - 1$. This procedure is performed recursively, until each node in $\mathcal{T}$ is mapped to a corresponding node in $\mathcal{T}_u$. Notice that in a (d, c)-bounded access tree, each non-leaf node x in $\mathcal{T}$ have at most c non-leaf child nodes, this recursive procedure can be terminable.

3. DEFINITIONS

In this section, we introduce the definitions and security models for bounded ciphertext policy attribute based encryption scheme.

3.1 BCPABE model

Definition 2. *A BCPABE scheme includes a tuple of probabilistic polynomial-time algorithms as follows.*

[3]The decryption cost depends on the non-leaf nodes in the γ-satisfied non-redundant tree the decryptor chooses.

- **Setup**$(d, c) \rightarrow$ (PP, MK): On input an implicit security parameter 1^k and two system parameters (d, c), the setup algorithm **Setup** outputs a public parameter PP and a master key MK.
- **KGen**$(\gamma, \text{MK}) \rightarrow (D)$: On input an attribute set γ and a master key MK, the key generation algorithm **KGen** outputs a secret key D.
- **Enc**$(\text{PP}, \mathcal{T}, m) \rightarrow (E)$: On input the public parameter PP, a (d, c)-bounded access tree $\mathcal{T}$ and a message m, the encryption algorithm **Enc** outputs a ciphertext E.
- **Dec**$(D, E) \rightarrow (m)$: On input a secret key D and a ciphertext E, if the attribute set in D satisfies the access tree in E, the decryption algorithm **Dec** decrypts the ciphertext E and returns a message m; otherwise, it outputs "$\perp$".

3.2 Security Model for BCPABE

Selective-Tree Model for BCPABE This model was first introduced by GJPS. The analogous selective-ID model lies in [2, 4, 5, 6]. We say that a BCPABE scheme is secure in the selective-tree CPA model if no probabilistic polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage in winning the following game.

Init $\mathcal{A}$ chooses an access tree $\mathcal{T}^*$ that he wishes to be challenged upon. The challenger runs **Setup** algorithm and gives $\mathcal{A}$ the resulting public parameter PP. It keeps the corresponding master key MK to itself.

Phase 1 $\mathcal{A}$ issues queries for secret keys related with many attribute sets γ_j, where γ_j does not satisfy the access tree $\mathcal{T}^*$ for all j.

Challenge Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length messages m_0, m_1 from the message space. The challenger chooses $\mu \in \{0, 1\}$ at random and encrypts m_μ with $\mathcal{T}^*$. Then, the ciphertext C^* is given to $\mathcal{A}$.

Phase 2 The same as Phase 1.

Guess $\mathcal{A}$ outputs a guess $\mu' \in \{0, 1\}$ and wins the game if $\mu' = \mu$.

We define $\mathcal{A}$'s advantage in this game as $|\Pr[\mu' = \mu] - \frac{1}{2}|$. The selective-tree CPA model can be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2, denoted as selective-tree CCA model.

4. OUR MAIN CONSTRUCTION

4.1 Basic BCPABE Scheme $\texttt{BCP}_1$

We now proceed the formal description of our first scheme $\texttt{BCP}_1$.

Setup(d, c) This algorithm takes two parameters (d, c) as its input. Define a real attribute set $\mathcal{U} = \{1, \cdots, n\}$ and a dummy attribute set $\mathcal{U}^* = \{n+1, \cdots, n+c-1\}$. Next, we define a (d, c)-universal access tree $\mathcal{T}_u$ as explained in Section 2.4. $(d, c, \mathcal{U}, \mathcal{U}^*, \mathcal{T}_u)$ are all used in the following algorithms.

Now, the algorithm generates the public parameter for this scheme. For each real attribute $j \in \mathcal{U}$, randomly choose a set of $|\Sigma_{\mathcal{T}_u}|$ numbers $\{t_{j,x}\}_{x \in \Sigma_{\mathcal{T}_u}}$ from Z_p. Further, for each dummy attribute $j \in \mathcal{U}^*$, randomly choose a set of $|\Sigma_{\mathcal{T}_u}|$ numbers $\{t^*_{j,x}\}_{x \in \Sigma_{\mathcal{T}_u}}$ from Z_p. Finally, randomly choose $y \in Z_p$. The public parameter $\text{PP} = \langle Y = e(g, g)^y, \{T_{j,x} = g^{t_{j,x}}\}_{j \in \mathcal{U}, x \in \Sigma_{\mathcal{T}_u}}, \{T^*_{j,x} = g^{t^*_{j,x}}\}_{j \in \mathcal{U}^*, x \in \Sigma_{\mathcal{T}_u}} \rangle$. The master key $\text{MK} = \langle y, \{t_{j,x}\}_{j \in \mathcal{U}, x \in \Sigma_{\mathcal{T}_u}}, \{t^*_{j,x}\}_{j \in \mathcal{U}^*, x \in \Sigma_{\mathcal{T}_u}} \rangle$.

KGen(γ, MK) This algorithm takes an attribute set $\gamma \subseteq \mathcal{U}$ and the master key MK as its input, then it outputs a secret key D which can be used for decrypting a ciphertext encrypted under a (d, c)-bounded access tree $\mathcal{T}$ iff $\mathcal{T}(\gamma) = 1$.

Now, the algorithm generates the secret key. For each user, choose a random polynomial q_x of degree $c-1$ for each non-leaf node x in the (d, c)-universal access tree. These polynomials are chosen in a top-down manner, satisfying $q_x(0) = q_{\text{p}(x)}(\text{index}(x))$ and $q_r(0) = y$. Once the polynomials have been fixed, it outputs the following secret key $D = \langle \gamma, \{D_{j,x} = g^{\frac{q_x(j)}{t_{j,x}}}\}_{j \in \gamma, x \in \Sigma_{\mathcal{T}_u}}, \{D^*_{j,x} = g^{\frac{q_x(j)}{t^*_{j,x}}}\}_{j \in \mathcal{U}^*, x \in \Sigma_{\mathcal{T}_u}} \rangle$.

Enc$(m, \text{PP}, \mathcal{T})$ This algorithm takes a message m, the public parameter PP and a (d, c)-bounded access tree $\mathcal{T}$ as its input.

Now, to encrypt the message m with the access tree $\mathcal{T}$, the algorithm first sets a map from $\mathcal{T}$ to $\mathcal{T}_u$ using the method mentioned in Section 2.4. Then, for each non-leaf node $x \in \mathcal{T}$, it chooses an arbitrary $(c - k_x)$-sized set ω_x[4] of dummy child nodes of x' in $\mathcal{T}_u$, $x' = \text{map}(x)$. After that, let $f(j, x) = 1$ if the node x in $\mathcal{T}$ has a child node associated with real attribute j; $f(j, x) = 0$ otherwise. Then, it chooses a random value $s \in Z_p$ and outputs the ciphertext $E = \langle \mathcal{T}, E' = m \cdot Y^s, \{E_{j,x} = T^s_{j,\text{map}(x)}\}_{j \in \mathcal{U}, x \in \Sigma_{\mathcal{T}}, f(j,x)=1}, \{E^*_{j,x} = T^{*s}_{j,\text{map}(x)}\}_{j \in \omega_x, x \in \Sigma_{\mathcal{T}}} \rangle$.

Dec(E, D) This algorithm takes a ciphertext E and a secret key D as its input. If the attribute set γ associated with D satisfies the access tree $\mathcal{T}$ in E, the algorithm continues doing follows; otherwise, output $\perp$.

A recursive algorithm $\text{DecryptNode}(E, D, x)$ takes the ciphertext E, the secret key D and a satisfied non-leaf node x in $\mathcal{T}$ as its input and outputs a group element of G_T or $\perp$. For each x's child node z,

- z is a real node, let $j = \text{att}(z)$. Then, we have:
  $F_{x,j} = \text{DecryptNode}(E, D, x)$
  $= \begin{cases} e(D_{j,\text{map}(x)}, E_{j,x}) = e(g, g)^{sq_{\text{map}(x)}(j)}, & \text{if } j \in \gamma; \\ \perp, & \text{otherwise.} \end{cases}$
- z is a dummy node. $j = \text{att}(z) \in w_x$. Then we have:
  $F_{x,j} = \text{DecryptNode}(E, D, x)$
  $= e(D^*_{j,\text{map}(x)}, E^*_{j,x}) = e(g, g)^{sq_{\text{map}(x)}(j)}$,

From the above procedure, for each non-leaf node x in $\mathcal{T}$, if $\mathcal{T}_x(\gamma) = 1$, we have at least $k_x + c - k_x = c$ different points $F_{x,j}$ to compute $e(g, g)^{sq_{\text{map}(x)}(0)}$ using Lagrange interpolation. By recursively executing such procedure in a down-to-top manner, and finally, it obtains $E'' = e(g, g)^{sq_r(0)} = e(g, g)^{sy}$, where r is the root of $\mathcal{T}$. The decryption algorithm outputs $m = E'/E''$.

4.2 Security Proof of $\texttt{BCP}_1$

THEOREM 1. *If the DBDH assumption holds in (G, G_T), then scheme $\texttt{BCP}_1$ is selective-tree CPA secure in the standard model.*

PROOF. Suppose there exists a polynomial-time adversary $\mathcal{A}$ who can attack $\texttt{BCP}_1$ in the selective-tree CPA model with non-negligible advantage ε. We construct a simulator $\mathcal{S}$ who can distinguish the DBDH tuple from a random tuple with non-negligible advantage $\frac{\varepsilon}{2}$.

[4] w.l.o.g. $w_x = \{n+1, \cdots, n+c-k_x\}$.

We first let the challenger set the groups G and G_T with an efficient bilinear map e and a generator g. The challenger flips a fair binary coin ν, outside of $\mathcal{S}$'s view. If $\nu = 1$, the challenger sets $(g, A, B, C, Z) \in \mathcal{D}_{bdh}$; otherwise it sets $(g, A, B, C, Z) \in \mathcal{D}_{rand}$.

Init The simulator $\mathcal{S}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses a challenge (d, c)-bounded access tree $\mathcal{T}^*$ it wishes to be challenged upon. The simulator $\mathcal{S}$ sets $Y = e(A, B) = e(g, g)^{ab}$. Then, $\mathcal{S}$ generates a (d, c)-universal access tree $\mathcal{T}_u$ and a map from $\mathcal{T}^*$ to $\mathcal{T}_u$. Randomly choose $\{r_{j,x}\}_{j\in\mathcal{U}, x\in\Sigma_{\mathcal{T}_u}}, \{r^*_{j,x}\}_{j\in\mathcal{U}^*, x\in\Sigma_{\mathcal{T}_u}}$ from Z_p.

For $j \in \mathcal{U}, x \in \Sigma_{\mathcal{T}_u}$,

$$T_{j,x} = \begin{cases} g^{r_{j,x}}, & \text{if } x = \text{map}(x'), x' \in \Sigma_{\mathcal{T}^*}, f(j, x') = 1; \\ B^{r_{j,x}}, & \text{otherwise.} \end{cases}$$

For $j \in \mathcal{U}^*, x \in \Sigma_{\mathcal{T}_u}$,

$$T^*_{j,x} = \begin{cases} g^{r^*_{j,x}}, & \text{if } j \in w_{x'}, x = \text{map}(x'), x' \in \Sigma_{\mathcal{T}^*}; \\ B^{r^*_{j,x}}, & \text{otherwise.} \end{cases}$$

Phase 1 $\mathcal{A}$ adaptively makes query for a secret key related with an attribute set γ such that $\mathcal{T}^*(\gamma) = 0$. To generate the secret key, $\mathcal{S}$ needs to assign a polynomial q_x for every non-leaf node in $\mathcal{T}$ and outputs a piece of secret key according to each non-leaf node.

With an attribute set γ as input, for any node $x \in \mathcal{T}_u$, we call a node x: a unsatisfied node iff there exists a unsatisfied node x' in $\mathcal{T}^*$ such that $\text{map}(x') = x$; a satisfied node iff there exists a satisfied node x' in $\mathcal{T}^*$ such that $\text{map}(x') = x$; a non-mapped node iff there exists no node x' in $\mathcal{T}^*$ so that $\text{map}(x') = x$. We define the following three procedures: PolyUnsat, PolySat and PolyNotCare.

For $j \in \gamma$,

$$D_{j,x} = \begin{cases} g^{\frac{bq_x(j)}{r_{j,x}}} = B^{\frac{q_x(j)}{r_{j,x}}}, & \text{if } f(j, x') = 1 \\ g^{\frac{bq_x(j)}{br_{j,x}}} = (g^{q_x(j)})^{\frac{1}{r_{j,x}}}, & \text{otherwise.} \end{cases}$$

For $j \in \mathcal{U}^*$,

$$D^*_{j,x} = \begin{cases} g^{\frac{bq_x(j)}{r^*_{j,x}}} = B^{\frac{q_x(j)}{r^*_{j,x}}}, & \text{if } j \in \omega_{x'}; \\ g^{\frac{bq_x(j)}{br^*_{j,x}}} = (g^{q_x(j)})^{\frac{1}{r^*_{j,x}}}, & \text{otherwise.} \end{cases}$$

Then, for each non-leaf child node z of x in $\mathcal{T}_u$,
If z is a non-mapped node,
PolyNotCare$(\mathcal{T}_z, \gamma, g^{q_x(\text{index}(z))})$;
If z is a satisfied node,
PolySat$(\mathcal{T}_z, \gamma, q_x(\text{index}(z)))$;
If z is a unsatisfied node,
PolyUnsat$(\mathcal{T}_z, \gamma, g^{q_x(\text{index}(z))})$.

Figure 5: PolyUnsat$(\mathcal{T}_x, \gamma, g^{\lambda_x})$

PolyUnsat$(\mathcal{T}_x, \gamma, g^{\lambda_x})$ for a unsatisfied node $x \in \Sigma_{\mathcal{T}}$ is defined as follows:

This procedure generates a polynomial q_x for a unsatisfied node x. We have a unsatisfied node x' such that $\text{map}(x') = x$. γ does not satisfy this access tree $\mathcal{T}^*_{x'}$, noted as $\mathcal{T}^*_{x'}(\gamma) = 0$, where $\mathcal{T}^*_{x'}$ is a subtree of $\mathcal{T}^*$. λ_x is an integer from Z_p. The unsatisfied root node x' have at most $k_{x'} - 1$ satisfied child node. Thus, it could implicitly sets $q_x(0) = \lambda_x$, and chooses $c - 1$ other points at random to completely fix q_x, including

- $c - k_{x'}$ points as $q_x(j)$ for the dummy nodes of x, where $j \in w_{x'}$;
- at most $k_{x'} - 1$ points as $q_x(\text{index}(z'))$ if z' is a satisfied leaf child node of x' or $q_x(\text{index}(z') + n + c - 1)$ where z' is a satisfied non-leaf child node of x'.

It executes the following steps in Figure 5.

PolySat$(\mathcal{T}_x, \gamma, \lambda_x)$ for a satisfied node $x \in \Sigma_{\mathcal{T}}$ is defined as follows:

This procedure generates a polynomial q_x for a satisfied node x. We have a satisfied node x' such that $\text{map}(x') = x$. λ_x is an integer from Z_p. It sets $q_x(0) = \lambda_x$, and chooses $c - 1$ other points at random to completely fix q_x. Thus, for each $j \in \mathcal{U} \cup \mathcal{U}^* \cup \{n + c, \cdots, n + 2c - 1\}$, we could obtain $q_x(j)$.

It executes the following steps in Figure 6.

For $j \in \gamma$,

$$D_{j,x} = \begin{cases} g^{\frac{bq_x(j)}{r_{j,x}}} = B^{\frac{q_x(j)}{r_{j,x}}}, & \text{if } f(j, x') = 1; \\ g^{\frac{bq_x(j)}{br_{j,x}}} = g^{\frac{q_x(j)}{r_{j,x}}}, & \text{otherwise.} \end{cases}$$

For $j \in \mathcal{U}^*$,

$$D^*_{j,x} = \begin{cases} g^{\frac{bq_x(j)}{r^*_{j,x}}} = B^{\frac{q_x(j)}{r^*_{j,x}}}, & \text{if } j \in \omega_{x'}; \\ g^{\frac{bq_x(j)}{br^*_{j,x}}} = g^{\frac{q_x(j)}{r^*_{j,x}}}, & \text{otherwise.} \end{cases}$$

Then, for each non-leaf child node z of x in $\mathcal{T}_u$,
If z is a non-mapped node,
PolyNotCare$(\mathcal{T}_z, \gamma, g^{q_x(\text{index}(z))})$;
If z is a satisfied node,
PolySat$(\mathcal{T}_z, \gamma, q_x(\text{index}(z)))$;
If z is a unsatisfied node,
PolyUnsat$(\mathcal{T}_z, \gamma, g^{q_x(\text{index}(z))})$.

Figure 6: PolySat$(\mathcal{T}_x, \gamma, \lambda_x)$

PolyNotCare$(\mathcal{T}_x, \gamma, g^{\lambda_x})$ for a non-mapped node $x \in \Sigma_{\mathcal{T}}$ is defined as follows:

This procedure generates a polynomial q_x for a non-mapped node x. It implicitly sets $q_x(0) = \lambda_x$, and chooses $c - 1$ other points at random to implicitly fix q_x. Thus, for $j \in \mathcal{U} \cup \mathcal{U}^* \cup \{n + c, \cdots, n + 2c - 1\}$, we could obtain $g^{q_x(j)}$. It outputs the following secret keys:

$$\{D_{j,x} = g^{\frac{bq_x(j)}{br_{j,x}}} = (g^{q_x(j)})^{\frac{1}{r_{j,x}}}\}_{j\in\gamma}, \{D^*_{j,x} = g^{\frac{bq_x(j)}{br^*_{j,x}}} = (g^{q_x(j)})^{\frac{1}{r^*_{j,x}}}\}_{j\in\mathcal{U}^*}$$

Then, for each non-leaf child node z of x in $\mathcal{T}_u$, it calls PolyNotCare$(\mathcal{T}_z, \gamma, g^{q_x(\text{index}(z))})$.

To give a secret key for an attribute set γ, $\mathcal{S}$ first runs PolyUnsat$(\mathcal{T}_r = \mathcal{T}, \gamma, A)$. Notice that we implicitly set $y = ab$ by $Y = e(A, B) = e(g, g)^y$. The secret key corresponding to each non-leaf node is recursively given by the above three procedures. Finally, it outputs

$$D = \langle \gamma, \{D_{j,x}\}_{j\in\gamma, x\in\Sigma_{\mathcal{T}_u}}, \{D^*_{j,x}\}_{j\in\mathcal{U}^*, x\in\Sigma_{\mathcal{T}_u}} \rangle$$

Therefore, $\mathcal{S}$ can answer each secret key query with an attribute set γ, where $\mathcal{T}^*(\gamma) = 0$. The distribution of these secret keys are identical to those in the real environment.

Challenge The adversary $\mathcal{A}$ will submit two challenge messages m_0 and m_1 to $\mathcal{S}$. Then, $\mathcal{S}$ chooses $\mu \in \{0,1\}$ at random, and returns an encryption of m_μ under the challenge access tree $\mathcal{T}^*$. The challenge ciphertext E is formed as:

$\langle \mathcal{T}^*, E' = m_\mu \cdot Z, \{E_{j,x} = C^{r_{j,\mathrm{map}(x)}}\}_{j\in\mathcal{U}, x\in\Sigma_{\mathcal{T}^*}, f(j,x)=1}, \{E^*_{j,x} = C^{r^*_{j,\mathrm{map}(x)}}\}_{j\in\omega_x, x\in\Sigma_{\mathcal{T}^*}}\rangle$

If $(g, A, B, C, Z) \in \mathcal{D}_{bdh}$ and we let $s = c$, then we have $Y^s = e(g,g)^{abc}$ and $E_{j,x} = C^{r_{j,\mathrm{map}(x)}} = (g^{r_{j,\mathrm{map}(x)}})^s$, $E^*_{j,x} = C^{r^*_{j,\mathrm{map}(x)}} = (g^{r^*_{j,\mathrm{map}(x)}})^s$. Therefore, the ciphertext is a valid random encryption of message m_μ

Otherwise, if $(g, A, B, C, Z) \in \mathcal{D}_{rand}$, we have $E' = m_\mu \cdot Z$. Since Z is randomly chosen from G_T, E' will be a random element of G_T from the adversary's view and the ciphertext contains no information about m_μ.

Phase 2 The simulator $\mathcal{S}$ acts exactly as it did in **Phase 1**.

Guess $\mathcal{S}$ outputs $\nu' = 1$ to indicate that it was given a tuple from $\mathcal{D}_{bdh}$ if $\mathcal{A}$ gives a correct guess $\mu' = \mu$; otherwise output $\nu' = 0$ to indicate that it was given a tuple from $\mathcal{D}_{rand}$.

Let us compute the success probability of $\mathcal{S}$:

In the case of $\nu = 0$ the adversary gains no information about μ. Therefore, we have $\Pr[\mu \neq \mu' | \nu = 0] = \frac{1}{2}$. Since the simulator guesses $\nu' = 0$ when $\mu \neq \mu'$, we have $\Pr[\nu' = \nu | \nu = 0] = \Pr[\nu' = 0 | \nu = 0] = \frac{1}{2}$.

In the case of $\nu = 1$, the adversary gets a valid ciphertext of m_μ. By definition, the adversary has ε to guess the correct μ', and thus $\Pr[\mu = \mu' | \nu = 1] = \frac{1}{2} + \varepsilon$. Since the simulator guesses $\nu' = 1$ when $\mu = \mu'$, we have $\Pr[\nu' = \nu | \nu = 1] = \Pr[\nu' = 1 | \nu = 1] = \frac{1}{2} + \varepsilon$.

The overall advantage of the simulator to output a correct $\nu' = \nu$ is $\Pr[\nu = \nu'] - \frac{1}{2} = \Pr[\nu = \nu', \nu = 0] + \Pr[\nu = \nu', \nu = 1] - \frac{1}{2} = \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot(\frac{1}{2} + \varepsilon) - \frac{1}{2} = \frac{\varepsilon}{2}$ □

5. BCPABE SCHEME $\mathtt{BCP}_2$ WITH CHOSEN CIPHERTEXT SECURITY

5.1 Extended BCPABE scheme $\mathtt{BCP}_2$

Now, by using one-time signature technique introduced in Section 2.3, we presented an extended scheme $\mathtt{BCP}_2$ achieving chosen ciphertext security.

The selective-tree CCA model was introduced in Section 3.2 and the similar security model can be found in [6].

We assume that there exist a bounded ciphertext policy attribute based encryption scheme $\mathtt{BCP}_1$ secure in the selective-tree CPA model as presented in Section 3.2, including four algorithms ($\mathtt{BCP}_1$.**Setup**, $\mathtt{BCP}_1$.**KGen**, $\mathtt{BCP}_1$.**Enc**, $\mathtt{BCP}_1$.**Dec**) and a secure one-time signature `ots` including three algorithms (`ots`.**KGen**, `ots`.**Sig**, `ots`.**Ver**).

Assume that the verification key vk from `ots` is a bit string of length l, and we write vk_i for the i-th bit in vk. Let $\mathcal{L}$ denote $\{1, 2, \cdots, l\}$. $\mathtt{BCP}_2$ constructed based on $\mathtt{BCP}_1$ and `ots` including the following algorithms:

Setup(d, c) This algorithm takes two system parameters (d, c) as its input. Then, it calls $\mathtt{BCP}_1$.**Setup**(d, c) to generate $\mathtt{BCP}_1$'s public parameter PP_1 and master key MK_1. In addition, it randomly chooses a set $\{t'_i\}_{i\in\{1,\cdots,2l\}}$ from Z_p and defines $T'_i = g^{t'_i}$. Now, it outputs the public parameter $\mathrm{PP} = \langle \mathrm{PP}_1, \{T'_i\}_{i\in\{1,\cdots,2l\}}\rangle$ and keeps the master key $\mathrm{MK} = \langle \mathrm{MK}_1, \{t'_i\}_{i\in\{1,\cdots,2l\}}\rangle$ to itself.

KGen(γ, MK) This algorithm takes an attribute set γ and the master key MK as its input. Then, it randomly chooses r' from Z_p and calls $\mathtt{BCP}_1$.**KGen**(γ, MK_1) to generate a user's secret key D_1 by using r' instead of y in MK_1 (i.e. $q_r(0) = r'$). For every $i \in \mathcal{L}$, let $D_{i,0} = g^{\frac{r_i}{t'_i}}$ and $D_{i,1} = g^{\frac{r_i}{t'_{l+i}}}$, where $\{r_i\}_{i\in\mathcal{L}}$ are randomly chosen from Z_p. Define $r = r' + \sum_{i\in\mathcal{L}} r_i$ and let $\hat{D} = g^{y-r}$. Finally, it outputs $D = \langle D_1, \{D_{i,0}, D_{i,1}\}_{i\in\mathcal{L}}, \hat{D}\rangle$.

Enc$(m, \mathrm{PP}, \mathcal{T})$ This algorithm takes a message m, the public parameter PP and a (d, c)-bounded access tree $\mathcal{T}$ as its input.

It first calls $\mathtt{BCP}_1$.**Enc**$(m, \mathrm{PP}_1, \mathcal{T})$ and obtains a partial ciphertext E_1. Then, a key pair $\langle sk, vk\rangle$ is obtained by running `ots`.**KGen**. For each $i \in \mathcal{L}$, it sets $E'_i = T'^s_i$ if $vk_i = 0$; $E'_i = T'^s_{l+i}$ otherwise. Let $\hat{E} = g^s$. [5] It runs `ots`.**Sig** with input $(sk, \langle E_1, \{E'_i\}_{i\in\mathcal{L}}, \hat{E}\rangle$ and obtain σ.

The output ciphertext $E = \langle E_1, \{E'_i\}_{i\in\mathcal{L}}, \hat{E}, \sigma, vk\rangle$.

Dec(D, E) This algorithm takes a secret key D and a ciphertext E as its input.

It first checks if σ is a valid signature on message $\langle E_1, \{E'_i\}_{i\in\mathcal{L}}, \hat{E}\rangle$ using vk. If valid, it proceeds the following; otherwise, output $\perp$.

It separates D_1 and E_1 from tuple (D, E) and consequently decrypts $e(g,g)^{sr'}$ according to $\mathtt{BCP}_1$.**Dec**.

For each $i \in \mathcal{L}$, it computes

$$e(g,g)^{sr_i} = \begin{cases} e(D_{i,0}, T'^s_i) = e(g^{\frac{r_i}{t'_i}}, g^{t'_i s}), & \text{if } vk_i = 0; \\ e(D_{i,1}, T'^s_{l+i}) = e(g^{\frac{r_i}{t'_{l+i}}}, g^{t'_{l+i} s}), & \text{if } vk_i = 1. \end{cases}$$

Finally, it computes $m = \frac{E'}{e(\hat{E},\hat{D})\cdot e(g,g)^{sr'}\cdot\prod_{i\in\mathcal{L}} e(g,g)^{sr_i}}$.

Compared with $\mathtt{BCP}_1$, $\mathtt{BCP}_2$'s ciphertext is augmented with l elements, while public parameter and secret key are both augmented $2l$ elements. Another method mentioned in [11] could be also used for extending $\mathtt{BCP}_1$ with CCA security level. It treats each verification key as an attribute. However, it has shorter additional size of ciphertext (1 element) but larger additional size of public parameter and secret key (2^l elements).

5.2 Security Proof of $\mathtt{BCP}_2$

The selective-tree CCA model is introduced in Section 3.2. We prove $\mathtt{BCP}_2$'s security based on the strong existentially unforgeable assumption of `ots` and the Decisional Bilinear Diffie-Hellman assumption.

THEOREM 2. *Suppose `ots` is a ε_{ots}-secure one-time signature scheme defined in Section 2.3. If the DBDH assumption holds in (G, G_T), then scheme $\mathtt{BCP}_2$ is selective-tree CCA secure in the standard model.*

PROOF. Suppose there exists a polynomial-time adversary $\mathcal{A}$ who can attack $\mathtt{BCP}_2$ in the selective-tree CCA model with non-negligible advantage ε. We construct a simulator $\mathcal{S}$ who can distinguish the DBDH tuple from a random tuple with non-negligible advantage $\frac{\varepsilon}{2} - \varepsilon_{ots}$.

[5] Here, s is consistent with the random value in E_1

We first let the challenger set the groups G and G_T with an efficient bilinear map e and a generator g. The challenger flips a fair binary coin ν, outside of $\mathcal{S}$'s view. If $\nu = 1$, the challenger sets $(g, A, B, C, Z) \in \mathcal{D}_{bdh}$; otherwise it sets $(g, A, B, C, Z) \in \mathcal{D}_{rand}$.

Init The simulator $\mathcal{S}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses a (d, c)-bounded access tree $\mathcal{T}^*$ it wishes to be challenged upon. $\mathcal{S}$ runs $\mathtt{ots}.\mathbf{KGen}$ to obtain $\langle sk^*, vk^* \rangle$. $\mathcal{S}$ sets $Y = e(g,g)^{ab} = e(A,B)$. $\mathcal{S}$ defines a (d, c)-universal access tree $\mathcal{T}$ and a map from $\mathcal{T}^*$ to $\mathcal{T}$. Then, it generates PP_1 as **Init** step in Section 4.2.

For $i \in \mathcal{L}$, randomly choose $\eta_i, \theta_i \in Z_p$ and implicitly set if $vk_i^* = 0$,

$$t'_i = \eta_i, T'_i = g^{\eta_i} \text{ and } t'_{l+i} = b\theta_{l+i}, T'_{l+i} = B^{\theta_{l+i}};$$

if $vk_i^* = 1$,

$$t'_i = b\eta_i, T'_i = B^{\eta_i} \text{ and } t'_{l+i} = \theta_{l+i}, T'_{l+i} = g^{\theta_{l+i}}.$$

The algorithm outputs public parameter

$$\mathrm{PP} = \langle \mathrm{PP}_1, \{T'_i\}_{i \in \{1, \cdots, 2l\}} \rangle$$

Phase 1 $\mathcal{A}$ is allowed to make secret key queries and decryption queries:

- Secret Key Query. $\mathcal{A}$ submits an attribute set γ such that $\mathcal{T}^*(\gamma) = 0$. $\mathcal{S}$ randomly chooses $r'', r'_i \in Z_p$ for $i \in \mathcal{L}$ and implicitly sets: $r' = ab + br'', r_i = br'_i$.

  According to the Phase 1 in Section 4, it calls $\mathrm{PolyUnsat}(\mathcal{T}_u, \gamma, A \cdot g^{r''})$ and obtains D_1. Then, it computes:

  $$\hat{D} = g^{y-r} = g^{ab - ab - br'' - \sum_{i \in \mathcal{L}} br'_i} = \frac{1}{B^{r'' + \sum_{i \in \mathcal{L}} r'_i}}$$

  and for $i \in \mathcal{L}$, if $vk_i^* = 0$,

  $$D_{i,0} = B^{\frac{r'_i}{\eta_i}} \text{ and } D_{i,1} = g^{\frac{r'_i}{\theta_{l+i}}};$$

  if $vk_i^* = 1$

  $$D_{i,0} = g^{\frac{r'_i}{\eta_i}} \text{ and } D_{i,1} = B^{\frac{r'_i}{\theta_{l+i}}}.$$

  Finally, it outputs the secret key

  $$D = \langle D_1, \{D_{i,0}, D_{i,1}\}_{i \in \mathcal{L}}, \hat{D} \rangle.$$

- Decryption Query. $\mathcal{A}$ submits a ciphertext $E = \langle E_1, \{E'_i\}_{i \in \mathcal{L}}, \hat{E}, \sigma, vk \rangle$ related with $\mathcal{T}$. $\mathcal{S}$ checks the signature σ using vk. If σ is invalid, S outputs $\perp$; otherwise $\mathcal{S}$ checks if $vk = vk^*$. If so, we call it **forge** event and $\mathcal{S}$ outputs $\nu' = 0$ to indicate that it was given a tuple from $\mathcal{D}_{rand}$. Now, the only case is $vk \neq vk^*$, $\mathcal{S}$ defines an attribute set γ such that $\mathcal{T}(\gamma) = 1$, if $\mathcal{T}^*(\gamma) = 0$, it will generates a secret key related with γ from Secret Key Query and use it to decrypt E; otherwise $\mathcal{T}^*(\gamma) = 1$, w.l.o.g. assuming $vk_j = 1, vk_j^* = 0$, $\mathcal{S}$ would generate a partial secret key for decrypting E as follows:

  - $\mathcal{S}$ randomly chooses $r'', r'_i \in Z_p$ for $i \in \mathcal{L}$ and implicitly sets: $r' = br'', r_i = br'_i$ for $i \neq j$ and $r_j = ab + br'_j$.
  - According to the Phase 1 in Section 4.2, it calls $\mathrm{PolySat}(\mathcal{T}, \gamma, r'')$ and obtains D_1 related with γ. Now, it could use D_1 decrypts E_1 and obtains $e(g,g)^{sr'}$.
  - For $i \in \mathcal{L}$ and $i \neq j$, if $vk_i^* = 0$,

    $$D_{i,0} = B^{\frac{r'_i}{\eta_i}} \text{ and } D_{i,1} = g^{\frac{r'_i}{\theta_{l+i}}};$$

    if $vk_i^* = 1$,

    $$D_{i,0} = g^{\frac{r'_i}{\eta_i}} \text{ and } D_{i,1} = B^{\frac{r'_i}{\theta_{l+i}}}.$$

  - For $i = j$, it only can generate $D_{j,1} = g^{\frac{r_j}{t'_{l+i}}} = g^{\frac{ab + br'_j}{b\theta_{l+j}}} = A^{\frac{1}{\theta_{l+j}}} \cdot g^{\frac{r'_j}{\theta_{l+j}}}$. This is enough for finishing our decryption since $vk_j = 1$ and T'^s_{l+j} is involved in ciphertext E. Note that $e(D_{j,1}, T'^s_{l+j}) = e(g,g)^{sr_j}$.
  - $\mathcal{S}$ then computes $\hat{D} = g^{y-r} = g^{ab - br'' - ab - \sum_{i \in \mathcal{L}} br'_i} = \frac{1}{B^{r'' + \sum_{i \in \mathcal{L}} r'_i}}$.
  - Finally, $\mathcal{S}$ outputs $m = \frac{E'}{e(\hat{E}, \hat{D}) \cdot e(g,g)^{sr'} \cdot \prod_{i \in \mathcal{L}} e(g,g)^{sr_i}}$.

Challenge The adversary $\mathcal{A}$ will submit two challenge messages m_0 and m_1 to $\mathcal{S}$. Then, $\mathcal{S}$ chooses $\mu \in \{0, 1\}$ at random, and returns an encryption of m_μ under the challenge access tree $\mathcal{T}^*$ as follows:

- It first generates an encryption E_1^* according to **Challenge** in Section 4.2.
- It generates a signature $\sigma^* = \mathtt{ots}.\mathbf{Sig}(sk^*, \langle E_1^*, \{C^{\eta_i}\}_{i \in \mathcal{L}, vk_i^* = 0}, \{C^{\theta_{l+i}}\}_{i \in \mathcal{L}, vk_i^* = 1}, C \rangle)$.
- It outputs the challenge ciphertext as $E^* = \langle E_1^*, \{C^{\eta_i}\}_{i \in \mathcal{L}, vk_i^* = 0}, \{C^{\theta_{l+i}}\}_{i \in \mathcal{L}, vk_i^* = 1}, C, \sigma^*, vk^* \rangle$

If $(g, A, B, C, Z) \in \mathcal{D}_{bdh}$ and we let $s = c$, then we have $Y^s = e(g,g)^{abc}$, $E'_i = C^{\eta_i} = (g^s)^{\eta_i} = T'^s_i$ and $E'_i = C^{\theta_{l+i}} = (g^s)^{\theta_{l+i}} = T'^s_{l+i}$. Therefore, the ciphertext is a valid random encryption of message m_μ.

Otherwise, if $(g, A, B, C, Z) \in \mathcal{D}_{rand}$, we have $E' = m_\mu \cdot Z$. Since Z is randomly chosen from G_T, E' will be a random element of G_T from the adversary's view and the ciphertext contains no information about m_μ.

Phase 2 The simulator $\mathcal{S}$ acts exactly as it did in **Phase 1**.

Guess $\mathcal{S}$ outputs $\nu' = 1$ to indicate that it was given a tuple from $\mathcal{D}_{bdh}$ if $\mathcal{A}$ gives a correct guess $\mu' = \mu$; otherwise output $\nu' = 0$ to indicate that it was given a tuple from $\mathcal{D}_{rand}$.

Let us compute the success probability of $\mathcal{S}$:

In the case of $\nu = 0$ the adversary gains no information about μ. Therefore, we have $\Pr[\mu \neq \mu' | \nu = 0] = \frac{1}{2}$. Since the simulator guesses $\nu' = 0$ when ($\mu \neq \mu'$, no **forge**) or (**forge**), we have $\Pr[\nu' = \nu | \nu = 0] = \Pr[\nu' = 0 | \nu = 0] = \Pr[\mu \neq \mu', \neg\text{forge} | \nu = 0] + \Pr[\textbf{forge} | \nu = 0] = \Pr[\mu \neq \mu' | \nu = 0] - \Pr[\mu \neq \mu', \textbf{forge} | \nu = 0] + \Pr[\textbf{forge} | \nu = 0] \geq \frac{1}{2} - \Pr[\textbf{forge} | \nu = 0] = \frac{1}{2} - \varepsilon_{ots}$.

In the case of $\nu = 1$, the adversary gets a valid ciphertext of m_μ. By definition, the adversary has ε to guess the correct

μ', and thus $\Pr[\mu = \mu'|\nu = 1] = \frac{1}{2} + \varepsilon$. Since the simulator guesses $\nu' = 1$ when ($\mu = \mu'$, no **forge**), we have $\Pr[\nu' = \nu|\nu = 1] = \Pr[\nu' = 1|\nu = 1] = \Pr[\mu = \mu', \neg\textbf{forge}|\nu = 1] = \Pr[\mu = \mu'|\nu = 1] - \Pr[\mu = \mu', \textbf{forge}|\nu = 1] \geq \frac{1}{2} + \varepsilon - \Pr[\textbf{forge}|\nu = 1] = \frac{1}{2} + \varepsilon - \varepsilon_{ots}$.

The overall advantage of the simulator to output a correct $\nu' = \nu$ is $\Pr[\nu = \nu'] - \frac{1}{2} = \Pr[\nu = \nu', \nu = 0] + \Pr[\nu = \nu', \nu = 1] - \frac{1}{2} \geq \frac{1}{2} \cdot (\frac{1}{2} - \varepsilon_{ots}) + \frac{1}{2} \cdot (\frac{1}{2} + \varepsilon - \varepsilon_{ots}) - \frac{1}{2} = \frac{\varepsilon}{2} - \varepsilon_{ots}$ □

6. COMPARISONS

6.1 Access Policy

In this section, we compare the expressive capability of access tree of $\mathtt{BCP}_1$ with that of GJPS scheme bounded by the same parameter (d, c). Actually, according to $\mathtt{BCP}_1$'s definition on the (d, c)-bounded access tree, each non-leaf node has a threshold value at most c and no more than c non-leaf nodes share one unique parent. Thus, one difference of the restriction on access trees between GJPS and $\mathtt{BCP}_1$ is if a non-leaf node x has a non-leaf child node, in GJPS the total number of x's child node must be no more than c while in $\mathtt{BCP}_1$ the total number of x's non-leaf child node must be no more than c. Therefore, our scheme accommodate more possible access policies chosen by the sender under the same pre-set bounds. (An example is shown in Figure 7)

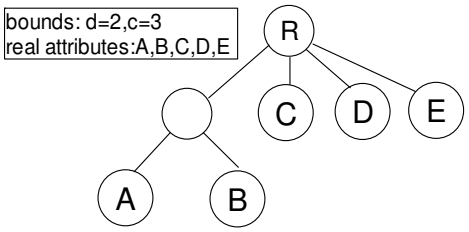

Figure 7: An example access tree accepted by $\mathtt{BCP}_1$ but not accepted by GJPS

6.2 Efficiency and Parameter Size

Now, we present the comparisons on the computational cost of each algorithm and the sizes of parameters between GJPS and $\mathtt{BCP}_1$ in Table 1, both of which are proved secure in the selective-tree CPA model. We assume that both schemes are initialized with same system parameters (d, c) and consider an encryption under a (d, c)-bounded access tree $\mathcal{T}$(this access tree must be chosen suitable for both schemes since there's a difference between the acceptable access trees of two schemes). $\mathcal{T}_n$ is $\mathcal{T}$'s normal form. The secret key is associated with an attribute set γ such that $\mathcal{T}(\gamma) = 1$ and $|\gamma| = x$. $\hat{\mathcal{T}}$, $\hat{\mathcal{T}}_n$ are the γ-satisfied non-redundant trees of $\mathcal{T}, \mathcal{T}_n$ with minimum non-leaf nodes, respectively. $|\mathcal{U}| = n, |\mathcal{U}^*| = c - 1, |\Sigma_{\mathcal{T}_u}| = \frac{c^d-1}{c-1}$. Here, TExp represents the cost of one modular exponentiation, TPair represents the cost of one bilinear pairing computing. $T_{S1}, T_{K1}, T_{E1}, T_{D1}$ represents the computational cost of **Setup**, **KGen**, **Enc** and **Dec** algorithms in GJPS. L_{P1}, L_{S1}, L_{C1} represents the size of public parameter, secret key and ciphertext in GJPS. The mark with "2" indicates the counterpart of $\mathtt{BCP}_1$.

Mark	times	Computational cost	
T_{S1}	1	$\mathsf{TPair} + (nc^{d-1} + c^d) \cdot \mathsf{TExp}$	✓
T_{S2}	1	$\mathsf{TPair} + (n \times \frac{c^d-1}{c-1} + c^d) \cdot \mathsf{TExp}$	
T_{K1}	1/user	$(xc^{d-1} + c^d - 1) \cdot \mathsf{TExp}$	✓
T_{K2}	1/user	$(x \times \frac{c^d-1}{c-1} + c^d - 1) \cdot \mathsf{TExp}$	
T_{E1}	many	$(1 + \lvert\Theta_{\mathcal{T}_n}\rvert + \sum_{x \in \Sigma_{\mathcal{T}_n}} (c - k_x)) \cdot \mathsf{TExp}$	
T_{E2}	many	$(1 + \lvert\Theta_{\mathcal{T}}\rvert + \sum_{x \in \Sigma_{\mathcal{T}}} (c - k_x)) \cdot \mathsf{TExp}$	✓
T_{D1}	many	$(\lvert\Sigma_{\hat{\mathcal{T}}_n}\rvert \times c - \lvert\Sigma_{\hat{\mathcal{T}}_n}\rvert + 1) \cdot \mathsf{TPair} + \lvert\Sigma_{\hat{\mathcal{T}}_n}\rvert \times c \cdot \mathsf{TExp}$	
T_{D2}	many	$(\lvert\Sigma_{\hat{\mathcal{T}}}\rvert \times c - \lvert\Sigma_{\hat{\mathcal{T}}}\rvert + 1) \cdot \mathsf{TPair} + \lvert\Sigma_{\hat{\mathcal{T}}}\rvert \times c \cdot \mathsf{TExp}$	✓
	number	Size	
L_{P1}	1	$\lvert G_T\rvert + (nc^{d-1} + c^d - 1) \cdot \lvert G\rvert$	✓
L_{P2}	1	$\lvert G_T\rvert + (n \times \frac{c^d-1}{c-1} + c^d - 1) \cdot \lvert G\rvert$	
L_{S1}	1/user	$(xc^{d-1} + c^d - 1) \cdot \lvert G\rvert$	✓
L_{S2}	1/user	$(x \times \frac{c^d-1}{c-1} + c^d - 1) \cdot \lvert G\rvert$	
L_{C1}	many	$\lvert G_T\rvert + (\lvert\Theta_{\mathcal{T}_n}\rvert + \sum_{x \in \Sigma_{\mathcal{T}_n}} (c - k_x)) \cdot \lvert G\rvert$	
L_{C2}	many	$\lvert G_T\rvert + (\lvert\Theta_{\mathcal{T}}\rvert + \sum_{x \in \Sigma_{\mathcal{T}}} (c - k_x)) \cdot \lvert G\rvert$	✓

Table 1: Comparisons between the scheme in GJPS and $\mathtt{BCP}_1$

Generally, c is set to be no less than 2, then $x \times \frac{c^d-1}{c-1} + c^d - 1 < 2 \times (xc^{d-1} + c^d - 1)$. This is because of the following deduction: $2 \leq c \Longrightarrow 2c^{d-1} - 1 < c^d \Longrightarrow \frac{c^d-1}{c-1} < 2c^{d-1}$. Therefore, we obtain that $T_{S2} < 2T_{S1}, T_{K2} < 2T_{K1}, L_{P2} < 2L_{P1}, L_{S2} < 2L_{S1}$.

7. CONCLUSION

Ciphertext policy attribute based encryption scheme differs from the traditional public key encryption scheme [2, 3, 4, 7, 8, 9] and key policy attribute based encryption scheme [11, 12], because it allows a sender to exert more control on the access policy of ciphertext. This is very efficient for users to distribute messages and also very applicable to the role-based access control system.

In this work, we have designed a provably secure bounded ciphertext policy attribute based encryption scheme $\mathtt{BCP}_1$ with shorter ciphertext and higher efficiency compared with previously proposed scheme [10]. The scheme $\mathtt{BCP}_1$ is proved selective-tree CPA secure in the standard model assuming that DBDH problem is hard. In addition, we apply the one-time signature techniques to obtain a chosen ciphertext secure extension $\mathtt{BCP}_2$. The security proof is reduced to the DBDH assumption and the strong existential unforgeability of one-time signature scheme.

Many problems remain open in this area. For example, it is imperative for continuingly improving the efficiency of BCPABE scheme. Another one, also referred as our future work is to design a new kind of BCPABE excluding the assistance of dummy nodes.

8. ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers and Rongxing Lu for their suggestions to improve this paper. Besides, this research is supported by National Nature Science Foundation of China (No.60673079 and No.60773086), National 973 Program (No.2007CB311201).

9. REFERENCES

[1] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[2] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[3] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[4] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[5] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[6] Ling Cheung and Calvin Newport. Provably secure ciphertext policy abe. In *ACM Conference on Computer and Communications Security*, pages 456–465, 2007.

[7] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[8] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, 1998.

[9] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

[10] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP (2)*, pages 579–591, 2008.

[11] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[12] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.

[13] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[14] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Cryptology ePrint Archive: 2008/290*, 2008.