Bounded Ciphertext Policy Attribute Based Encryption

Vipul Goyal[⋆], Abhishek Jain[⋆], Omkant Pandey[⋆], and Amit Sahai[⋆]

Department of Computer Science, UCLA
{vipul,abhishek,omkant,sahai}@cs.ucla.edu

Abstract. In a ciphertext policy attribute based encryption system, a user's private key is associated with a set of attributes (describing the user) and an encrypted ciphertext will specify an access policy over attributes. A user will be able to decrypt if and only if his attributes satisfy the ciphertext's policy.
In this work, we present the first construction of a ciphertext-policy attribute based encryption scheme having a security proof based on a number theoretic assumption and supporting advanced access structures. Previous CP-ABE systems could either support only very limited access structures or had a proof of security only in the generic group model. Our construction can support access structures which can be represented by a bounded size access tree with threshold gates as its nodes. The bound on the size of the access trees is chosen at the time of the system setup. Our security proof is based on the standard Decisional Bilinear Diffie-Hellman assumption.

1 Introduction

In many access control systems, every piece of data may legally be accessed by several different users. Such a system is typically implemented by employing a trusted server which stores all the data in clear. A user would log into the server and then the server would decide what data the user is permitted to access. However such a solution comes with a cost: what if the server is compromised? An attacker who is successful in breaking into the server can see all the sensitive data in clear.

One natural solution to the above problem is to keep the data on the server encrypted with the private keys of the users who are permitted to access it. However handling a complex access control policy using traditional public key encryption systems can be difficult. This is because the access policy might be described in terms of the *properties or attributes* that a valid user should have rather than in terms of the actual identities of the users. Thus, a priori, one may

[⋆] This research was supported in part from grants from the NSF ITR and Cybertrust programs (including grants 0627781, 0456717, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, an Alfred P. Sloan Foundation Fellowship, and an Okawa Foundation Research Grant.

not even know the exact list of users authorized to access a particular piece of data.

The concept of attribute based encryption (ABE) was introduced by Sahai and Waters [1] as a step towards developing encryption systems with high expressiveness. Goyal et al [2] further developed this idea and introduced two variants of ABE namely ciphertext-policy attribute based encryption (CP-ABE) and key-policy attribute based encryption (KP-ABE). In a CP-ABE system, a user's private key is associated with a set of attributes (describing the *properties* that the user has) and an encrypted ciphertext will specify an access policy over attributes. A user will be able to decrypt if and only if his attributes satisfy the ciphertext's policy. While a construction of KP-ABE was offered by [2], constructing CP-ABE was left as an important open problem.

Subsequently to Goyal et al [2], Bethencourt et al [3] gave the first construction of a CP-ABE system. Their construction however only had a security argument in the generic group model. Cheung and Newport [4] recently gave a CP-ABE construction supporting limited type of access structures which could be represented by **AND** of different attributes. Cheung and Newport also discussed the possibility of supporting more general access structures with threshold gates. However as they discuss, a security proof of this generalization would involve overcoming several subtleties. In sum, obtaining a CP-ABE scheme for more advanced access structures based on any (even relatively non-standard) number theoretic assumption has proven to be surprisingly elusive.

Our Results. We present the first construction of a ciphertext-policy attribute based encryption scheme having a security proof based on a standard number theoretic assumption and supporting advanced access structures. Our construction can support access structures which can be represented by a bounded size access tree with threshold gates as its nodes. The bound on the size of the access trees is chosen at the time of the system setup and is represented by a tuple (d, num) where d represents the maximum depth of the access tree and num represents the maximum number of children each non-leaf node of the tree might have. We stress that any access tree satisfying these upper bounds on the size can be dynamically chosen by the encrypter. Our construction has a security proof based on the standard Decisional Bilinear Diffie-Hellman (BDH) assumption. We note that previous CP-ABE systems could either support only very limited access structures [4] or had a proof of security only in the generic group model [3] (rather than based on a number theoretic assumption). Further, we show how to extend our constructions to support non-monotonic access policies. Finally, we observe that our constructions for non-monotonic access policies can in fact support any access formula with bounded polynomial size.

Our Techniques. Our construction can be seen as a way to reinterpret Key-Policy ABE schemes (e.g. [2]) with a fixed "universal" tree access structure as a CP-ABE scheme. Such a reinterpretation presents some problems because in a KP-ABE scheme, the key material for each attribute is "embedded" into the access structure in a unique way depending on where it occurs in the access policy. To overcome this difficulty, we introduce many "copies" of each attribute for every position in the access structure tree where it can occur. This causes a significant increase in private key size, but does not significantly affect ciphertext size. However, since the actual access structure to be used for a particular ciphertext must be embedded into the fixed "universal" tree access structure in the KP-ABE scheme, this causes a blowup in ciphertext size. This effect can be moderated by having multiple parallel CP-ABE schemes with different sized "universal" tree access structures underlying the scheme, which allows for a trade-off between ciphertext size and the size of the public parameters and private keys.

Note that in general a Boolean formula of size n can be represented by a balanced formula of size $O(n^{2/\log(3/2)})$ (roughly $O(n^{3.42})$). Thus, in general our methodology would yield a ciphertext blowup of $O(n^{3.42})$ group elements. As such, our result can be seen as a "feasibility result" for CP-ABE for general Boolean formulas of bounded size. We leave constructing more efficient CP-ABE schemes based on number-theoretic assumptions as an important open question.

2 Background

We first give formal definitions for the security of Bounded Ciphertext Policy Attribute Based Encryption (BCP-ABE). Then we give background information on bilinear maps and our cryptographic assumption. Like the work of Goyal et al. [2], we describe our constructions for access trees. Roughly speaking, given a set of attributes $\{P_1, P_2, \ldots, P_n\}$, an access tree is an access structure $\mathcal{T} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$, where each node in the tree represents a threshold gate (see Section 3 for a detailed description). We note that contrary to the work of Goyal et al., in our definitions, users will be identified with a set of attributes while access trees will be used to specify policies for encrypting data.

A Bounded Ciphertext Policy Attribute Based Encryption scheme consists of four algorithms.

Setup (d, num) This is a randomized algorithm that takes as input the implicit security parameter and a pair of system parameters (d, num). These parameters will be used to restrict the access trees under which messages can be encrypted in our system. It outputs the public parameters PK and a master key MK.
Key Generation (γ, MK) This is a randomized algorithm that takes as input – the master key MK and a set of attributes γ. It outputs a decryption key D corresponding to the attributes in γ.
Encryption $(M, \mathrm{PK}, \mathcal{T}')$ This is a randomized algorithm that takes as input – the public parameters PK, a message M, and an access tree $\mathcal{T}'$ over the universe of attributes, with depth $d' \leq d$, and where each non-leaf node x has at most num child nodes. The algorithm will encrypt M and output the ciphertext E. We will assume that the ciphertext implicitly contains $\mathcal{T}'$.
Decryption (E, D) This algorithm takes as input – the ciphertext E that was

encrypted under the access tree $\mathcal{T}'$, and the decryption key D for an attribute set γ. If the set γ of attributes satisfies the access tree $\mathcal{T}'$ (i.e. $\gamma \in \mathcal{T}'$), then the algorithm will decrypt the ciphertext and return a message M.

We now discuss the security of a bounded ciphertext-policy ABE scheme. We define a selective-tree model for proving the security of the scheme under the chosen plaintext attack. This model can be seen as analogous to the selective-ID model [5–7] used in identity-based encryption (IBE) schemes [8–10].

Selective-Tree Model for BCP-ABE. Let $\mathcal{U}$ be the universe of attributes fixed by the security parameter. The system parameters d, num are also defined.

Init The adversary declares the access tree $\mathcal{T}'$, that he wishes to be challenged upon.
Setup The challenger runs the Setup algorithm of ABE and gives the public parameters to the adversary.
Phase 1 The adversary is allowed to issue queries for private keys for many attribute sets γ_j, where γ_j does not satisfy the access tree $\mathcal{T}'$ for all j.
Challenge The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b with $\mathcal{T}'$. The ciphertext is passed on to the adversary.
Phase 2 Phase 1 is repeated.
Guess The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[b' = b] - \frac{1}{2}$. We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 1 *A bounded ciphertext-policy attribute-based encryption scheme (BCP-ABE) is secure in the Selective-Tree model of security if all polynomial time adversaries have at most a negligible advantage in the Selective-Tree game.*

2.1 Bilinear Maps

We present a few facts related to groups with efficiently computable bilinear maps. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_1$ and e be a bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}_1$ is a bilinear group if the group operation in $\mathbb{G}_1$ and the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.2 The Decisional Bilinear Diffie-Hellman (BDH) Assumption

Let $a, b, c, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}_1$. The decisional BDH assumption [7, 1] is that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^z)$ with more than a negligible advantage. The advantage of $\mathcal{B}$ is

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

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

3 Access Trees

In our constructions, user decryption keys will be identified with a set γ of attributes. A party who wishes to encrypt a message will specify through an access tree structure a policy that private keys must satisfy in order to decrypt. We now proceed to explain the access trees used in our constructions.

Access Tree. Let $\mathcal{T}$ be a tree representing an access structure. 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 $0 < k_x \leq num_x$. For ease of presentation, we use the term *cardinality* to refer to the number of children of a node. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

We fix the root of an access tree to be at level 0. Let $\Phi_{\mathcal{T}}$ denote the set of all the non-leaf nodes in the tree $\mathcal{T}$. Further, let $\Psi_{\mathcal{T}}$ be the set of all the non-leaf nodes at depth $d - 1$, where d is the depth of $\mathcal{T}$. To facilitate working with the access trees, we define a few functions. We denote the parent of the node x in the tree by $\text{parent}(x)$. The access tree $\mathcal{T}$ also defines an ordering between the children of every node, that is, the children of a node x are numbered from 1 to num_x. The function $\text{index}(x)$ returns such a number associated with a node x, where the index values are uniquely assigned to nodes in an arbitrary manner for a given access structure. For simplicity, we provision that $\text{index}(x) = \text{att}(x)$, when x is a leaf node and $\text{att}(x)$ is the attribute associated with it.

Satisfying an Access Tree. Let $\mathcal{T}$ be an access tree with root r. Denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at the node x. Hence $\mathcal{T}$ is the same as $\mathcal{T}_r$. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$ recursively as follows. 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 return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 iff $\text{att}(x) \in \gamma$.

Universal Access Tree. Given a pair of integer values (d, num), define a complete num-ary tree $\mathcal{T}$ of depth d, where each non-leaf node has a threshold value of num. The leaf nodes in $\mathcal{T}$ are empty, i.e., no attributes are assigned to the leaf nodes. Next, $num - 1$ new leaf nodes are attached to each non-leaf node x, thus increasing the cardinality of x to $2 \cdot num - 1$ while the threshold value num is left intact. Choose an arbitrary assignment of *dummy* attributes (explained later in Section 4) to these newly added leaf nodes[1] for each x. The resultant tree $\mathcal{T}$ is called a (d, num)-universal access tree (or simply the universal access tree when d, num are fixed by the system).

Bounded Access Tree. We say that $\mathcal{T}'$ is a (d, num)-bounded access tree if it has depth $d' \leq d$, and each non-leaf node in $\mathcal{T}'$ exhibits a cardinality at most num.

Normal Form. Consider a (d, num)-bounded access tree $\mathcal{T}'$. We say that $\mathcal{T}'$ exhibits the (d, num)-normal form if (a) it has depth $d' = d$, and (b) all the leaves in $\mathcal{T}'$ are at depth d. Any (d, num)-bounded access tree $\mathcal{T}'$ can be converted to the (d, num)-normal form (or simply the normal form when d, num are fixed by the system) in the following way in a top down manner, starting from the root node r'. Consider a node x at level l_x in $\mathcal{T}'$. If the depth d_x of the subtree $\mathcal{T}'_x$ is less than $(d - l_x)$, then insert a vertical chain of $(d - l_x - d_x)$ nodes (where each node has cardinality 1 and threshold 1) between x and parent(x). Repeat the procedure recursively for each child of x. Note that conversion to the normal form does not affect the satisfying logic of an access tree.

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

4 Small Universe Construction

Before we explain the details of our construction, we first present an overview highlighting the main intuitions behind our approach.

4.1 Overview of Our Construction

Fixed Tree Structure. We first give the outline of a basic target system. For simplicity, let us consider a very simple and basic access tree $\mathcal{T}$. The tree $\mathcal{T}$ has

[1] From now onwards, by *dummy nodes*, we shall refer to the leaf nodes with dummy attributes associated with them.

depth, say, d; and all leaf nodes in the tree are at depth d. Each non-leaf node x is a "k_x-out-of-num_x" threshold gate where k_x and num_x are fixed beforehand. Thus the "structure" of the access tree is *fixed*. However, the leaf nodes of $\mathcal{T}$ are "empty", i.e., no attributes are associated with them. At the time of encryption, an encrypter will assign attributes to each leaf-node, in order to define the access structure completely. That is, once the encrypter assigns an attribute to each leaf node in $\mathcal{T}$, it fixes the set of "authorized sets of attributes". A user having keys corresponding to an authorized set will be able to decrypt a message encrypted under the above access structure.

We now explain how such a system can be constructed. Recall that $\Psi_{\mathcal{T}}$ denotes the set of non-leaf nodes at depth $d-1$. Each leaf child z of an $x \in \Psi_{\mathcal{T}}$ will be assigned an attribute j.[2] However, the same j may be assigned to z_1 as well as z_2 where z_1 is a child node of $x_1 \in \Psi_{\mathcal{T}}$, and z_2 is a child node of $x_2 \in \Psi_{\mathcal{T}}$. Thus, any given attribute j may have at most $|\Psi_{\mathcal{T}}|$ distinct parent nodes. Intuitively, these are all the distinct positions under which j can appear as an attribute of a leaf in the tree $\mathcal{T}$. Now, during system setup, we will publish a unique public parameter corresponding to each such appearance of j, for each attribute j. Next, consider a user A with an attribute set γ. Imagine an access tree $\mathcal{T}'$ that has the same "structure" as $\mathcal{T}$, except that each node $x \in \Psi_{\mathcal{T}'}$ has cardinality $|\gamma|$ instead of num_x (while the threshold is still k_x). Additionally, each attribute $j \in \gamma$ is attached to a distinct leaf child of each node $x \in \Psi_{\mathcal{T}'}$. A will be assigned a private key that is computed for such an access tree $\mathcal{T}'$ as in the KP-ABE construction of Goyal et al [2]. Now, suppose that an encrypter $\mathcal{E}$ has chosen an assignment of attributes to the leaf nodes in $\mathcal{T}$ to define it completely. Let $f(j,x)$ be a function that outputs 1 if an attribute j is associated with a leaf child of x and 0 otherwise. Then, $\mathcal{E}$ will compute (using the public parameters published during system setup) and release a ciphertext component $E_{j,x}$ corresponding to an attribute j attached to a leaf child of $x \in \Psi_{\mathcal{T}}$ (i.e., iff $f(j,x) = 1$). A receiver who possesses an authorized set of attributes for the above tree can choose from his private key - the components $D_{j,x}$, such that $f(j,x) = 1$; and use them with corresponding $E_{j,x}$ during the decryption process.

Varying the Thresholds. The above system, although dynamic, may be very limited in its expressibility for some applications. In order to make it more expressible, we can further extend the above system as follows. At the time of encryption, an encrypter is now given the flexibility of choosing the threshold value between 1 and some maximum fixed value, (say) *num* for each node x in the access tree.

As a first step, we will construct $\mathcal{T}$ as a complete *num*-ary tree of depth d, where each non-leaf node is a "*num*-out-of-*num*" threshold gate. As earlier, the leaf nodes in the tree are empty. Next, we introduce a $(num-1)$-sized set of special attributes called *dummy* attributes [1] that are different from the usual

[2] For simplicity of exposition, we provision that an encrypter cannot assign the same attribute j to two child nodes z_1, z_2 of a given x. We note that this restriction can be removed by some simple modifications. Details will be given in the full version [11].

attributes (which we will henceforth refer to as *real* attributes[3]). Now, attach $(num-1)$ leaf nodes to each $x \in \Phi_{\mathcal{T}}$, and assign a dummy attribute to each such newly-added leaf node (henceforth referred to as dummy nodes).

Note that a dummy attribute j may have atmost $|\Phi_{\mathcal{T}}|$ parent nodes. Intuitively, these are all the distinct positions where j can appear as an attribute of a dummy leaf in $\mathcal{T}$. Therefore, during the system setup, for each dummy attribute j, we will publish a unique public parameter corresponding to *each appearance* of j (in addition to the public parameters corresponding to the real attributes as in the previous description). Next, consider a user A with an attribute set γ. Imagine an access tree $\mathcal{T}'$ that is similar to $\mathcal{T}$, except that each node $x \in \Psi_{\mathcal{T}'}$ has $|\gamma|$ leaf child (dummy nodes not inclusive) instead of num (while the threshold is still num). Additionally, each attribute $j \in \gamma$ is attached to a distinct leaf child of each node $x \in \Psi_{\mathcal{T}'}$. A will be assigned a private key that is computed for such an access tree $\mathcal{T}'$ as in the KP-ABE construction of Goyal et al [2] (the difference from the previous description for fixed trees is that here A will additionally receive key-components corresponding to dummy attributes). Now, at the time of encryption, an encrypter E will first choose a threshold value $k_x \leq num$ for each $x \in \Phi_{\mathcal{T}}$. Next, E will choose an assignment of real attributes to the leaf nodes in $\mathcal{T}$, and an arbitrary $(num - k_x)$-sized subset ω_x of dummy child nodes of each $x \in \Phi_{\mathcal{T}}$. Finally, E will release the ciphertext components as in the previous description. E will additionally release a ciphertext component corresponding to each dummy node in ω_x, for each $x \in \Phi_{\mathcal{T}}$. Now, consider a receiver with an attribute set γ. For any $x \in \Phi_{\mathcal{T}}$, if k_x children of x can be satisfied with γ, then the receiver can use the key-components (from his private key) corresponding to the dummy attributes in order to satisfy each dummy leaf $z \in \omega_x$; thus satisfying the num-out-of-num threshold gate x.

Varying Tree Depth and Node Cardinality. Finally, we note that the above system can be further extended to allow an encrypter to choose the depth of the access tree and also the cardinality of each node; thus further increasing the expressibility of the system. To do this, we will assume an upper bound on the maximum tree depth d and the maximum node cardinality num, fixed beforehand. We will then make use of the techniques presented in the latter part of Section 3 to achieve the desired features. Details are given in the construction.

4.2 The Construction

Let $\mathbb{G}_1$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. In addition, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. A security parameter, κ, will determine the size of the groups. We also define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We will associate each attribute with a unique element in $\mathbb{Z}_p^*$. Our construction follows.

[3] As the name suggests, we will identify users with a set of "real" attributes, while the dummy attributes will be used for technical purposes, i.e., varying the threshold of the nodes when needed.

Setup (d, num) This algorithm takes as input two system parameters, namely, (a) the maximum tree depth d, and (b) the maximum node cardinality num. The algorithm proceeds as follows. Define the universe of real attributes $\mathcal{U} = \{1, \ldots, n\}$, and a $(num-1)$-sized universe of dummy attributes[4] $\mathcal{U}^* = \{n+1, \ldots, n+num-1\}$. Next, define a (d, num)-universal access tree $\mathcal{T}$ as explained in the section 3. In the sequel, $d, num, \mathcal{U}, \mathcal{U}^*, \mathcal{T}$ will all be assumed as implicit inputs to all the procedures.

Now, for each real attribute $j \in \mathcal{U}$, choose a set of $|\Psi_{\mathcal{T}}|$ numbers $\{t_{j,x}\}_{x \in \Psi_{\mathcal{T}}}$ uniformly at random from $\mathbb{Z}_p$. Further, for each dummy attribute $j \in \mathcal{U}^*$, choose a set of $|\Phi_{\mathcal{T}}|$ numbers $\{t^*_{j,x}\}_{x \in \Phi_{\mathcal{T}}}$ uniformly at random from $\mathbb{Z}_p$. Finally, choose y uniformly at random in $\mathbb{Z}_p$. The public parameters PK are:

$$Y = e(g,g)^y,\ \{T_{j,x} = g^{t_{j,x}}\}_{j \in \mathcal{U}, x \in \Psi_{\mathcal{T}}},\ \{T^*_{j,x} = g^{t^*_{j,x}}\}_{j \in \mathcal{U}^*, x \in \Phi_{\mathcal{T}}}$$

The master key MK is:

$$y,\ \{t_{j,x}\}_{j \in \mathcal{U}, x \in \Psi_{\mathcal{T}}},\ \{t^*_{j,x}\}_{j \in \mathcal{U}^*, x \in \Phi_{\mathcal{T}}}$$

Key Generation (γ, MK) Consider a user A with an attribute set γ. The key generation algorithm outputs a private key D that enables A to decrypt a message encrypted under a (d, num)-bounded access tree $\mathcal{T}'$ iff $\mathcal{T}'(\gamma) = 1$.

The algorithm proceeds as follows. For each user, choose a random polynomial q_x for each non-leaf node x in the universal access tree $\mathcal{T}$. These polynomials are chosen in the following way in a top-down manner, starting from the root node r. For each x, set the degree c_x of the polynomial q_x to be one less than the threshold value, i.e., $c_x = num - 1$. Now, for the root node r, set $q_r(0) = y$ and choose c_r other points of the polynomial q_r randomly to define it completely. For any other non-leaf node x, set $q_x(0) = q_{\mathrm{parent}(x)}(\mathrm{index}(x))$ and choose c_x other points randomly to completely define q_x. Once the polynomials have been decided, give the following secret values to the user:

$$\{D_{j,x} = g^{\frac{q_x(j)}{t_{j,x}}}\}_{j \in \gamma, x \in \Psi_{\mathcal{T}}},\ \{D^*_{j,x} = g^{\frac{q_x(j)}{t^*_{j,x}}}\}_{j \in \mathcal{U}^*, x \in \Phi_{\mathcal{T}}}$$

The set of above secret values is the decryption key D.

Encryption $(M, \mathrm{PK}, \mathcal{T}')$ To encrypt a message $M \in \mathbb{G}_2$, the encrypter $\mathcal{E}$ first chooses a (d, num)-bounded access tree $\mathcal{T}'$. $\mathcal{E}$ then chooses an assignment of real attributes to the leaf nodes in $\mathcal{T}'$.

Now, to be able to encrypt the message M with the access tree $\mathcal{T}'$, the encrypter first converts it to the normal form (if required). Next, $\mathcal{E}$ defines a map between the nodes in $\mathcal{T}'$ and the universal access tree $\mathcal{T}$ as explained in section 3. Finally, for each non-leaf node x in $\mathcal{T}'$, $\mathcal{E}$ chooses an arbitrary $(num - k_x)$-sized set ω_x of dummy child nodes of $\mathrm{map}(x)$ in $\mathcal{T}$.

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

$$\langle \mathcal{T}', E' = M \cdot Y^s, \{E_{j,x} = T^s_{j,\mathrm{map}(x)}\}_{j \in \mathcal{U}, x \in \Psi_{\mathcal{T}'} : f(j,x)=1}, \{E^*_{j,x} = T^{*s}_{j,\mathrm{map}(x)}\}_{j = \mathrm{att}(z) : z \in \omega_x, x \in \Phi_{\mathcal{T}'}} \rangle$$

[4] Recall the distinction between real attributes and dummy attributes that was introduced in the Overview section.

Decryption (E, D) We specify our decryption procedure as a recursive algorithm. For ease of exposition, we present the simplest form of the decryption algorithm here. The performance of the decryption procedure can potentially be improved by using the techniques explained in [2].

We define a recursive algorithm DecryptNode(E, D, x) that takes as input the ciphertext E, the private key D, and a node x in $\mathcal{T}'$. It outputs a group element of $\mathbb{G}_2$ or $\perp$. First, we consider the case when x is a leaf node. Let $j = \text{att}(x)$ and w be the parent of x. Then, we have:

$$\text{DecryptNode}(E, D, x) = \begin{cases} e(D_{j,\text{map}(w)}, E_{j,w}) = e(g^{\frac{q_{\text{map}(w)}(j)}{t_{j,\text{map}(w)}}}, g^{s \cdot t_{j,\text{map}(w)}}) & \text{if } j \in \gamma \\ \perp \text{ otherwise} \end{cases}$$

which reduces to $e(g, g)^{s \cdot q_{\text{map}(w)}(j)}$ when $j \in \gamma$. We now consider the recursive case when x is a non-leaf node in $\mathcal{T}'$. The algorithm proceeds as follows: For all nodes z that are children of x, it calls DecryptNode(E, D, z) and stores the output as F_z. Additionally, for each dummy node $z \in \omega_x$ (where ω_x is a select set of dummy nodes of map(x) in $\mathcal{T}$ chosen by the encrypter), it invokes a function DecryptDummy(E, D, z) that is defined below, and stores the output as F_z. Let j be the dummy attribute associated with z. Then, we have:

$$\text{DecryptDummy}(E, D, z) = e(D^*_{j,\text{map}(x)}, E^*_{j,x}) = e(g^{\frac{q_{\text{map}(x)}(j)}{t^*_{j,\text{map}(x)}}}, g^{s \cdot t^*_{j,\text{map}(x)}}),$$

which reduces to $e(g, g)^{s \cdot q_{\text{map}(x)}(j)}$. Let Ω_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. Further, let S_x be the union of the sets Ω_x and ω_x. Thus we have that $|S_x| = num$. Let $\hat{g} = e(g, g)$. If no k_x-sized set Ω_x exists, then the node x was not satisfied and the function returns $\perp$. Otherwise, we compute:

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}, \qquad \text{where } \substack{i=\text{att}(z) \text{ if } z \text{ is a leaf node} \\ i=\text{index}(\text{map}(z)) \text{ otherwise} \\ S'_x=\{i:z\in S_x\}} \\ &= \prod_{z \in \Omega_x} F_z^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} F_z^{\Delta_{i,S'_x}(0)} \\ &= \begin{cases} \prod_{z \in \Omega_x} (\hat{g}^{s \cdot q_{\text{map}(x)}(i)})^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} (\hat{g}^{s \cdot q_{\text{map}(x)}(i)})^{\Delta_{i,S'_x}(0)} & \text{if } x \in \Psi_{\mathcal{T}'} \\ \prod_{z \in \Omega_x} (\hat{g}^{s \cdot q_{\text{map}(z)}(0)})^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} (\hat{g}t^{s \cdot q_{\text{map}(x)}(i)})^{\Delta_{i,S'_x}(0)} & \text{else} \end{cases} \\ &= \begin{cases} \prod_{z \in S_x} \hat{g}^{s \cdot q_{\text{map}(x)}(i) \cdot \Delta_{i,S'_x}(0)} & \text{if } x \in \Psi_{\mathcal{T}'} \\ \prod_{z \in \Omega_x} (\hat{g}^{s \cdot q_{\text{map}(\text{parent}(z))}(\text{index}(\text{map}(z)))})^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} (\hat{g}^{s \cdot q_{\text{map}(x)}(i)})^{\Delta_{i,S'_x}(0)} & \text{else} \end{cases} \\ &= \prod_{z \in S_x} \hat{g}^{s \cdot q_{\text{map}(x)}(i) \cdot \Delta_{i,S'_x}(0)} \\ &= \hat{g}^{s \cdot q_{\text{map}(x)}(0)} = e(g, g)^{s \cdot q_{\text{map}(x)}(0)} \qquad \text{(using polynomial interpolation)} \end{aligned}$$

and return the result.

Now that we have defined DecryptNode, the decryption algorithm simply invokes it on the root r' of $\mathcal{T}'$. We observe that DecryptNode$(E, D, r') = e(g, g)^{sy}$ iff $\mathcal{T}'(\gamma) = 1$ (note that $F_{r'} = e(g, g)^{s \cdot q_{\text{map}(r')}(0)} = e(g, g)^{s \cdot q_r(0)} = e(g, g)^{sy}$, where r is the root of the universal tree $\mathcal{T}$). Since $E' = M \cdot e(g, g)^{sy}$, the decryption algorithm simply divides out $e(g, g)^{sy}$ and recovers M.

Theorem 1 *If an adversary can break our scheme in the Selective-Tree model, then a simulator can be constructed to play the Decisional BDH game with a non-negligible advantage.*

Proof: See full version [11].

5 Non-Monotonic Access Trees

One limitation of our original construction is that it does not support *negative* constraints in a ciphertext's access formula. With some minor modifications to our small universe construction, we can allow an encrypter to use non-monotonic ciphertext policies. Below we highlight the necessary modifications in the small universe case.

We introduce explicit attributes that indicate the *negative* of attributes in the system. A user will be assigned a negative attribute for each attribute *not* present in his attribute set. In this manner, each user will have $|\mathcal{U}|$ number of attributes. It is known that by applying DeMorgan's law, we can transform a non-monotonic access tree $\mathcal{T}''$ into $\mathcal{T}'$ so that $\mathcal{T}'$ represents the same access scheme as $\mathcal{T}''$, but has **NOTs** only at the leaves, where the attributes are. Further, we can replace an attribute j with its corresponding negative attribute $\bar{j}$ if the above transformation results in a **NOT** gate at the leaf to which j is associated. Now consider a (d, num)-bounded non-monotonic access tree $\mathcal{T}''$ chosen by an encrypter. Using the above mechanism, the encrypter first transforms it to $\mathcal{T}'$ such that the interior gates of $\mathcal{T}'$ consist only of positive threshold gates, while both positive and negative attributes may be associated with the leaf nodes. Then, the encryption and decryption procedures follow as in the original construction.

Supporting any Access Formula of Bounded Polynomial Size. It is known that any access formula can be represented by a non-monotonic NC^1 circuit [12]. It is intuitive to see that any circuit of logarithmic depth can be converted to a tree with logarithmic depth. To this end, we note that our modified construction for non-monotonic access trees can support any access formula of bounded polynomial size.

6 Discussion and Extensions

We discuss various extensions to our scheme.

Large Universe Case. In our previous construction, the size of public parameters corresponding to the real attributes grows linearly with the size of the universe of real attributes. Combining the tricks presented in section 4 with those in [2], we construct another scheme that allows us to use arbitrary strings as attributes in the system, yet the public parameters corresponding to the real attributes grow only linearly in a parameter n which we fix as the maximum number of leaf child nodes of a node in an access tree we can encrypt under. Details will be given in the full version [11].

Non-Monotonic Access Policies in the Large Universe Case. We note that the solution for supporting non-monotonic access policies in the small universe construction is inapplicable in the large universe case. This is because the attributes in the system may not be fixed at the time of key generation. However, we can leverage the techniques from [13] in order to support non-monotonic access policies in the large universe case. Details will be give in the full version [11].

Delegation of Private Keys. Similar to the system of Goyal et al. [2], our constructions come with the added capability of delegation of private keys. Details will be given in the full version [11].

References

1. Sahai, A., Waters, B.: Fuzzy Identity Based Encryption. In: Advances in Cryptology – Eurocrypt. Volume 3494 of LNCS., Springer (2005) 457–473
2. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute Based Encryption for Fine-Grained Access Conrol of Encrypted Data. In: ACM conference on Computer and Communications Security (ACM CCS). (2006)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (2007) 321–334
4. Cheung, L., Newport, C.: Provably Secure Ciphertext Policy ABE. In: ACM conference on Computer and Communications Security (ACM CCS). (2007)
5. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Advances in Cryptology – Eurocrypt. Volume 2656 of LNCS., Springer (2003)
6. Canetti, R., Halevi, S., Katz, J.: Chosen Ciphertext Security from Identity Based Encryption. In: Advances in Cryptology – Eurocrypt. Volume 3027 of LNCS., Springer (2004) 207–222
7. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In: Advances in Cryptology – Eurocrypt. Volume 3027 of LNCS., Springer (2004) 223–238
8. Shamir, A.: Identity Based Cryptosystems and Signature Schemes. In: Advances in Cryptology – CRYPTO. Volume 196 of LNCS., Springer (1984) 37–53
9. Boneh, D., Franklin, M.: Identity Based Encryption from the Weil Pairing. In: Advances in Cryptology – CRYPTO. Volume 2139 of LNCS., Springer (2001) 213–229
10. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: IMA Int. Conf. (2001) 360–363
11. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded Ciphertext Policy Attribute Based Encryption Avaialble at: http://eprint.iacr.org/2008/.
12. Brent, R.P.: The parallel evaluation of general arithmetic expressions. Journal of ACM **21** (1974) 201–206
13. Ostrovsky, R., Sahai, A., Waters, B.: Attribute Based Encryption with Non-Monotonic Access Structures. In: ACM conference on Computer and Communications Security (ACM CCS). (2007)