Efficient and Provable Secure Ciphertext-Policy Attribute-Based Encryption Schemes

Luan Ibraimi[1], Qiang Tang[1], Pieter Hartel[1], Willem Jonker[1,2]

[1] Faculty of EWI, University of Twente, the Netherlands
[2] Philips Research, the Netherlands

Abstract. In a ciphertext-policy attribute-based encryption (CP-ABE) scheme, the data is encrypted under an access policy defined by a user who encrypts the data and a user secret key is associated with a set of attributes which identify the user. A user can decrypt the ciphertext if and only if his attributes satisfy the access policy. In CP-ABE, since the user enforces the access policy at the encryption phase, the policy moves with the encrypted data. This is important for data storage servers where data confidentiality must be preserved even if the server is compromised or un-trusted. In this paper, we provide an efficient CP-ABE scheme which can express any access policy represented by a formula involving $\wedge$ and $\vee$ boolean operators. The scheme is secure under Decision Bilinear Diffie-Hellman assumption (DBDH). Furthermore, we extend the expressivity of the scheme by including *of* (threshold) operator in addition to $\wedge$ and $\vee$ operators. We provide a comparison with existing CP-ABE schemes and show that our schemes are more efficient. Especially, the computational work done by the decryptor is reduced.

1 Introduction

Public-key cryptography is an asymmetric scheme that uses a pair of keys for encryption - a private key which is kept secret and a public key which is widely distributed. If Alice wants to send a confidential message to Bob, she can encrypt the message with the public key of Bob and only Bob can decrypt the message using his private key. In a Public-Key Infrastructure (PKI), a public key must be obtained from, or at least be certified by the Trusted Third Party (TTP) of the PKI. In Identity-Based Encryption (IBE) any string (for example `bob@acm.org`) can be used to generate a public key without involvement of the TTP [6,18,10]. IBE thus creates a degree of flexibility that a PKI cannot offer. However, if Alice does not know the identity of her party, but instead she only knows certain attributes of the recipient, then neither a PKI nor IBE will work. For example imagine that Alice wishes to communicate with her former classmates, but she does not know their email address.

The solution to this problem is provided by Attribute-Based Encryption (ABE), which identifies a user with a set of attributes [16]. In their seminal paper Sahai and Waters use biometric measurements as attributes in the following way. A secret key based on a set of attributes ω, can decrypt a ciphertext encrypted with a public key based on a set of attributes ω', only if the sets ω and ω' overlap sufficiently as determined by a threshold value t. In the sequel we will refer to the Sahai and Waters scheme as the SW scheme. A more general policy to decide which attributes are required to decrypt a message is provided by an access tree. For example the access tree $\tau = class1978 \wedge mycollege \vee myteacher$ states that all students with the attribute $class1978$ who studied at $mycollege$ as well as the teacher possessing the attribute $myteacher$ would satisfy the policy.

There are two variants of ABE: Key-Policy based ABE (KP-ABE) [12] and Ciphertext-Policy based ABE (CP-ABE)[3,9]. In KP-ABE, the ciphertext is associated with a set of attributes and the secret key is associated with the access tree. The encryptor does not define the privacy policy and has no control over who has access to the data except by defining the set of descriptive attributes necessary to decrypt the ciphertext. The trusted authority who generates user's secret key defines the combination of attributes for which the secret key can be used. In CP-ABE, the idea is reversed: now the ciphertext is associated with the access tree and the encrypting party determines the policy under which the data can be decrypted, while the secret key is associated with a set of attributes.

Related Work. Pirreti et al. [15] give a construction and implementation of a modified SW scheme, which, compared to the original SW scheme, drastically reduces computational overhead in the Encryption and the Key Generation phase. The Pirreti et al. [15] scheme is secure in the random oracle model, which, is weaker than the security of the SW scheme since the security of the cryptosystem depends on the security of the hash function and there is no real implementation of a true random oracle. Goyal et al. [12] introduce the idea of KP-ABE where the secret key associated with the access tree controls which ciphertext a user is able to decrypt. In the Goyal et al. scheme, when a user makes a secret key request, the trusted authority determines which combination of attributes must appear in the ciphertext for the user to decrypt. The Goyal et al. scheme is an extension of SW scheme where instead of using the Shamir [17] secret sharing technique in the private key, the trusted authority uses a more generalized form of secret sharing to enforce a monotonic access tree. Chase [8] constructs a multi-authority ABE scheme, which allows multiple independent authorities to monitor attributes and distribute secret keys. A related work to KP-ABE is a predicate encryption paradigm or searching on encrypted data [13,1,5,7]. Predicate encryption has the advantages of providing ciphertext anonymity by hiding the access structures, however, the system is less expressive compared to schemes which leave the access structures in the clear. Smart [19] gives an access control data scheme which encrypts data to an arbitrary collection of identities using a variant of the Boneh-Franklin IBE scheme. However, the problem of resisting attack from colluding users is not addressed.

The first CP-ABE scheme proposed by Bethencourt et al.[3] uses threshold secret sharing to enforce the policy in the encryption phase. We will henceforth refer to this scheme as the BSW scheme. The main drawback of the BSW scheme is that it requires polynomial interpolation to reconstruct the secret, thus many expensive pairing and exponentiation operations in the decryption phase are required. The scheme is secure in the generic group model, which model provides evidence to the hardness of the problem, without giving security proof which reduces the problem of breaking the scheme to a well-studied complexity-theoretic problem. The CP-ABE proposed by Cheung and Newport [9] does not use threshold secret sharing but uses random elements to enforce the policy in the encryption phase. We will henceforth refer to this scheme as the CN scheme. The CN scheme has two drawbacks. Firstly, the CN scheme is not sufficiently expressive since it supports only policies with logical conjunction. Secondly, the size of the ciphertext and secret key increases linearly with the total number of attributes in the system. This makes the CN scheme inefficient. Goyal at el. [11] give a "bounded" CP-ABE construction. The disadvantage of their scheme is that the depth of the access trees d under which messages can be encrypted is defined in the Setup phase. Thus, the user who wants to encrypt a message is restricted to use only an access tree which has the depth $d' \leq d$.

Contribution. In this paper we focus on the efficiency of the CP-ABE scheme having a security proof based on a standard complexity-theoretic assumption. Previous CP-ABE systems could either support only $\wedge$ nodes in the access structures [9], or have a security proof only in the generic group model [3] or specify the depth of the access tree in the Setup phase [11]. We propose two schemes which are (1) more efficient, and (2) at least as expressive as the BSW and CN schemes. Our contribution is twofold:

- We present a new technique for realizing Ciphertext-Policy ABE systems which does not use threshold secret sharing. We first show how to achieve this construction which we will refer to it as a basic CP-ABE scheme. In the scheme the encryptor defines the privacy policy through an access tree which is n-ary tree represented by $\wedge$ *and/or* $\vee$ nodes. Realizing a scheme which does not use threshold secret sharing is important for resource constraint devices since calculating polynomial interpolations to construct the secret is computationally expensive.
- Next, we extend the basic CP-ABE scheme and provide a second CP-ABE scheme which uses Shamir's (k, t) threshold secret sharing technique [17]. The access tree is an n-ary tree represented by $\wedge$, $\vee$ and *of* (threshold) nodes. We compare the efficiency of our scheme with the BSW scheme and show that our scheme requires less computations in the key generation, encryption and decryption phase.

Organization. The sequel of the paper is organized as follows. In section 2 we review concepts of the access structure, secret sharing, CP-ABE and bilinear pairing. In section 3 we introduce our new basic CP-ABE scheme which is secure under DBDH assumption. In section 4 we provide an extensions of the basic CP-ABE scheme. We extend the expressivity of the scheme by including *of* operators in addition to $\wedge$ and $\vee$ operators. In section 5 we discuss how to update the user attribute set and the access policy, and how to achieve anonymous CP-ABE scheme. The last section concludes the paper.

2 Background

First, we give definition for an access structure. Next, we give background information on secret sharing, specifically for unanimous consent control by modular addition scheme and the Shamir's secret sharing scheme. Then we give the formal security definition of the ciphertext-policy attribute-based encryption (CP-ABE) scheme. Finally, we give background information on bilinear maps and the Decision Bilinear Diffie-Hellman (DBDH) assumption.

2.1 Access Structures

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

In CP-ABE, instead of parties we use attributes and the access structure $\mathbb{A}$ will contain the set of authorized attributes.

2.2 Secret Sharing Schemes

In designing our CP-ABE schemes we will make use of two different secret-sharing schemes: unanimous consent control by modular addition scheme and the Shamir secret sharing scheme.

Unanimous Consent Control by Modular Addition Scheme

In a unanimous consent control by modular addition scheme [14], there is a dealer who splits a secret s into t shares in a such way that all shares are required to reconstruct the secret s. To share the secret s, $0 \leq s \leq p-1$ for some integer p, the dealer generates a $t-1$ random numbers s_i such that $1 \leq s_i \leq p-1$, $1 \leq i \leq t-1$ and $s_t = s - \sum_{i=1}^{t-1} s_i \; mod \, p$. The secret s is recovered by: $s = \sum_{i=1}^{t} s_i$. Shares s_i, $1 \leq i \leq t$ are distributed to parties P_i, $1 \leq i \leq t$. For each party P_i, $1 \leq i \leq t$, the shares are random numbers between 0 and $p-1$, thus no party has any information about s except the dealer.

Shamir's Secret Sharing Scheme

In Shamir's secret sharing technique [17] a secret s is divided into n shares in a such way that any subset of t shares, where $t \leq n$, can together reconstruct the secret; no subset smaller than t can reconstruct the secret. The technique is based on polynomial interpolation where a polynomial $y = f(x)$ of degree $t-1$ is uniquely defined by t points (x_i, y_i). The details of the scheme are as follows:

1. Setup. The dealer D wants to distribute the secret $s > 0$ among t users.
   1)D chooses a prime $p > max(s, n)$, and defines $a_0 = s$.
   2)D selects $t - 1$ random coefficients $a_1, ..., a_{t-1}, 0 \leq a_j \leq p - 1$, and defines the random polynomial over $\mathbb{Z}_p$, $f(x) = \Sigma_{j=0}^{t-1} a_j x^j$.
   3)D computes $s_i = f(i)\ mod\ p$, and sends securely the share s_i to user p_i together with the public index i.
2. Pooling of shares. Any group of t or more users pool their distinct shares $(x, y) = (i, s_i)$ allowing computation of the coefficients a_j of $f(x)$ by Lagrange interpolation, $f(x) = \Sigma_{i=0}^{t-1} l_j(x)$ where $l_j(x) = \Pi_{1 \leq j \leq t, j \neq i} \frac{x - x_j}{x_i - x_j}$. The secret is $f(0) = a_0 = s$.

2.3 Ciphertext-Policy ABE

CP-ABE schemes consist of four algorithms: [3]:

- $\mathsf{Setup}(k)$. The setup algorithm takes as input a security parameter k and outputs the public parameters pk and a master key mk.
- $\mathsf{Keygen}(\omega, mk)$. The algorithm takes as input the master key mk and a set of attributes ω. The algorithm outputs a secret key sk_ω associated with ω.
- $\mathsf{Encrypt}(m, \tau, pk)$. The encryption algorithm takes as input the public key pk, a message m, and an access tree τ representing an access structure. The algorithm will return the ciphertext c_τ such that only users who have the secret key generated from the attributes that satisfy the access tree will be able to decrypt the message.
- $\mathsf{Decrypt}(c_\tau, sk_\omega)$. The decryption algorithm takes as input a ciphertext c_τ, a secret key sk_ω associated with ω, and it outputs a message m or an error symbol $\perp$.

Using Attributes for Encryption. We assume that the Trusted Authority (TA) is responsible for publishing the attribute set Ω. For instance, in a healthcare domain, $TA_{Healthcare}$ may be responsible for defining the attribute set $\Omega_{Healthcare}$, which may contain attributes such as: doctor, nurse, HIV patient etc, and in a university domain $TA_{University}$ may be responsible for defining the attribute set $\Omega_{University}$, which may contain attributes such as: job position, age, research interest etc. We assume that the process of obtaining a secret key sk_ω associated to a set of attributes ω is straightforward. The user has to go to the TA to apply for sk_ω and "prove" that he/she indeed possess the attribute set ω.

Security Model for CP-ABE. Semantic security under chosen-plaintext attack (CPA) is modelled by an IND-sAtt-CPA game. The game is carried out between a challenger and an adversary $\mathcal{A}$, where the challenger simulates the protocol execution and answers queries from $\mathcal{A}$. Specifically, the game is as follows:

1. Init. The adversary chooses the challenge access tree τ^* and gives it to the challenger.

2. Setup. The challenger runs Setup to generate (pk, mk) and gives the public key pk to the adversary $\mathcal{A}$.
3. Phase1. $\mathcal{A}$ makes a secret key request to the Keygen oracle for any attribute set $\omega = \{a_j | a_j \in \Omega\}$, with the restriction that $a_j \notin \tau^*$. The challenger returns $\mathsf{Keygen}(\omega, mk)$.
4. Challenge. $\mathcal{A}$ sends to the challenger two messages m_0, m_1.The challenger picks a random bit $b \in \{0, 1\}$ and returns $c_b = \mathsf{Encrypt}(m_b, \tau^*, pk)$.
5. Phase2. $\mathcal{A}$ can continue querying Keygen with the same restriction as in Phase1.
6. Guess. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

Definition 2. *A CP-ABE scheme is said to be secure against an adaptive chosen-plaintext attack (CPA) if any polynomial-time adversary has only a negligible advantage in the IND-sAtt-CPA game, where the advantage is defined to be* $\epsilon = |\Pr[b' = b] - \frac{1}{2}|$.

Note: The above game between the challenger and $\mathcal{A}$ can be easily extended to handle chosen-ciphertext attacks by allowing decryption queries in Phase1 and Phase2.

Our scheme is proved secure in the selective-attribute (sAtt) model, in which the adversary must provide the challenge access tree he wishes to attack before he receives the public parameters from the challenger. Suppose, that the adversary in the Init phase chooses the challenge access tree $\tau^* = (A \wedge B) \vee C$. In Phase1, the adversary can make secret key requests to Keygen oracle for any attribute set ω with the restriction that attributes $A, B, C \notin \omega$. The selective-attribute (sAtt) model can be considered to be analogous to the selective-ID (sID) model [4] used in identity-based encryption schemes, in which the adversary commits ahead of time the ID^* it will attack, and where the adversary can make secret key requests to Keygen oracle for any ID such that $ID \neq ID^*$.

2.4 Review of Pairing

We briefly review the basis of bilinear pairing. A pairing (or, bilinear map) satisfies the following properties:

1. $\mathbb{G}_0$ and $\mathbb{G}_1$ are two multiplicative groups of prime order p.
2. g is a generator of $\mathbb{G}_0$.
3. $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ is an efficiently-computable bilinear map with the following properties:
   - Bilinear: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p^*$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
   - Non-degenerate: $\hat{e}(g, g) \neq 1$.

$\mathbb{G}_0$ is said to be a bilinear group if the group operation in $\mathbb{G}_0$ can be computed efficiently and if there exists a group $\mathbb{G}_1$ and an efficiently-computable bilinear map $\hat{e}$ as defined above.

Decision Bilinear Diffie-Hellman Assumption

The Decision Bilinear Diffie-Hellman (DBDH) problem is defined as follows. Given $g, g^a, g^b, g^c \in \mathbb{G}_0$ as input, the adversary must distinguish a valid tuple $\hat{a}(g, g)^{abc} \in \mathbb{G}_1$ from the random element $Z \in \mathbb{G}_1$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the Decision Bilinear Diffie-Hellman (DBDH) problem in $\mathbb{G}_0$ if:

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, Z) = 0]| \geq \epsilon.$$

Here the probability is over the random choice of $a, b, c \in \mathbb{Z}_p^*$, the random choice of $Z \in \mathbb{G}_1$, and the random bits of $\mathcal{A}$ (the adversary is a nondeterministic algorithm).

Definition 3. *We say that the (t, ϵ)-DBDH assumption holds if no t-time algorithm has advantage at least ϵ in solving the DBDH problem in $\mathbb{G}_0$.*

3 Basic Construction

In this section, first, we give a description of the structure of the access policy used in our basic construction, and later we present the construction of the encryption scheme.

Policy Representation

In our scheme the access tree is a n-ary tree, in which leaves are attributes and inner nodes are $\wedge$ and $\vee$ boolean operators. Intuitively, the access tree is a policy which specifies which combination of attributes can decrypt the ciphertext. Consider the following example where a patient wants to specify access restrictions on his medical data. The patient can enforce the access policy in the encryption phase. Each member from the medical staff who has enough attributes should be able to decrypt the encrypted message. For instance, a patient wants to allow his data to be seen by Doctor A who works at Department A or by Doctor B who works at Department B. Using boolean operators the patient defines the following access policy: $\tau_{Data} = (Doc.A \wedge Dep.A) \vee (Doc.B \wedge Dep.B)$.

To decrypt the ciphertext which is encrypted according to the τ_{Data} access tree, the decryptor must possess a secret key, which is associated with the attribute set which satisfies τ_{Data}. To decide whether an access tree is satisfied we interpret each attribute as a logical variable. Possession of the secret key for the corresponding attribute makes the logical variable true. If the decryptor does not possess the attribute, the variable is false. For the policy above there are several different sets of attributes that can satisfy the access tree, such as: the secret key associating with the attribute set $\{Doc.A, Dep.A\}$, the secret key associating with the attribute set $\{Doc.B, Dep.B\}$, or the secret key associating with all attributes defined in the access tree.

Basic Scheme

We now present our version of the four CP-ABE algorithms:

1. Setup(k) : On input of the security parameter k, this algorithm generates the following.
   (a) Generate a bilinear group $\mathbb{G}_0$ of prime order p with a generator g and bilinear map $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$
   (b) Generate the attribute set $\Omega = \{a_1, a_2, \ldots a_n\}$, for some integer n, and random elements $\alpha, t_1, t_2 \ldots t_n \in \mathbb{Z}_p^*$.
   Let $y = \hat{e}(g, g)^\alpha$ and $T_j = g^{t_j}$ $(1 \leq j \leq n)$. The public key is $pk = (\hat{e}, g, y, T_j \, (1 \leq j \leq n))$ and the master secret key is $mk = (\alpha, t_j \, (1 \leq j \leq n))$.
2. Keygen(ω, mk) : The algorithm performs as follows.
   (a) Select a random value $r \in \mathbb{Z}_p^*$ and compute $d_0 = g^{\alpha - r}$.
   (b) For each attribute a_j in ω, compute $d_j = g^{rt_j^{-1}}$.
   (c) Return the secret key $sk_\omega = (d_0, \forall a_j \in \omega : d_j)$
3. Encrypt(m, τ, pk) : To encrypt a message $m \in \mathbb{G}_1$ the algorithm proceeds as follows:
   (a) First level encryption: Select a random element $s \in \mathbb{Z}_p^*$ and compute $c_0 = g^s$ and
   $$c_1 = m \cdot y^s = m \cdot \hat{e}(g, g)^{\alpha s}$$
   (b) Second level encryption: Set the value of the root node of τ to be s, mark all child nodes as un-assigned, and mark the root node assigned. Recursively, for each un-assigned non-leaf node, do the following:
      - If the symbol is $\wedge$ and its child nodes are marked un-assigned, we use a unanimous consent control by modular addition scheme to assign a value to each child node. To do that, for each child node except the last one, assign a random value s_i where $1 \leq s_i \leq p - 1$, and to the last child node assign the value $s_t = s - \sum_{i=1}^{t-1} s_i \mod p$. Mark this node assigned.
      - If the symbol is $\vee$, set the values of each child node to be s. Mark this node assigned.

   Values of the leaves of τ are used to produce ciphertext components.
   (c) For each leaf attribute $a_{j,i} \in \tau$, compute $c_{j,i} = T_j^{s_i}$ where i denotes the index of the attribute in the access tree. The index values are uniquely assigned to leave nodes in an ordering manner for a given access structure.
   (d) Return the ciphertext $c_\tau = (\tau, c_0, c_1, \forall a_{j,i} \in \tau : c_{j,i})$.

Figure 1 is an example of assigning secret shares s_i to the access tree.

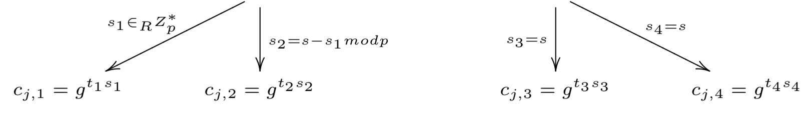

Fig 1. Assigning secret shares to each leaf node in the access tree $\tau = (T_1 \wedge T_2) \vee (T_3 \vee T_4)$

4. $\mathsf{Decrypt}(c_\tau, sk_\omega)$: If ω does not satisfy τ, return $\perp$, otherwise the algorithm chooses the smallest set $\omega' \subseteq \omega$ (we assume that this can be computed efficiently by the decryptor) that satisfies τ and performs as follows:
   (a) For every attribute $a_j \in \omega'$, compute

$$\begin{aligned} \prod_{a_j \in \omega'} \hat{e}(c_{j,i}, d_j) &= \prod_{a_j \in \omega'} \hat{e}(T_j^{s_i}, g^{rt_j^{-1}}) \\ &= \prod_{a_j \in \omega'} \hat{e}(g^{t_j s_i}, g^{rt_j^{-1}}) \\ &= \hat{e}(g, g)^{rs} \end{aligned}$$

   (b) Compute

$$\begin{aligned} \hat{e}(c_0, d_0) \cdot \hat{e}(g, g)^{rs} &= \hat{e}(g^s, g^{\alpha - r}) \cdot \hat{e}(g, g)^{rs} \\ &= \hat{e}(g^s, g^\alpha) \end{aligned}$$

   (c) Return m', where

$$\begin{aligned} m' &= \frac{c_1}{\hat{e}(g^s, g^\alpha)} \\ &= \frac{m \cdot \hat{e}(g, g)^{\alpha s}}{\hat{e}(g^s, g^\alpha)} \\ &= m \end{aligned}$$

We briefly discus security properties of our scheme. A full security proof is given in Appendix A.

Collusion Resistent. The most important property that every CP-ABE scheme must have is to prevent collusion. This means that different users can not combine their secret keys and decrypt a ciphertext that the colluding users should not have access to. To prevent collusion, the KeyGen algorithm of our scheme generates a different random value r for each user, keys generated for different users can not be combined since they are randomized. To decrypt the message the attacker must know how to recover $\hat{e}(g, g)^{\alpha s}$. To do that the attacker must first recover $\hat{e}(g, g)^{rs}$, which would require the attacker to have the secret key blinded with the same random value r.

3.1 Efficiency Analysis

The number of calculations in the Encryption algorithm depends on the number of attributes in the access tree τ. Encryption requires $|\tau| + 1$ exponentiations in $\mathbb{G}_0$ and one exponentiation in $\mathbb{G}_1$. The number of calculations in the KeyGen algorithm depends on the number of attributes in the set ω that the user has.

	Our Scheme			The CN Scheme		
	Exp.($\mathbb{G}_0$)	Exp.($\mathbb{G}_1$)	Pairing	Exp.($\mathbb{G}_0$)	Exp.($\mathbb{G}_1$)	Pairing
Encrypt	$\lvert\tau\rvert+1$	1	/	$\lvert\Omega\rvert+1$	1	/
Keygen	$\lvert\omega\rvert+1$	/	/	$\lvert\Omega\rvert+1$	/	/
Decrypt	/	/	$\lvert\omega'\rvert+1$	/	/	$\lvert\Omega\rvert+1$
	Ω is the set of all attributes defined in the Setup phase τ is the access tree ω is the set of attributes the user has, $\omega \subseteq \Omega$ ω' the set of attributes satisfying the access tree, $\omega' \subseteq \omega$					

Table 1. Comparison of our basic CP-ABE scheme with CN scheme

KeyGen requires $|\omega| + 1$ exponentiations in $\mathbb{G}_0$. The number of calculations in the Decryption algorithm depends on the number of attributes in the attribute set ω' . Decryption requires $|\omega'| + 1$ pairing operations. Decryption also requires $|\omega'|$ multiplications but no exponentiations in $\mathbb{G}_1$.

In Table 1 we compare our CP-ABE scheme with the CN scheme. We count the number of calculations in the Encryption, Key Generation, and Decryption phases. Compared to the CN scheme, our scheme requires fewer computations in the Encryption, Key Generation and Decryption phase.

4 Extension of the Expressivity

In the basic scheme the access tree is a n-ary tree represented by $\wedge$ and $\vee$ nodes. This allows the user who performs encryption to express any privacy policy using boolean formulas. Ideally, we would like to have an n-ary access tree which supports *of* (threshold) operators, similar to the BSW scheme. The essential idea is to allow the encryptor to define the minimum number of attributes from a given list of attributes that the decryptor has to posses in order to decrypt the message. For instance, to decrypt the ciphertext encrypted under the policy $\tau =2$ *of (class1978, mycollege, myteacher)*, the decryptor must have at least two out of three attributes.

We can extend the basic CP-ABE scheme to support *of* nodes as follows:

1. Setup is same as in basic CP-ABE scheme.
2. KeyGen is same as in basic CP-ABE scheme.
3. Encrypt(m, τ, pk) : To encrypt a message $m \in \mathbb{G}_1$ the algorithm proceeds as follows:
   (a) First level encryption: Select a random element $s \in \mathbb{Z}_p^*$ and compute $c_0 = g^s$ and

$$\begin{aligned} c_1 &= m \cdot y^s \\ &= m \cdot \hat{e}(g, g)^{\alpha s} \end{aligned}$$

   (b) Second level encryption: Set the value of the root node to be s, mark all child nodes as un-assigned, and mark the root node assigned. Recursively, for each un-assigned non-leaf node, do the following:

- If the symbol is *of* (threshold operator), and its child nodes are marked un-assigned, the secret s is divided using (t, n) Shamir's secret sharing technique where $t \neq n$, and n is the total number of child nodes and t is the number of child nodes necessary to reconstruct the secret. To each child node a share secret $s_i = f(i)$ is assigned. Mark this node assigned.
- If the symbol is $\wedge$, and its child nodes are marked un-assigned, the secret s is divided using (t, n) Shamir's secret sharing technique where $t = n$, and n is the number of the child nodes. To each child node a share secret $s_i = f(i)$ is assigned. Mark this node assigned.
- If the symbol is $\vee$, and its child nodes are marked un-assigned, the secret s is divided using (t, n) Shamir's secret sharing technique where $t = 1$ and n is the number of the child nodes. To each child node a share secret $s_i = f(i)$ is assigned. Mark this node assigned.

Values of the leaves of τ are used to produce ciphertext components.

(c) For each leaf attribute $a_{j,i} \in \tau$, compute $c_{j,i} = T_j^{s_i}$, where i denote the index of the attribute in the access tree.

(d) Return the ciphertext: $c_\tau = (\tau, c_0, c_1, \forall a_{j,i} \in \tau : c_{j,i})$.

Figure 2 is an example of assigning secret shares s_i to the access tree.

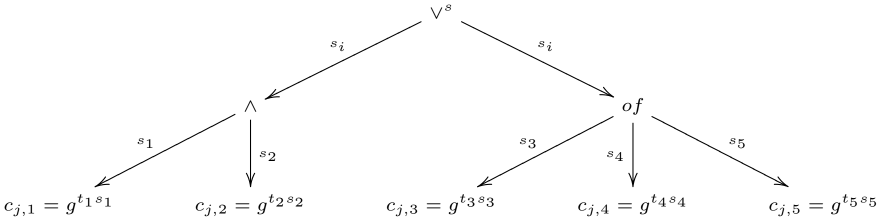

Fig 2. Assigning secret shares to each leaf node in the access tree $\tau = (T_1 \wedge T_2) \vee 2\ of\ (T_3, T_4, T_5)$

4. $\mathsf{Decrypt}(c_\tau, sk_\omega)$: If ω does not satisfy τ, return $\perp$, otherwise the algorithm chooses the smallest set $\omega' \subseteq \omega$ that satisfies τ and performs as follows:

(a) For every attribute $a_j \in \omega'$, compute

$$\begin{aligned}\prod_{a_j \in \omega'} \hat{e}(c_{j,i}, d_j)^{l_i(0)} &= \hat{e}(T_j^{s_i}, g^{rt_j^{-1}})^{l_i(0)} \\ &= \prod_{a_j \in \omega'} \hat{e}(g^{t_j s_i}, g^{rt_j^{-1}})^{l_i(0)} \\ &= \prod_{a_j \in \omega'} \hat{e}(g, g)^{rs_i l_i(0)} \\ &= \hat{e}(g, g)^{rs}\end{aligned}$$

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

	Our Scheme			The BSW Scheme		
	Exp.($\mathbb{G}$)	Exp.($\mathbb{G}_1$)	Pairing	Exp.($\mathbb{G}$)	Exp.($\mathbb{G}_1$)	Pairing
Encrypt	$\lvert\tau\rvert+1$	1	/	$2\lvert\tau\rvert+1$	1	/
KeyGen	$\lvert\omega\rvert+1$	/	/	$2\lvert\omega\rvert+1$	/	/
Decrypt	/	$\lvert\omega'\rvert$	$\lvert\omega'\rvert+1$	/	$\lvert\omega'\rvert$	$2\lvert\omega'\rvert$
(Note:)	Ω is the set of all attributes defined in the Setup phase τ is the access tree ω is the set of attributes the user has, $\omega \subseteq \Omega$ ω' the set of attributes satisfying the access tree, $\omega' \subseteq \omega$					

Table 2. Comparison of our extended CP-ABE scheme with BSW scheme

(b) Compute

$$\begin{aligned}\hat{e}(c_0, d_0) \cdot \hat{e}(g, g)^{rs} &= \hat{e}(g^s, g^{\alpha - r}) \cdot \hat{e}(g, g)^{rs} \\ &= \hat{e}(g^s, g^{\alpha})\end{aligned}$$

(c) Return m', where

$$\begin{aligned}m' &= \frac{c_1}{\hat{e}(g^s, g^{\alpha})} \\ &= \frac{m \cdot \hat{e}(g, g)^{\alpha s}}{\hat{e}(g^s, g^{\alpha})} \\ &= m\end{aligned}$$

A full security proof is presented in Appendix B. In Table 2 we give a comparison of the efficiency of our extended CP-ABE scheme with BSW scheme. Compared to the BSW scheme, our scheme requires fewer computations in the Encryption, Key Generation and Decryption phase.

5 Discussion

Updating the Attribute Set. In CP-ABE granting or revoking an attribute from the user is a challenging task. Revocation is difficult since there is no way to prevent the user from not using the issued attribute secret key, since the attribute is not connected solely with one user. Pirretti et al. [15] propose to use time framed attributes where each attribute would be valid for a specific time frame. However this would require the trusted authority to update the list of attributes regularly.

Granting additional attributes is less difficult than revoking. There are two options for granting attributes. One option is to keep a list of users and the corresponding random values r generated during Key Generation phase. The trusted authority needs the random variable r to update the attribute set for each user, since for each attribute a_j the KeyGen algorithm computes $d_j = g^{rt_j^{-1}}$. Another option, which would not require maintaining a list of users, is to do everything from the beginning, issue again secret keys for each attribute for the updated user set.

Updating the Access Policy. In a CP-ABE scheme the message encryptor may update his access policy without entirely decrypting the ciphertext. Since in the scheme, the user defines the access policy using an access tree, the change of the policy means the change of the access tree τ. Suppose the user wants to update his privacy policy by updating the access tree from $\tau = (T_1 \wedge T_2) \vee (T_3 \vee T_4)$ represented in Fig 1. to a different access tree $\tau' = (T_1 \wedge T_2) \vee (T_3 \wedge T_4)$ represented in Figure 3.

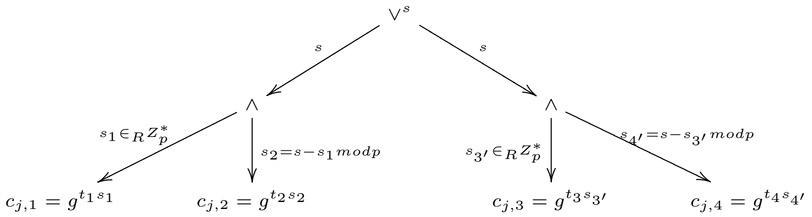

Fig 3. Assigning secret shares to each leaf node in the access tree $\tau' = (T_1 \wedge T_2) \vee (T_3 \wedge T_4)$

Recall from section 3. To encrypt a message $m \in \mathbb{G}_1$ under $\tau = (T_1 \wedge T_2) \vee (T_3 \vee T_4)$ the Encrypt algorithm selects a random element $s \in \mathbb{Z}_p^*$ in the first level encryption, then it sets: $c_1 = m \cdot y^s = m \cdot \hat{e}(g, g)^{\alpha s}$ and $c_0 = g^s$. The second level encryption is based on τ. The algorithm sets: $\forall a_{j,i} \in \tau : c_{j,1} = T_1^{s_1}$, $c_{j,2} = T_2^{s_2}$, $c_{j,3} = T_3^{s_3}$, $c_{j,4} = T_4^{s_4}$. The final ciphertext is $c_\tau = (\tau, c_0, c_1, \forall a_{j,i} \in \tau : c_{j,1}, c_{j,2}, c_{j,3}, c_{j,4})$.

To update the privacy from τ to τ', there is no reason to modify the first level encryption, since the second level encryption enforces the policy. Therefore, to update the policy over the encrypted data the user has to update only the second level encryption. The new ciphertext will be different only at $c_{j,3}$ and $c_{j,4}$ therefore the updates are made only at $c_{j,3}$ and $c_{j,4}$. The new ciphertext elements are: $\forall a_{j,i} \in \tau' : c_{j,1} = T_1^{s_1}$, $c_{j,2} = T_2^{s_2}$, $c_{j,3} = T_3^{s_{3'}}$, $c_{j,4} = T_4^{s_{4'}}$. The new final ciphertext is $c_{\tau'} = (\tau', c_0, c_1, \forall a_{j,i} \in \tau' : c_{j,1}, c_{j,2}, c_{j,3}, c_{j,4})$.

Updating the privacy policy without totally decrypting the ciphertext, requires the user to know the random value s used in the Encryption phase. This is a trade-off for the encryptor since it has to keep a list of the random variables used during each encryption.

Achieving Anonymous CP-ABE. In CP-ABE, when a message sender encrypts a message, along with the encrypted message, the sender specifies in clear text the policy τ used to encrypt the message. However, the policy may reveal some sensitive information about the message being encrypted. Suppose, Alice encrypts the message under the policy $\tau = Psychiatrist \wedge Neurologists$. From the policy, an adversary may conclude that Alice has a mental disorder. The ideal solution to prevent the adversary or unintended decrypter to learn some information about the message being encrypted is to remove τ from the cipehertext.

Therefore, to decrypt the ciphertext, Bob must try all possible sets ω' until τ is satisfied. Although this can be computationally inefficient, it ensures that if Bob does not posses the right attributes he learns almost nothing about the policy τ which controlls access to the message. Moreover, he doesn't learn what attribute he would need to obtain from trusted authority in order to decrypt the message.

6 Conclusion and Future Work

We have shown how to improve the efficiency of a CP-ABE scheme. Firstly, we present a new technique to construct a CP-ABE scheme which does not use threshold secret sharing. The encryptor specifies the policy in the encryption phase using an n-ary tree which consists from $\vee$ and $\wedge$ nodes. Our main result is less computation in the encryption, key generation and decryption phase. Then, we show a modified version of the first scheme which is more expressive compared to the basic scheme but uses threshold secret sharing. In the extended version the policy is expressed as an n-ary tree access tree which consists of $\vee$, $\wedge$ and *of* nodes. For future work, it would be interesting to construct an efficient anonymous collusion-resistent CP-ABE which would allow the decryptor to decrypt the ciphertext without incorporating the access tree τ in the ciphertext.

Acknowledgments

We thank Asim Muhammad and Peter Van Liesdonk for their suggestions and comments.

References

1. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *Journal of Cryptology*, 21(3):350–391, 2008.
2. A. Beimel. Secure Schemes for Secret Sharing and Key Distribution. *Phd Thesis,Israel Institute of Technology, Technion, Haifa, Israel*, 1996.
3. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 321–334, 2007.
4. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. *LNCS*, 3027:223–238, 2004.
5. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. *LNCS*, pages 506–522, 2004.
6. D. Bonehl and M. Franklin. Identity-Based Encryption from the Weil Pairing. *Advances in Cryptology-Crypto 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, 2001.
7. X. Boyen and B. Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). *LNCS*, 4117:290, 2006.

8. M. Chase. Multi-authority Attribute Based Encryption. *LNCS*, 4392:515, 2007.
9. L. Cheung and C. Newport. Provably secure ciphertext policy ABE. *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 456–465, 2007.
10. C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. *LNCS*, pages 360–363, 2001.
11. V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded Ciphertext Policy Attribute Based Encryption. *LNCS*, 5126:579–591, 2008.
12. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 89–98, 2006.
13. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *LNCS*, 4965:146, 2008.
14. A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
15. M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 99–112, 2006.
16. A. Sahai and B. Waters. Fuzzy identity-based encryption. *Advances in Cryptology–Eurocrypt*, 3494:457–473, 2005.
17. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
18. A. Shamir. Identity-based cryptosystems and signature schemes. *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 47–53, 1985.
19. N. Smart. Access control using pairing based cryptography. *Proceedings of The Cryptographer's Track at RSA Conference (CT-RSA 2003)*, pages 111–121, 2003.

A Security Proof for the basic CP-ABE scheme

We prove the following theorem.

Theorem 1. *Suppose the DBDH assumption holds. Then no polynomial adversary can break the basic scheme of section 3 with a challenge access tree* τ^*

Suppose the adversary $\mathcal{A}$ can win the IND-sAtt-CPA game with a non-negligible advantage ϵ. We show how to use the adversary $\mathcal{A}$ to build a simulator β that is able to solve the DBDH assumption with advantage $\epsilon/2$. Before the game starts, the challenger sets the groups $\mathbb{G}_0$ and $\mathbb{G}_1$, a generator g of group $\mathbb{G}_0$, a mapping function $\hat{e}$, and selects at random: $a, b, c, \theta \in \mathbb{Z}_p^*$. The challenger flips a coin μ and sets $Z_\mu = \hat{e}(g,g)^{abc}$ if $\mu = 0$ and $Z_\mu = \hat{e}(g,g)^\theta$ otherwise. The challenger gives to the simulator β the DBDH challenge: $(g, A, B, C, Z_\mu) = (g, g^a, g^b, g^c, Z_\mu)$. β will act as $\mathcal{A}$'s challenger in the IND-sAtt-CPA game as follows:

Init. The adversary chooses the challenge access tree τ^* and gives it to the simulator.

Setup. The simulator selects at random $x' \in \mathbb{Z}_p$ and implicitly sets $\alpha = ab + x'$ by letting $\hat{e}(g,g)^\alpha = \hat{e}(g,g)^{ab}\hat{e}(g,g)^{x'}$. For all $a_j \in \mathit{\Omega}$, $(1 \leq j \leq n)$, it chooses a random $k_j \in \mathbb{Z}_p$ and sets $T_j = B^{1/k_j}$ (thus $t_j = b/k_j$) if $a_j \notin \tau*$ or $T_j = g^{k_j}$ if $a_j \in \tau*$ (thus $t_j = k_j$). The simulator, β, sends the public parameters to $\mathcal{A}$.

Phase1. $\mathcal{A}$ makes secret key requests for any set of attributes $\omega_j = \{a_j | a_j \in \mathit{\Omega}\}$ with the restriction that $a_j \notin \tau*$. On each request the challenger chooses a random variable $r'^{(j)} \in \mathbb{Z}_p^*$ and sets $d_0 = g^{x' - r'^j b}$. Thus, implicitly it sets $r = ab + r'b$ since:

$$\begin{aligned} d_0 &= g^{x' - r'^j b} \\ &= g^{\alpha - ab - r'^j b} \\ &= g^{\alpha - (ab + r'^j b)} \end{aligned}$$

For each $a_j \in \omega_j$ the simulator has to construct secret key components of the form $d_j = g^{r t_j^{-1}}$. Since the simulator implicitly sets $r = ab + r'b$ and $t_j = b/k_j$ for each $a_j \notin \tau*$, the valid form of the secret key component would be $d_j = g^{(ab+r'b)k_j/b}$. For each $a_j \in \omega_j$ the simulator sets $d_j = A^{k_j} g^{k_j r'}$. This is a valid secret key component and can be computed by the simulator since:

$$\begin{aligned} d_j &= g^{(ab+r'b)k_j/b} \\ &= g^{ak_j i} g^{k_j r'} \\ &= A^{k_j} g^{k_j r'} \end{aligned}$$

The simulator β sends to $\mathcal{A} : sk_{\omega_j} = (d_0, \forall a_j \in \omega_j : d_j)$.

Challenge. $\mathcal{A}$ submits two messages $m_0, m_1 \in \mathbb{G}_1$. The simulator flips a fair binary coin b, and returns the encryption of m_b. The encryption of m_b is done as follows:

1. First level encryption:

$$c_0 = g^c$$

   and

$$\begin{aligned} c_1 &= m_b \hat{e}(g,g)^{\alpha c} \\ &= m_b \hat{e}(g,g)^{(ab+x')c} \\ &= m_b \hat{e}(g,g)^{abc} \hat{e}(g^c, g^{x'}) \\ &= m_b Z_\mu \hat{e}(g^c, g^{x'}) \end{aligned}$$

2. Second level encryption: Set the value of the root node of $\tau*$ to be g^c, mark all child nodes as un-assigned, and mark the root node assigned. Recursively, for each un-assigned non-leaf node do the following.
   - If the symbol is $\wedge$ and its child nodes are marked un-assigned, for each child except the last one the simulator chooses h_i where $1 \leq h_i \leq p-1$, and assigns g^{h_i} to them, and to the last child it assigns $g^{h_t} = g^c / \sum_{i=1}^{t-1} g^{h_i}$. Mark this node assigned.
   - If the symbol is $\vee$, set the values of each child node to be g^c. Mark this node assigned.
3. For every $a_{j,i} \in \tau^*$, compute $c_{j,i} = g^{h_i k_j}$.

The ciphertext $c_{\tau^*} = (\tau^*, c_0, c_1, \forall a_{j,i} \in \tau^* : c_{j,i})$ is sent to $\mathcal{A}$ as a "challenge ciphertext".

Phase2. $\mathcal{A}$ can continue secret key requests with the same restriction as in Phase1.

Guess. $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

If $b' = b$, the simulator β will guess that $\mu = 0$ and $Z_\mu = e(g,g)^{abc}$, otherwise will guess that $\mu = 1$ and $Z_\mu = e(g,g)^\theta$. When $Z_\mu = e(g,g)^{abc}$ the simulator β gives the perfect simulation and $c_{\tau*}$ is a valid ciphertext. Therefore the advantage of the adversary is:

$$\Pr[b' = b | Z_\mu = e(g,g)^{abc}] = \frac{1}{2} + \epsilon$$

If $\mu = 1$ then $Z_\mu = e(g,g)^\theta$ and $c_{\tau*}$ is random ciphertext for the adversary, and the adversary does not gain information about m_b. Hence we have:

$$\Pr[b' \neq b | Z_\mu = e(g,g)^\theta] = \frac{1}{2}$$

Since the simulator β guesses $\mu' = 0$ when $b' = b$ and $\mu' = 1$ when $b' \neq b$, the overall advantage of β to solve DBDH assumption is:

$$\frac{1}{2} \Pr[\mu' = \mu | \mu = 0] + \frac{1}{2} \Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{\epsilon}{2}$$

B Security Proof for the extended CP-ABE scheme

The security proof from appendix A can be applied to the extended CP-ABE scheme from section 4 as well. The game played between the simulator β and the adversary $\mathcal{A}$ is the same as in section A with a small difference in the generation of the challenge ciphertext components where the simulator uses a different approach to assign shares to leave nodes in the second level encryption. This is necessary because the access tree contains an additional operator, *of* (threshold) operator, compared to the basic scheme in section 3. The simulation of the second level encryption will be as follows:

The simulator β sets the value of the root node of $\tau *$ to be g^c, it marks all child nodes as un-assigned, and marks the root node assigned. Recursively, for each un-assigned non-leaf child node do the following:

- If the symbol is *of* (threshold operator), and its child nodes are marked un-assigned, the simulator chooses $g^{f(i)}$ for each child node, where $f(i)$ is a polynomial of degree $t-1$, t is the number of child nodes to reconstruct the secret, i is the index (order) of the attributes in τ^* and $f(0) = c$. Mark this node assigned.
- If the symbol is $\wedge$, and its child nodes are marked un-assigned, the simulator chooses $g^{f(i)}$ for each child node, where $f(i)$ is a polynomial of degree $n-1$, n is the total number of the child nodes, i is the index (order) of the attributes in τ^* and $f(0) = c$. Mark this node assigned.
- If the symbol is $\vee$, and its child nodes are marked un-assigned, the simulator chooses $g^{f(i)}$ for each child node, where $f(i)$ is a polynomial of degree 0, i is the index (order) of the attributes in τ^* and $f(0) = c$. Mark this node assigned.

For each leaf attribute $a_{j,i} \in \tau$, compute $c_{j,i} = g^{f(i)k_j}$.
As in section A, the advantage of the simulator β to solve DBDH assumption is: $\epsilon/2$.