Provably Secure Ciphertext Policy ABE

Ling Cheung[*]
MIT CSAIL, Cambridge, MA, USA
lcheung@theory.csail.mit.edu

Calvin Newport[†]
MIT CSAIL, Cambridge, MA, USA
cnewport@theory.csail.mit.edu

ABSTRACT

In ciphertext policy attribute-based encryption (CP-ABE), every secret key is associated with a set of attributes, and every ciphertext is associated with an access structure on attributes. Decryption is enabled if and only if the user's attribute set satisfies the ciphertext access structure. This provides fine-grained access control on shared data in many practical settings, e.g., secure database and IP multicast.

In this paper, we study CP-ABE schemes in which access structures are AND gates on positive and negative attributes. Our basic scheme is proven to be chosen plaintext (CPA) secure under the decisional bilinear Diffie-Hellman (DBDH) assumption. We then apply the Canetti-Halevi-Katz technique to obtain a chosen ciphertext (CCA) secure extension using one-time signatures. The security proof is a reduction to the DBDH assumption and the strong existential unforgeability of the signature primitive.

In addition, we introduce hierarchical attributes to optimize our basic scheme—reducing both ciphertext size and encryption/decryption time while maintaining CPA security. We conclude with a discussion of practical applications of CP-ABE.

Categories and Subject Descriptors

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

General Terms

Security, Theory

Keywords

Access Control, Attribute-Based Encryption, Public Key Cryptography

[*]Cheung was supported by NSF Award #CCR-0326277.

[†]Newport was supported by NSF Award #CCR-0121277 and USAF, AFRL Award #FA9550-04-1-0121.

1. INTRODUCTION

In traditional public key crypto systems, the communication model is one-to-one, in the sense that any message encrypted using a particular public key can be decrypted only with the corresponding secret key. The same holds for identity-based encryption (IBE) [14], where user public keys can be arbitrary bit strings such as email addresses. In practice, however, many applications of encryption are natural examples of one-to-many communication. For instance, documents in a shared database are accessed by multiple users, and cable television programs are viewed by multiple subscribers.

One-to-many communication can be secured using one-to-one public key cryptosystems in a straightforward manner. For example, a sender may encrypt data with a symmetric encryption key, and distribute this data key to every intended receiver using public key encryption. This general scheme is simple to implement, but inefficient in the number of encryption operations and in the size of ciphertexts (both linear in the total number of intended receivers). A better solution is broadcast encryption [8, 11, 2], where a sender specifies a set of receivers (or revoked users) during encryption. Any intended receiver can decrypt using his secret key, while revoked users cannot—even if they collude.

Although broadcast encryption is efficient, it requires that receivers are represented individually. A sender must maintain a list of prospective receivers, as well as authorization information associated with each receiver. In order to derive a receiver set for a particular message, the sender queries his receiver database with some appropriate policy information.

In some application scenarios, it is desirable to be able to encrypt without exact knowledge of the set of intended receivers. For example, in a secure database of an intelligence agency, one may specify that a certain document can be accessed only by agents in the domestic surveillance program. In that situation, it is much more natural to encrypt to the single attribute "domestic spy", instead of an enumerative list of all domestic spies. (It is conceivable that the sender is not even authorized to obtain such a list.)

Attribute-Based Encryption. Attribute-based encryption (ABE) offers this desired ability to encrypt without exact knowledge of the receiver set. It enforce access policies, defined on attributes, within the encryption procedure. This idea was first introduced by Sahai and Waters (SW) as an application of their fuzzy IBE scheme [13], where both ciphertexts and secret keys are associated with sets of attributes. Decryption is enabled if and only if the ciphertext and secret key attribute sets overlap by at least a fixed threshold value d.

Two variants of ABE were subsequently proposed. In the key policy variant (KP-ABE) of Goyal, Pandey, Sahai and Waters (GPSW) [10], every ciphertext is associated with a set of attributes, and every user secret key is associated with a threshold access structure on attributes.[1] Decryption is enabled if and only if the ciphertext attribute set satisfies the access structure on the user secret key. In the ciphertext policy variant (CP-ABE) of Bethencourt, Sahai and Waters (BSW) [1], the situation is reversed: attributes are associated with user secret keys and access structures with ciphertexts.

To date, all existing ABE schemes involve some form of threshold secret sharing construction. In [13, 10], shares of a system master secret are embedded into user secret keys, while in [1] shares of the randomness in an encryption are embedded into ciphertext components. In this paper, we break from this tradition and consider AND-gates on positive and negative attributes as our access structures. In this simpler setting, we obtain efficient schemes that are provably secure under standard complexity assumptions. Since AND gates are sufficient in many application scenarios, our approach retains significant potential.

Our Contributions. We first present a CP-ABE scheme that is chosen plaintext (CPA) secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. Access structures in this scheme are AND-gates on positive and negative attributes. We then apply the Canetti-Halevi-Katz technique to obtain a chosen ciphertext (CCA) secure extension, using one-time signatures. The security proof is a reduction to the DBDH assumption and the strong existential unforgeability of the signature primitive. Since strongly existentially unforgeable signatures can be constructed under the standard computational Diffie-Hellman (CDH) assumption [3], the security of our CCA scheme reduces to DBDH and CDH. To the best of our knowledge, this is the first formal CCA security proof for CP-ABE.

We observe that attributes can be arranged into logical hierarchies, which in turn can be used to improve the efficiency of our basic scheme. Essentially, a hierarchy allows us to use fewer group elements to represent all attributes in the system, thereby reducing the ciphertext size, the number of exponentiations in encryption and the number of pairings in decryption. This optimized scheme is proven to be CPA secure.

Finally, we note that policies in the form of disjunctive normal form formulas can be enforced by encrypting the same message to every AND gate in the formula. We discuss some subtleties in the security of this proposal, and leave the formal proof as important future work.

Related Work. As mentioned above, the concept of ABE was proposed in [13] and later extended in [10, 1]. Both the fuzzy IBE scheme of [13] and the KP-ABE scheme of [10] are proven secure under the DBDH assumption. For the CP-ABE scheme of [1], there appears to be inherent difficulties in reducing security to a well-known complexity assumption. Mostly likely, this is due to the way in which secret key components are "tied together" to avoid collusion attacks. Indeed, in this paper we use a different technique to bind secret key components and we are able to prove security based on DBDH. The trade-off inherent in our technique is that ciphertext size and encryption/decryption time grow linearly with n, the total number of attributes in the system. For small AND gates, the optimization of Section 5 brings the factor n down to $\log n$. In contrast, the ciphertext size and encryption/decryption time of the BSW scheme are linear in the size of the access tree, independent of n.

The authors of [13, 1] suggest that chosen ciphertext security can be achieved using the Fujisaki-Okamoto transformation [9] or the Canetti-Halevi-Katz (CHK) technique [4]. A more concrete outline is given in [10], using a large universe construction[2] and delegation of user secret keys. In this paper, we implement CCA security without a large universe construction (and hence without the use of hash functions).

In [5], Chase answers an open problem posed in [13] and presents a scheme in which (disjoint sets of) attributes are assigned by multiple authorities. Our technique for binding together secret key components is similar to Chase's technique for "distributing" the system master secret among multiple attribute authorities. Chase also outlines a CP-ABE system in which access structures are monotone (i.e., no negation) conjunctive normal form formulas. That approach, however, is inefficient, as each authority corresponds to one clause in the conjunction and the same attribute appearing in different clauses must be duplicated at every relevant authority.

On the practical side, Pirreti et al. introduced a secure information management architecture based on ABE primitives [12]. The original SW scheme is implemented and optimized. It is also shown that complex policies can be implemented efficiently using constructions that are secure in the random oracle model.

Overview. Section 2 defines the CP-ABE primitive and the CPA security game. Section 3 presents the basic construction for AND gates and the CPA security proof. Section 4 introduces the CCA security game and our CCA secure scheme. A more efficient version of the basic scheme is presented in Section 5, and a heuristic is given in Section 6 for disjunctive normal form policies. Finally, two applications scenarios (selective data sharing and group key management) are described in Section 7, and concluding remarks follow in Section 8.

2. CIPHERTEXT POLICY ABE

Intuitively, an access structure on attributes is a rule W that returns either 0 or 1 given a set S of attributes. We say that S *satisfies* W (written $S \models W$) if and only if W answers 1 on S. As mentioned in Section 1, access structures may be Boolean expressions, threshold trees, etc.

A *ciphertext policy attribute-based encryption (CP-ABE)* system consists of four fundamental algorithms: *Setup*, *Encrypt*, *KeyGen* and *Decrypt*.

[1]In [10], an access structure is a tree in which every internal node is a threshold gate and every leaf is labeled by an attribute.

[2]In a large universe construction, any bit string can be used as an attribute, provided each encryption involves no more than a fixed number of attributes. Additional hashing and exponentiation operations are required.

Setup. Setup takes as input a security parameter κ and returns a public key PK and a master key MK. The public key is used for encryption. The master key, held by the central authority, is used to generate user secret keys.

KeyGen. KeyGen takes as input the master key MK and a set $S \subseteq \mathcal{N}$ of attributes. It returns a secret key SK associated with S.

Encrypt. Encrypt takes as input the public key PK, a message M and an access structure W. It returns a ciphertext CT such that a secret key generated from attribute set S can be used decrypt CT if and only if $S \models W$.

Decrypt. Decrypt takes as input a ciphertext CT and a secret key SK. It returns the message M if S satisfies W, where S is the attribute set used to generate SK.

2.1 CPA Security Game for CP-ABE

A CP-ABE scheme is said to be secure against *chosen plaintext attacks (CPA)* if no probabilistic polynomial-time adversaries have non-negligible advantage in this game.

Init The adversary chooses the challenge access structure W and gives it to the challenger.

Setup The challenger runs the Setup algorithm and gives the adversary PK.

Phase 1 The adversary submits S for a KeyGen query. Provided $S \not\models W$, the challenger answers with a secret key SK for S. This can be repeated adaptively.

Challenge The adversary submits two messages M_0 and M_1 of equal length. The challenger chooses $\mu \in \{0, 1\}$ at random and encrypts M_μ to W. The resulting ciphertext CT is given to the adversary.

Phase 2 Same as Phase 1.

Guess The adversary outputs a guess μ' of μ.

Notice, collusion resistance follows from the fact the adversary may make multiple secret key queries both before and after selecting challenge plaintexts. We also point out that our CPA security game is weaker than that of [1], because the adversary must submit a challenge access structure *before* the setup phase. This is essential in our security proofs (cf. Sections 3 and 4), because the simulator uses information from the challenge access structure to set up public key elements. This weaker form is sometimes called *selective ID security.* See the full version of this work for a discussion of the stronger variant called *adaptive ID security*, and its implications for our scheme [7].

3. BASIC CONSTRUCTION

For notational simplicity, let the set of attributes be $\mathcal{N} := \{1, \ldots, n\}$ for some natural number n. We refer to attributes i and their negations $\neg i$ as *literals.* In this section, we consider access structures that consist of a single AND gate whose inputs are literals. This is denoted $\bigwedge_{i \in I} \underline{i}$, where $I \subseteq \mathcal{N}$ and every $\underline{i}$ is a literal (i.e., i or $\neg i$).

Setup. This algorithm selects:

- a bilinear group G of prime order p, with bilinear map $e : G \times G \to G_1$,
- random elements $y, t_1, \ldots, t_{3n}$ in $\mathbb{Z}_p$ and a random generator g of G.

	1	2	3	...	n
Positive	T_1	T_2	T_3		T_n
Negative	T_{n+1}	T_{n+2}	T_{n+3}		T_{2n}
Don't Care	T_{2n+1}	T_{2n+2}	T_{2n+3}		T_{3n}

Figure 1: Public Key Components

Let $Y := e(g, g)^y$ and $T_k := g^{t_k}$ for each $k \in \{1, \ldots, 3n\}$. The public key is $PK := \langle e, g, Y, T_1, \ldots, T_{3n} \rangle$. The master secret key is $MK := \langle y, t_1, \ldots, t_{3n} \rangle$.

Intuitively, the public key elements T_i, T_{n+i} and T_{2n+i} correspond to the three types of occurrences of i: positive, negative and *don't care.* This is illustrated in Figure 1. Because of the technique we use to randomize secret key components, we must provide a *don't care* element for each attribute i not appearing in the AND gate. This should become clear after we introduce KeyGen and Decrypt.

Encrypt. Given a message $M \in G_1$ and an AND gate $W = \bigwedge_{i \in I} \underline{i}$, the Encrypt algorithm first selects a random $s \in \mathbb{Z}_p$ and sets $\tilde{C} := M \cdot Y^s$ and $\hat{C} := g^s$. For each $i \in I$, let C_i be T_i^s if $\underline{i} = i$ and T_{n+i}^s if $\underline{i} = \neg i$. For each $i \in \mathcal{N} \setminus I$, let C_i be T_{2n+i}^s. The ciphertext is $CT := \langle W, \tilde{C}, \hat{C}, \{C_i | i \in \mathcal{N}\} \rangle$.

In total, Encrypt performs $n + 1$ exponentiations in G, one exponentiation in G_1 and one multiplication in G_1. The ciphertext contains $n + 1$ elements of G, one element of G_1 and the description of W.

KeyGen. Let S denote the input attribute set. Every $i \notin S$ is implictly considered a negative attribute. First, KeyGen selects random r_i from $\mathbb{Z}_p$ for every $i \in \mathcal{N}$ and sets $r := \sum_{i=1}^{n} r_i$. Let $\hat{D}$ be g^{y-r}. For each $i \in \mathcal{N}$, let D_i be $g^{\frac{r_i}{t_i}}$ if $i \in S$; otherwise, let it be $g^{\frac{r_i}{t_{n+i}}}$. Finally, let F_i be $g^{\frac{r_i}{t_{2n+i}}}$ for every $i \in \mathcal{N}$. The secret key is defined as $SK := \langle \hat{D}, \{\langle D_i, F_i \rangle | i \in \mathcal{N}\} \rangle$.

Note that we use the equation $r = \sum_{i=1}^{n} r_i$ to bind together the D_i elements. (Similarly for the F_i elements.) This is a key difference between our scheme and the BSW scheme, and it is crucial in our reduction proof. The F_i elements are provided because every r_i must be recovered in order to decrypt. If i is a *don't care* for a particular encryption operation (i.e., i does not occur in the AND gate W), then F_i will be used for decryption, instead of D_i.

Decrypt. Suppose the input ciphertext is of the form $CT = \langle W, \tilde{C}, \hat{C}, \{C_i | i \in \mathcal{N}\} \rangle$, where $W = \bigwedge_{i \in I} \underline{i}$. Also, let S denote the attribute set used to generate the input secret key $SK = \langle \hat{D}, \{\langle D_i, F_i \rangle | i \in \mathcal{N}\} \rangle$. For each $i \in I$, Decrypt computes the pairing $e(C_i, D_i)$. If $\underline{i} = i$ and $i \in S$, then

$$e(C_i, D_i) = e(g^{t_i \cdot s}, g^{\frac{r_i}{t_i}}) = e(g, g)^{r_i \cdot s}.$$

Similarly, if $\underline{i} = \neg i$ and $i \notin S$, then

$$e(C_i, D_i) = e(g^{t_{n+i} \cdot s}, g^{\frac{r_i}{t_{n+i}}}) = e(g, g)^{r_i \cdot s}.$$

For each $i \notin I$, Decrypt computes the pairing

$$e(C_i, F_i) = e(g^{t_{2n+i} \cdot s}, g^{\frac{r_i}{t_{2n+i}}}) = e(g, g)^{r_i \cdot s}.$$

Decrypt finishes as follows: $M = \frac{\tilde{C}}{Y^s} = \frac{\tilde{C}}{e(g,g)^{y \cdot s}}$, where

$$e(g,g)^{y \cdot s} = e(g^s, g^{y-r}) \cdot e(g,g)^{r \cdot s} = e(\hat{C}, \hat{D}) \cdot \prod_{i=1}^{n} e(g,g)^{r_i \cdot s}.$$

In total, Decrypt performs $n+1$ pairings and n multiplications in G_1. There are no exponentiations. If $|I|$ is small, most of the work is done on *don't care* elements. In Section 5, we show that attributes can be arranged in a tree-based hierarchy, thereby reducing the overhead associated with *don't care* elements. The linear factor n in both ciphertext size and encryption/decryption time drops to $\log(n)$.

3.1 Discussions

We remark that the fuzzy IBE scheme of [13] can also be used to encrypt to AND gates with negation, by adding a new attribute "$\neg i$" for every original attribute i. Since the threshold value d is a system-wide parameter in the SW scheme, default attributes must be added in order to encrypt to AND gates with fewer than d inputs. These default attributes play a similar role as our *don't care* elements. In comparison, our scheme treats negation and *don't care* in a more streamlined fashion. Moreover, our scheme does not involve any secret sharing construction, therefore no exponentiations are necessary in our decryption. In contrast, SW decryption requires d exponentiations in order to perform polynomial interpolation. Finally, the optimization of Section 5 is based on the equation $r = \sum_{i=1}^{n} r_i$ for secret keys. It is not clear how to optimize the SW scheme in a similar way, because SW secret keys are constructed from threshold secret sharing.

3.2 CPA Security Proof

We now reduce CPA security of our scheme to the decisional bilinear Diffie-Hellman (DBDH) assumption. This implies that breaking CPA security under our scheme is at least as difficult as breaking the DBDH assumption. The latter is generally considered a hard problem.

DEFINITION 3.1 (DBDH). *Let $e : G \times G \to G_1$ be an efficiently computable bilinear map, where G has prime order p. The* decisional bilinear Diffie-Hellman *assumption is said to hold in G if no probabilistic polynomial-time adversary is able to distinguish the tuples $\langle g, g^a, g^b, g^c, e(g,g)^{abc} \rangle$ and $\langle g, g^a, g^b, g^c, e(g,g)^z \rangle$ with non-negligible advantage, where $a, b, c, z \in \mathbb{Z}_p$ and generator $g \in G$ are chosen independently and uniformly at random.*

THEOREM 3.2. *If a probabilistic polynomial-time adversary wins the CP-ABE game with non-negligible advantage, then we can construct a simulator that distinguishes a DBDH tuple from a random tuple with non-negligible advantage.*

PROOF. Suppose adversary *Adv* can win the CP-ABE game with advantage ϵ. We construct a simulator *Sim* that can distinguish a DBDH tuple from a random tuple with advantage $\frac{\epsilon}{2}$. Let $e : G \times G \to G_1$ be an efficiently computable bilinear map, where G has prime order p. First the DBDH challenger selects at random: $a, b, c, z \in \mathbb{Z}_p$, $\nu \in \{0,1\}$ and generator $g \in G$. It defines Z to be $e(g,g)^{abc}$ if $\nu = 0$ and $e(g,g)^z$ otherwise. The challenger then gives the simulator $\langle g, A, B, C, Z \rangle = \langle g, g^a, g^b, g^c, Z \rangle$. The simulator *Sim* now plays the role of challenger in the CP-ABE game.

	$i \in I$		
	$\underline{i} = i$	$\underline{i} = \neg i$	$i \notin I$
T_i	g^{α_i}	B^{α_i}	B^{α_i}
T_{n+i}	B^{β_i}	g^{β_i}	B^{β_i}
T_{2n+i}	B^{γ_i}	B^{γ_i}	g^{γ_i}

Figure 2: Public Key in CPA Simulation

Init. During the init phase, *Sim* receives the challenge gate $W = \bigwedge_{i \in I} \underline{i}$ from adversary *Adv*.

Setup. To provide a public key to *Adv*, *Sim* sets Y to be $e(A,B) = e(g,g)^{ab}$. For each $i \in \mathcal{N}$, *Sim* chooses random $\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_p$ and constructs T_i, T_{n+i}, and T_{2n+i} as in Figure 2.

Phase 1. *Adv* submits a set $S \subseteq \mathcal{N}$ in a secret key query, where $S \not\models W$. There must exist $j \in I$ such that: either $j \in S$ and $\underline{j} = \neg j$, or $j \notin S$ and $\underline{j} = j$. *Sim* chooses such j. Without loss of generality, assume that $j \notin S$ and $\underline{j} = j$.

For every $i \in \mathcal{N}$, *Sim* chooses r'_i at random from $\mathbb{Z}_p$. It then sets $r_j := ab + r'_j \cdot b$ and, for every $i \neq j$, it sets $r_i := r'_i \cdot b$. Finally, it sets $r := \sum_{i=1}^{n} r_i = ab + \sum_{i=1}^{n} r'_i \cdot b$. The $\hat{D}$ component of the secret key can be computed as $\prod_{i=1}^{n} \frac{1}{B^{r'_i}} = g^{-\sum_{i=1}^{n} r'_i \cdot b} = g^{ab-r}$.

Recall that $j \in I \setminus S$ and $\underline{j} = j$, therefore the D_j component can be computed as:

$$D_j := A^{\frac{1}{\beta_j}} \cdot g^{\frac{r'_j}{\beta_j}} = g^{\frac{ab + r'_j \cdot b}{b \cdot \beta_j}} = g^{\frac{r_j}{b \cdot \beta_j}}.$$

For $i \neq j$, we have a few cases.

(1) $i \in S$.

(a) $i \in I \wedge \underline{i} = i$. $D_i := B^{\frac{r'_i}{\alpha_i}} = g^{\frac{r_i}{\alpha_i}}$.

(b) $(i \in I \wedge \underline{i} = \neg i) \vee i \notin I$. $D_i := g^{\frac{r'_i}{\alpha_i}} = g^{\frac{r_i}{b \cdot \alpha_i}}$.

(2) $i \notin S$.

(a) $(i \in I \wedge \underline{i} = i) \vee i \notin I$. $D_i := g^{\frac{r'_i}{\beta_i}} = g^{\frac{r_i}{b \cdot \beta_i}}$.

(b) $i \in I \wedge \underline{i} = \neg i$. $D_i := B^{\frac{r'_i}{\beta_i}} = g^{\frac{r_i}{\beta_i}}$.

The F_i components are computed similarly. First,

$$F_j := A^{\frac{1}{\gamma_j}} \cdot g^{\frac{r'_j}{\gamma_j}} = g^{\frac{ab + r'_j \cdot b}{b \cdot \gamma_j}} = g^{\frac{r_j}{b \cdot \gamma_j}}.$$

For $i \neq j$, we have two cases.

(a) $i \in I$. $F_i := g^{\frac{r'_i}{\gamma_i}} = g^{\frac{r_i}{b \cdot \gamma_i}}$.

(b) $i \notin I$. $F_i := B^{\frac{r'_i}{\gamma_i}} = g^{\frac{r_i}{\gamma_i}}$.

Challenge. *Adv* submits two messages M_0 and M_1 of equal length. *Sim* chooses $\mu \in \{0,1\}$ at random and sets $\tilde{C} := M_\mu \cdot Z$. *Sim* gives *Adv* the following ciphertext CT.

$$W, \tilde{C}, C, \{C^{\alpha_i} | i \in I \wedge \underline{i} = i\},$$
$$\{C^{\beta_i} | i \in I \wedge \underline{i} = \neg i\}, \{C^{\gamma_i} | i \notin I\}$$

Phase 2. Same as Phase 1.

Guess. *Adv* produces a guess μ' of μ. If $\mu' = \mu$, *Sim* answers "DBDH" in the DBDH game. Otherwise, *Sim* answers "random". If $Z = e(g,g)^{abc}$, then CT is a valid ciphertext, in which case the advantage of Adv is ϵ. Hence,

$$\mathbf{P}[Sim \to \text{"DBDH"}|Z = e(g,g)^{abc}]$$
$$= \mathbf{P}[\mu' = \mu|Z = e(g,g)^{abc}] = \frac{1}{2} + \epsilon.$$

If $Z = e(g,g)^z$, then $\tilde{C}$ is completely random from the view of Adv. Therefore $\mu' \neq \mu$ holds with probability exactly $\frac{1}{2}$, regardless of the distribution on μ'. Hence,

$$\mathbf{P}[Sim \to \text{"random"}|Z = e(g,g)^z]$$
$$= \mathbf{P}[\mu' \neq \mu|Z = e(g,g)^z] = \frac{1}{2}.$$

It follows that Sim's advantage in the DBDH game is $\frac{\epsilon}{2}$. $\square$

4. CHOSEN CIPHERTEXT SECURITY

In [4], Canetti, Halevi and Katz gave a generic construction for CCA secure public key encryption, using CPA secure IBE and strongly existentially unforgeable signatures. The main idea is to associate one-time signature keys $\langle K_v, K_s\rangle$ with each encryption operation. The verification key K_v is viewed as an identity in the IBE scheme to which the message M is encrypted. The resulting ciphertext is then signed using the signing key K_s. This signature is sent along with the ciphertext and must be verified before decryption.

As it turns out, the same general technique can be applied to ABE schemes. In [10], Goyal et al. gave a CPA secure KP-ABE scheme and outlined an CCA secure extension, in which the message M is encrypted using an additional attribute corresponding to the bit-string representation of K_v. This extension relies on a large universe construction [13] (where arbitrary bit-string attributes can be added after initial setup) and a delegation mechanism for secret keys.

In this section, we also apply the CHK technique to obtain a CCA secure extension, but we do so without a large universe construction. Instead, we modify our setup algorithm to explicitly handle special "attributes" corresponding to bits in K_v. This incurs an additional overhead that is linear in the length of K_v.

4.1 Strong Existential Unforgeability

A signature scheme consists of three algorithms: SigKeyGen, Sign and Verify. SigKeyGen is a probabilistic algorithm that outputs a signing-verification key pair $\langle K_s, K_v\rangle$. Sign is a probabilistic algorithm that produces a signature σ from K_s and a message M. Finally, Verify is a deterministic algorithm that maps $\langle M, \sigma, K_v\rangle$ to a bit. The signature σ is said to be valid for M and K_v if Verify returns 1.

A signature scheme is said to be *strongly existentially unforgeable (SEU)* under adaptive chosen message attacks if no probabilistic polynomial-time adversary has non-negligible success probability in the following game.

Setup. The challenger runs SigKeyGen to obtain $\langle K_s, K_v\rangle$ and gives the adversary K_v.

Signature Queries. The adversary submits message M. The challenger runs Sign with K_s and responds with signature σ. This may be repeated adaptively.

Output. The adversary outputs a pair $\langle M^*, \sigma^*\rangle$. If Verify returns 1 on $\langle M^*, \sigma^*, K_v\rangle$ and $\langle M^*, \sigma^*\rangle$ is not among the pairs generated during the query phase, then the adversary wins the game.

4.2 CCA Secure CP-ABE Scheme

We now use strongly existentially unforgeable signatures to achieve CCA security. Assume that K_v is a bit string of length m, and we write $K_{v,i}$ for the i-th bit in K_v. Intuitively, we expand the set of attributes to include the m bits of K_v. These new attributes must be handled differently than normal attributes in $\mathcal{N}$, because every user must be able to decrypt regardless of the particular choice of K_v.

Let $\mathcal{M}$ denote $\{1, \ldots, m\}$. The four algorithms of CP-ABE are as follows.

Setup. As before, Setup selects G, G_1, e, g and $y, t_1, \ldots, t_{3n}$, and sets $Y := e(g,g)^y$ and $T_k := g^{t_k}$ for each $k \in \{1, \ldots, 3n\}$. In addition, Setup selects random $u_1, \ldots, u_{2m}$ in $\mathbb{Z}_p$ and defines $U_k := g^{u_k}$ for each $k \in \{1, \ldots, 2m\}$. The public key is $PK := \langle e, g, Y, T_1, \ldots, T_{3n}, U_1, \ldots, U_{2m}\rangle$. The master secret key is $MK := \langle y, t_1, \ldots, t_{3n}, u_1, \ldots, u_{2m}\rangle$.

Encrypt. To encrypt a message $M \in G_1$ to an AND gate $W = \bigwedge_{i \in I} \underline{i}$, a key pair $\langle K_v, K_s\rangle$ is first obtained by running SigKeyGen. Then Encrypt selects a random $s \in \mathbb{Z}_p$ and sets $\langle \tilde{C}, \hat{C}, C_1, \ldots, C_n\rangle$ as before. For each $i \in \mathcal{M}$, let E_i be U_i^s if $K_{v,i} = 0$ and U_{m+i}^s otherwise.

Now Encrypt runs Sign with K_s to obtain a signature σ on $\langle W, \tilde{C}, \hat{C}, \{C_i | i \in \mathcal{N}\}, \{E_i | i \in \mathcal{M}\}\rangle$. The ciphertext is $CT := \langle W, \tilde{C}, \hat{C}, \{C_i | i \in \mathcal{N}\}, \{E_i | i \in \mathcal{M}\}, \sigma, K_v\rangle$.

KeyGen. As before, KeyGen selects random $r_i \in \mathbb{Z}_p$ and define $\langle D_i, F_i\rangle$ for every $i \in \mathcal{N}$. In addition, it selects random w_i for every $i \in \mathcal{M}$. Define $r := \sum_{i=1}^{n} r_i + \sum_{i=1}^{m} w_i$ and let $\hat{D}$ be g^{y-r}. For every $i \in \mathcal{M}$, let G_i^0 be $g^{\frac{w_i}{u_i}}$ and G_i^1 be $g^{\frac{w_i}{u_{m+i}}}$. The secret key is $SK := \langle \hat{D}, \{\langle D_i, F_i\rangle | i \in \mathcal{N}\}, \{\langle G_i^0, G_i^1\rangle | i \in \mathcal{M}\}\rangle$.

Decrypt. Given a ciphertext $\langle W, \tilde{C}, \hat{C}, \{C_i | i \in \mathcal{N}\}, \{E_i | i \in \mathcal{M}\}, \sigma, K_v\rangle$, Decrypt first runs the Verify algorithm on σ, K_v and $\langle W, \tilde{C}, \hat{C}, \{C_i | i \in \mathcal{N}\}, \{E_i | i \in \mathcal{M}\}\rangle$. If σ is valid, it proceeds with decryption; otherwise it returns default $\perp$.

Suppose the secret key is defined over an attribute set S and is of the form $\langle \hat{D}, \{\langle D_i, F_i\rangle | i \in \mathcal{N}\}, \{\langle G_i^0, G_i^1\rangle | i \in \mathcal{M}\}\rangle$. As before, Decrypt computes the pairing $e(C_i, D_i)$ for each $i \in I$ and the pairing $e(C_i, F_i)$ for each $i \in \mathcal{N} \setminus I$. This recovers $e(g,g)^{r_i \cdot s}$ for every $i \in \mathcal{N}$, provided $S \models W$.

For each $i \in \mathcal{M}$, Decrypt computes $e(E_i, G_i^0)$ if $K_{v,i} = 0$:

$$e(E_i, G_i^0) = e(g^{u_i \cdot s}, g^{\frac{w_i}{u_i}}) = e(g,g)^{w_i \cdot s}.$$

Similarly, if $K_{v,i} = 1$, it computes $e(g,g)^{w_i \cdot s}$ as $e(E_i, G_i^1)$. Then M is recovered as before: $M = \frac{\tilde{C}}{Y^s} = \frac{\tilde{C}}{e(\hat{C}, \hat{D}) \cdot e(g,g)^{r \cdot s}}$, where

$$e(g,g)^{r \cdot s} = \prod_{i=1}^{n} e(g,g)^{r_i \cdot s} \cdot \prod_{i=1}^{m} e(g,g)^{w_i \cdot s}.$$

4.3 CCA Security Proof

CCA security for CP-ABE is defined as the statement that all probabilistic polynomial-time adversaries have at most negligible advantage in the following game.

Init and Setup Same as CPA security game.

Phase 1 The adversary makes, adaptively, any combination of secret key and decryption queries.

Secret Key Query The adversary submits a set S of attributes. The challenger returns a secret key SK for S, provided $S \not\models W$.

Decryption Query The adversary submits a ciphertext CT encrypted to W. The adversary loses the game if CT is not a valid ciphertext; otherwise, the challenger returns the plaintext M.

Challenge Same as CPA security game.

Phase 2 Same as Phase 1, except that CT^* may not be submitted for decryption.

Guess The adversary outputs a guess μ' of μ.

We reduce CCA security of our scheme to the SEU assumption on signatures and the DBDH assumption. As with CPA, this implies breaking CCA security is at least as hard as breaking the SEU or DBDH assumptions.

THEOREM 4.1. *Assume the signature scheme is strongly existentially unforgeable. If a probabilistic polynomial-time adversary wins the CCA security game with non-negligible advantage, then we can construct a simulator that distinguishes, with non-negligible advantage, a DBDH tuple from a random tuple.*

PROOF. Suppose adversary *Adv* can win the CCA game with non-negligible advantage ϵ. We construct a simulator *Sim* as follows.

First, *Sim* receives $\langle g, A, B, C, Z\rangle$ from the DBDH challenger and runs SigKeyGen to obtain $\langle K_s^*, K_v^*\rangle$. Then *Sim* plays the role of challenger in the CCA game.

Init. *Sim* receives the challenge gate $W = \bigwedge_{i\in I} \underline{i}$ from *Adv*.

Setup. To provide a public key PK to *Adv*, *Sim* sets $Y = e(g,g)^{ab} = e(A,B)$. For each $i \in \mathcal{N}$, *Sim* chooses random $\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_p$ and defines t_i, t_{n+i}, t_{2n+i} as in the CPA proof. For each $i \in \mathcal{M}$, *Sim* chooses random $\eta_i, \xi_i \in \mathbb{Z}_p$. There are two cases.
(1) $K^*_{v,i} = 0$. Then $u_i := \eta_i$ and $u_{m+i} := b\cdot\xi_i$.
(1) $K^*_{v,i} = 1$. Then $u_i := b\cdot\eta_i$ and $u_{m+i} := \xi_i$.
Recall that every public key component T_i can be computed by raising either g or B to the appropriate exponent. The same holds for every U_i.

Phase 1: Secret Key Query. *Adv* submits $S \subseteq \mathcal{N}$ such that $S \not\models W$. As in the CPA proof, choose a witness j. Without loss of generality, assume that $j \notin S$ and $\underline{j} = j$.

For every $i \in \mathcal{N}$, *Sim* chooses $r'_i \in \mathbb{Z}_p$ at random and define r_i as in the CPA proof. In addition, *Sim* chooses $w'_i \in \mathbb{Z}_p$ at random and sets $w_i := w'_i \cdot b$ for every $i \in \mathcal{M}$. Finally, let $r := \sum_{i=1}^n r_i + \sum_{i=1}^m w_i = ab + \sum_{i=1}^n r'_i\cdot b + \sum_{i=1}^m w'_i\cdot b$.

The $\hat{D}$ component of SK is set to be

$$\prod_{i=1}^n \frac{1}{B^{r'_i}} \cdot \prod_{i=1}^m \frac{1}{B^{w'_i}} = g^{-\sum_{i=1}^n r'_i\cdot b - \sum_{i=1}^m w'_i\cdot b} = g^{ab-r}.$$

For $i \in \mathcal{N}$, the D_i and F_i components are computed exactly as in the CPA proof. For $i \in \mathcal{M}$, we have two cases.

- $K^*_{v,i} = 0$. $G^0_i := B^{\frac{w'_i}{\eta_i}} = g^{\frac{w_i}{\eta_i}}$ and $G^1_i := g^{\frac{w'_i}{\xi_i}} = g^{\frac{w_i}{b\cdot\xi_i}}$.
- $K^*_{v,i} = 1$. $G^0_i := g^{\frac{w'_i}{\eta_i}} = g^{\frac{w_i}{b\cdot\eta_i}}$ and $G^1_i := B^{\frac{w'_i}{\xi_i}} = g^{\frac{w_i}{\xi_i}}$.

Phase 1: Decryption Query. *Adv* submits a ciphertext $\langle T, \tilde{C}, \hat{C}, \{C_i | i \in \mathcal{N}\}, \{E_i | i \in \mathcal{M}\}, \sigma, K_v\rangle$. *Sim* verifies the signature σ using K_v. If σ is invalid, *Sim* aborts the DBDH simulation and we call this an abort event. Otherwise, *Sim* checks if $K_v = K_v^*$. If so, we call it a forge event and *Sim* gives a random answer in the DBDH simulation (either "DBDH" or "random"). If $K_v \neq K_v^*$, then *Sim* fixes $j \in \mathcal{M}$ such that $K_{v,j} \neq K^*_{v,j}$. Without loss of generality, assume $K_{v,j} = 1$ and $K^*_{v,j} = 0$.

Let S be the set of attributes defined as follows: for each $i \in \mathcal{N}$, we place $i \in S$ if and only if $i \in I$ and $\underline{i} = i$. Note that S satisfies W. Now *Sim* produces a *partial* secret key for S, which contains enough components to decrypt CT.

For every $i \in \mathcal{N}$, *Sim* chooses r'_i at random from $\mathbb{Z}_p$ and sets $r_i := r'_i \cdot b$. For every $i \in \mathcal{M}$, *Sim* also chooses w'_i at random from $\mathbb{Z}_p$. For $i \neq j$, set $w_i := w'_i \cdot b$. Set $w_j := ab + w'_j \cdot b$. Finally, let $r := \sum_{i=1}^n r_i + \sum_{i=1}^m w_i = ab + \sum_{i=1}^n r'_i \cdot b + \sum_{i=1}^m w'_i \cdot b$.

Notice, the ab exponent is "hidden" at the j-th bit of the signature verification key (where K_v and K_v^* differ). In contrast, to answer the secret key query, ab is hidden at an attribute that witnesses the mismatch between S and W.

As in the secret key query, $\hat{D}$ is set to be:

$$\prod_{i=1}^n \frac{1}{B^{r'_i}} \cdot \prod_{i=1}^m \frac{1}{B^{w'_i}} = g^{-\sum_{i=1}^n r'_i\cdot b - \sum_{i=1}^m w'_i\cdot b} = g^{ab-r}.$$

For each $i \in \mathcal{N}$, D_i is computed as follows.

- $i \in I \wedge \underline{i} = i$. $D_i := B^{\frac{r'_i}{\alpha_i}} = g^{\frac{r_i}{\alpha_i}}$.
- $i \in I \wedge \underline{i} = \neg i$. $D_i := B^{\frac{r'_i}{\beta_i}} = g^{\frac{r_i}{\beta_i}}$.
- $i \notin I$. $D_i := g^{\frac{r'_i}{\beta_i}} = g^{\frac{r_i}{b\cdot\beta_i}}$.

For the F_i components, we have two cases.

- $i \in I$. $F_i := g^{\frac{r'_i}{\gamma_i}} = g^{\frac{r_i}{b\cdot\gamma_i}}$.
- $i \notin I$. $F_i := B^{\frac{r'_i}{\gamma_i}} = g^{\frac{r_i}{\gamma_i}}$.

For each $i \in \mathcal{M}$ with $i \neq j$, we have two cases.

- $K^*_{v,i} = 0$. $G^0_i := B^{\frac{w'_i}{\eta_i}} = g^{\frac{w_i}{\eta_i}}$ and $G^1_i := g^{\frac{w'_i}{\xi_i}} = g^{\frac{w_i}{b\cdot\xi_i}}$.
- $K^*_{v,i} = 1$. $G^0_i := g^{\frac{w'_i}{\eta_i}} = g^{\frac{w_i}{b\cdot\eta_i}}$ and $G^1_i := B^{\frac{w'_i}{\xi_i}} = g^{\frac{w_i}{\xi_i}}$.

Now consider j. By assumption we have $K^*_{v,j} = 0$, therefore $G^0_j = g^{\frac{w_j}{\eta_j}} = g^{\frac{ab+w'_j\cdot b}{\eta_j}}$. This is the only component in the secret key that *Sim* cannot produce, because it contains the exponent ab. However, since $K_{v,j} = 1$, G^0_j is *not* necessary for the decryption of CT. *Sim* can in fact produce G^1_j as $A^{\frac{1}{\xi_j}} \cdot g^{\frac{w'_j}{\xi_j}} = g^{\frac{ab+w'_j\cdot b}{b\cdot\xi_j}} = g^{\frac{w_j}{b\cdot\xi_j}}$. Finally, *Sim* decrypts using this partial secret key and gives M to *Adv*.

Challenge. *Adv* submits two messages M_0 and M_1. *Sim* chooses $\mu \in \{0,1\}$ at random and sets $\tilde{C} := M_\mu \cdot Z$. Then *Sim* signs the following using K_s: $T, \tilde{C}, C, \{C^{\alpha_i} | i \in I, \underline{i} = i\}$, $\{C^{\beta_i} | i \in I, \underline{i} = \neg i\}, \{C^{\gamma_i} | i \in \mathcal{N} \setminus I\}, \{C^{\eta_i} | i \in \mathcal{M}, K^*_{v,i} = 0\}, \{C^{\xi_i} | i \in \mathcal{M}, K^*_{v,i} = 1\}$. These are given to the adversary, along with the signature σ and the verification key K_v^*.

Phase 2. Same as Phase 1.

Guess. *Adv* produces a guess μ' of μ. If $\mu' = \mu$, *Sim* answers "DBDH" in the DBDH simulation. Otherwise, *Sim* answers "random".

First, we observe that abort occurs only if *Adv* loses in the CCA game by submitting an invalid ciphertext. If $Z = e(g,g)^{abc}$, *Adv* receives a valid ciphertext during the challenge phase. If in addition forge does not occur, *Sim* has the same advantage as *Adv*.

$$\begin{aligned}
&\mathbf{P}[Sim \to \text{“DBDH”}|Z = e(g,g)^{abc}] \\
&= \mathbf{P}[\mu' = \mu|Z = e(g,g)^{abc}] - \mathbf{P}[\mathsf{forge}, \mu' = \mu|Z = e(g,g)^{abc}] \\
&\quad + \mathbf{P}[\mathsf{forge}, Sim \to \text{“DBDH”}|Z = e(g,g)^{abc}] \\
&\geq \mathbf{P}[\mu' = \mu|Z = e(g,g)^{abc}] - \mathbf{P}[\mathsf{forge}|Z = e(g,g)^{abc}] \\
&= \frac{1}{2} + \epsilon - \mathbf{P}[\mathsf{forge}|Z = e(g,g)^{abc}]
\end{aligned}$$

If $Z = e(g,g)^z$, then $\tilde{C}$ is completely random from the view of the adversary. In that case, $\mu' \neq \mu$ holds with probability exactly $\frac{1}{2}$.

$$\begin{aligned}
&\mathbf{P}[Sim \to \text{“random”}|Z = e(g,g)^z] \\
&= \mathbf{P}[\mu' \neq \mu|Z = e(g,g)^z] - \mathbf{P}[\mathsf{forge}, \mu' \neq \mu|Z = e(g,g)^z] \\
&\quad + \mathbf{P}[\mathsf{forge}, Sim \to \text{“random”}|Z = e(g,g)^z] \\
&\geq \mathbf{P}[\mu' \neq \mu|Z = e(g,g)^z] - \mathbf{P}[\mathsf{forge}|Z = e(g,g)^z] \\
&= \frac{1}{2} - \mathbf{P}[\mathsf{forge}|Z = e(g,g)^z]
\end{aligned}$$

Putting the two pieces together, we know that *Sim*'s advantage is at least $\frac{\epsilon}{2} - \mathbf{P}[\mathsf{forge}]$. It remains to prove that $\mathbf{P}[\mathsf{forge}]$ is negligible. To do so, we construct *Sim′* that can win the SEU game with probability at least $\mathbf{P}[\mathsf{forge}]$.

Instead of running SigKeyGen to obtain $\langle K_v^*, K^*s\rangle$, *Sim′* obtains K_v^* from the SEU challenger. Then *Sim′* proceeds as *Sim*. During the challenge phase of the CCA game, *Sim′* obtains the signature σ from the SEU challenger (whereas *Sim* obtains σ by running Sign). If forge occurs, *Sim′* submits the forgery to the SEU challenger and wins. Note that *Sim′* makes at most one signature query. Thus *Sim′* wins the SEU game with probability at least $\mathbf{P}[\mathsf{forge}]$. By the SEU assumption, $\mathbf{P}[\mathsf{forge}]$ must be negligible. □

5. HIERARCHICAL ATTRIBUTES

Recall the basic structure of our CP-ABE schemes. During KeyGen, we associate a random exponent r_i to each $i \in \mathcal{N}$. These exponents are “tied together” using the group element $g^{y-r} = g^{y-\sum_{i=1}^{n} r_i}$, where y is a system master secret. During decryption, the proper matching of attributes allows us to recover $e(g,g)^{r_i \cdot s}$ for every attribute i, including those that do not appear in the AND gate $W = \bigwedge_{i\in I} \underline{i}$. This allows us to recover $e(g,g)^{r\cdot s}$ and subsequently $e(g,g)^{y\cdot s}$.

Clearly, the complexity of encryption/decryption is linear in n, because we must handle every r_i in order to recover r. The same holds for ciphertext size. This is unsatisfactory for small AND gates: if the number of literals appearing in W is small, then most of our work involves *don't care* elements.

In this section, we show that *don't care* elements can be handled more efficiently. The main idea is to arrange attributes into a logical hierarchy, such that a single *don't care* can replace all of those represented in a subtree. This is illustrated in Figure 3, where the system in question has $n = 8$ attributes.

In this example, we augment the public key with randomly chosen group elements $T_{[1..2]}, \ldots, T_{[7..8]}, T_{[1..4]}, T_{[5..8]}, T_{[1..8]}$. If, for example, the encryptor cares only about attributes 1 and 4, then he provides three *don't care* elements in the ciphertext: T_{2n+2}^s, T_{2n+3}^s and $T_{[5..8]}^s$. In contrast, he would have to provide six *don't care* elements under the original scheme: $T_{2n+2}^s, T_{2n+3}^s, T_{2n+5}^s, T_{2n+6}^s, T_{2n+7}^s, T_{2n+8}^s$.

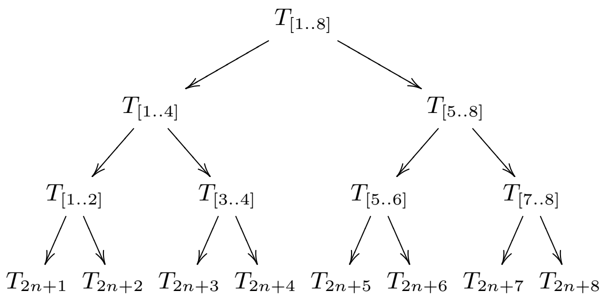

Figure 3: Public Key Elements for Attribute Tree.

Essentially, the hierarchy allows us to cover the complement of I with fewer group elements. This reduces not only the ciphertext size, but also the number of exponentiations in encryption and the number of pairings in decryption. For small AND gates, these costs are brought down from n to $\log(n)$. For large AND gates, the costs will be dominated by operations on attributes that actually appear in the AND gate. In the worst case, every other attribute is included in the AND gate, requiring $\frac{|I|}{2}$ matching *don't care* elements.

The price we pay for this optimization is an increase of roughly n group elements in public/secret key size and an increase of roughly n exponentiations in Setup and KeyGen. Given the storage capacity of modern devices, the larger key size is typically not an issue. The added computations are also not prohibitive, because both Setup and KeyGen are one-time operations whose costs are amortized over the system lifetime.

We now describe in detail the optimized scheme.

Setup. Setup defines a binary tree with one leaf for each attribute $i \in \mathcal{N}$. Assume that attributes are assigned to leaves in ascending order from left to right. Each non-leaf x is associated with the integer range $[i..j]$, where i is the leftmost leaf descended from x and j is the rightmost leaf descended from x. We call this structure the *attribute tree.*

In addition to the public key elements described in Section 3, Setup generates one public key element for every non-leaf node in the attribute tree.[3] This is done as follows: for each *internal* node $x = [i..j]$, Setup chooses $t_x \in \mathbb{Z}_p$ at random and defines $T_x := g^{t_x}$.

Since there are roughly n internal nodes in the attribute tree, the number of group elements in the public key grows from $3n + 1$ to roughly $4n$, and the same is true for the number of exponentiations performed by Setup.

KeyGen. In addition to the secret key elements described in Section 3, KeyGen generates one secret key element for every non-leaf node in the attribute tree. Specifically, for each non-leaf node $x = [i..j]$, KeyGen defines $r_x := \sum_{k=i}^{j} r_k$ and $F_x := g^{\frac{r_x}{t_x}}$. (Recall that KeyGen chooses random exponent r_i for every attribute i.)

[3]The root node can be used to encrypt to every user in the system, regardless of his attributes.

The number of group elements in the secret key grows from $2n+1$ to roughly $3n$, and the same is true for the number of exponentiations performed by KeyGen.

Encrypt. Given message M and AND gate $W = \bigwedge_{i \in I} \underline{i}$, Encrypt selects a random $s \in \mathbb{Z}_p$ and defines $\tilde{C}$ and $\hat{C}$ as in Section 3. To construct the C_i components, Encrypt runs the following recursive procedure (called Traverse) on the root of the access tree. For simplicity of presentation, assume that n is a power of 2.

Traverse returns either 1 or 0 on a given *internal* node in the access tree. Let the current node be x_0 and let x_1 and x_2 denote its children. We have two cases.

- x_1 and x_2 are leaves. Let i and $i+1$ be the attributes associate with x_1 and x_2, respectively.
  - $i \in I$ or $i+1 \in I$. Define C_i and C_{i+1} as in Section 3 and include them in the ciphertext. Return 1.
  - Neither i nor $i+1$ is in I. Return 0.
- x_1 and x_2 are internal nodes. Run Traverse on both.
  - If Traverse returns 1 on both x_1 and x_2, return 1.
  - If Traverse returns 0 on both x_1 and x_2, return 0.
  - Otherwise, assume without loss of generality that Traverse returns 0 on x_1 and 1 on x_2. Define $C_{x_1} := T_{x_1}^s$ and include it in the ciphertext. Return 1.

Finally, Encrypt adds to the ciphertext the necessary indexing information, linking each C_x to the node x.

Decrypt. Let x be a node such that C_x appears in the ciphertext.

- If x is a leaf representing i and $i \in I$, then Decrypt pairs C_x with D_i to obtain $e(g,g)^{r_x \cdot s}$.
- Otherwise, Decrypt pairs C_x with F_x to get $e(g,g)^{r_x \cdot s}$.

Then Decrypt computes $e(g,g)^{r \cdot s}$ as $\prod_x e(g,g)^{r_x \cdot s}$. This is correct because r_x is the sum of all r_i's in the subtree below x in the attribute tree.

5.1 Discussions

We remark that binary trees are chosen simply for notational convenience. Depending on the particular application, attributes can be arranged into semantic categories (cf. the selective data sharing example of Section 7). In general, there is a trade-off between public/secret key size and ciphertext size: if we include more *don't care* elements in the system, corresponding to different subsets of $\mathcal{N}$, we are more likely to be able to represent the whole set $\mathcal{N}$ with fewer group elements. However, encryption time will increase due to the complexity of finding a minimal covering. This trade-off may be valuable in low-bandwidth applications.

We also believe that a similar optimization can be obtained for our CCA scheme. Details are left as future work.

5.2 CPA Security Proof

We modify the CPA proof of Section 3 to accommodate the new changes. The following portions of the proof are affected by the optimization.

Setup. For $i \in \mathcal{N}$, *Sim* calculates the elements T_i, T_{n+1} and T_{2n+i} as in Section 3. Then it calculates T_x for *non-leaf* node $x = [i..j]$ in the attribute tree. First it chooses δ_x at random from $\mathbb{Z}_p$. If there exists $i' \in [i..j]$ such that i' appears in the challenge gate W, then Setup defines $t_x := b \cdot \delta_x$. Otherwise, $t_x := \delta_x$. Then T_x is computed by raising either B or g to δ_x.

Phase 1. *Adv* submits an attribute set $S \subseteq \mathcal{N}$ such that $S \not\models W$. As in Section 3, *Sim* choose an appropriate j and defines r'_i, r_i, $\hat{D}$, D_i and F_i for every $i \in \mathcal{N}$.

Sim must also define F_x for each non-leaf node $x = [i..i']$ in the attribute tree. We have three cases.

- No $k \in [i..i']$ appears W. In this case, $T_x = g^{\delta_x}$ and $F_x = g^{\frac{r_x}{\delta_x}}$, where $r_x = \sum_{k=i}^{i'} r_k$. Since j appears in W, we know that $j \notin [i..i']$. Therefore $r_k = b \cdot r'_k$ for every $k \in [i..i']$ and *Sim* computes F_x as $(\prod_{k=i}^{i'} B^{r'_k})^{\frac{1}{\delta_x}}$.
- Some $k \in [i..i']$ appears W but $j \notin [i..i']$. In this case, $T_x = g^{b \cdot \delta_x}$ and $F_x = g^{\frac{r_x}{b \cdot \delta_x}}$, where $r_x = \sum_{k=i}^{i'} r_k$. Since $j \notin [i..i']$, we know that $r_k = b \cdot r'_k$ for every $k \in [i..i']$. *Sim* computes F_x as $(\prod_{k=i}^{i'} g^{r'_k})^{\frac{1}{\delta_x}}$.
- $j \in [i..i']$. Since j appears in W, we know that $T_x = g^{b \cdot \delta_x}$ and $F_x = g^{\frac{r_x}{b \cdot \delta_x}}$, where $r_x = \sum_{k=i}^{i'} r_k$. Since $j \in [i..i']$, we have $r_x = ab + \sum_{k=i}^{i'} b \cdot r'_k$. *Sim* computes F_x as $(A \cdot \prod_{k=i}^{i'} g^{r'_k})^{\frac{1}{\delta_x}}$.

Challenge. *Sim* chooses random bit μ and defines $\tilde{C}$ as in Section 3. Then it runs Traverse on the attribute tree. Let the current node be x_0 and let x_1 and x_2 denote its children. We have three cases.

- x_1 and x_2 are leaves, and C_i and C_{i+1} are included in the ciphertext. *Sim* computes C_i as in Section 3. More precisely, there are three cases.
  - $i \in I \wedge \underline{i} = i$. Compute C_i as C^{α_i}.
  - $i \in I \wedge \underline{i} = \neg i$. Compute C_i as C^{β_i}.
  - $i \notin I$. Compute C_i as C^{γ_i}.

  Similarly for C_{i+1}.
- x_1 and x_2 are internal nodes, and C_{x_1} is included in the ciphertext. Using the definition of Traverse, it is easy to check that no attribute i in the subtree below x_1 appears in W. *Sim* sets $C_{x_1} := C^{\delta_{x_1}}$.
- x_1 and x_2 are internal nodes, and C_{x_2} is included in the ciphertext. Similar to the previous case.

The rest of the proof proceeds as in Section 3.

6. THRESHOLD ACCESS TREES

Ideally, we would like the ability to encrypt to arbitrary threshold access trees similar to those in [1]. Specifically, consider the following definition of access trees[4]: each internal node in an access tree corresponds to a threshold gate, while all leaf nodes are AND gates. The basic scheme of Section 3 can be extended naturally as follows. To encrypt, the secret exponent s used to mask the message M (in the ciphertext component $\tilde{C} = M \cdot Y^s$) is placed at the root node of the access tree. Based on the threshold gate at the root, shares of s are constructed and placed on the root's children. This is repeated until all nodes in the tree are filled, including the leaves. Then the shares at the leaves are used to produce ciphertext components as in the basic scheme. We illustrate this procedure with an example.

Consider the access tree in Figure 4 and a CP-ABE instance with five attributes $\{1, \ldots, 5\}$. The threshold gate at

[4]Access Trees are access structures that control the ability to decrypt. They are not to be confused with the attribute trees of Section 5.

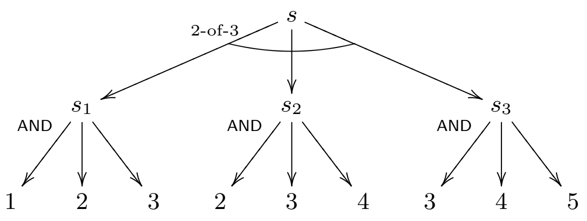

Figure 4: Example of Threshold Access Tree.

the root is 2-of-3, so we select a random degree-1 polynomial p such that $p(0) = s$. For $i = 1, 2, 3$, we set $s_i := p(i)$. Then s can be recovered given any two of the values s_1, s_2 and s_3. At the leaf level, we treat the AND gates as we do in the basic scheme of Section 3. Take, for instance, the leftmost AND gate. The ciphertext includes the following group elements: $\langle g^{s_1}, T_1^{s_1}, T_2^{s_1}, T_3^{s_1}, T_{2\cdot5+4}^{s_1}, T_{2\cdot5+5}^{s_1}\rangle$. This allows any user with attributes $1, 2, 3$ to recover $e(g,g)^{r\cdot s_1} = \prod_{i=1}^{5} e(g,g)^{r_i\cdot s_1}$. The other AND gates are represented similarly in the ciphertext, using the tuples $\langle g^{s_2}, T_{2\cdot5+1}^{s_2}, T_2^{s_2}, T_3^{s_2}, T_4^{s_2}, T_{2\cdot5+5}^{s_2}\rangle$ and $\langle g^{s_3}, T_{2\cdot5+1}^{s_3}, T_{2\cdot5+2}^{s_3}, T_3^{s_3}, T_4^{s_3}, T_5^{s_3}\rangle$. Thus, any user satisfying at least two of the three AND gates can decrypt by recovering $e(g,g)^{r\cdot s}$.

Unfortunately, this extended scheme is insecure. The key point is that s_1, s_2, and s_3 are *not* independent random values. Consider a user with attributes $\neg 1, 2, 3, 4, \neg 5$. Only the middle AND gate is satisfied by this attribute set, and hence decryption should not be allowed. However, notice that 5 is a *don't care* in the first two AND gates. Therefore the user can recover $e(g,g)^{r_5\cdot s_1}$ and $e(g,g)^{r_5\cdot s_2}$. Since s_1 and s_2 are two distinct points on the same degree-1 polynomial, $e(g,g)^{r_5\cdot s_3}$ can be recovered using interpolation. Moreover, the user has attributes 3 and 4, and attributes 1 and 2 are *don't care*'s in the last AND gate, therefore he can obtain (legitimately) $e(g,g)^{r_i\cdot s_3}$ for $i = 1, 2, 3, 4$. Combining these with $e(g,g)^{r_5\cdot s_3}$ from interpolation, the user can now recover $e(g,g)^{r\cdot s_3}$, which in turns allows him to decrypt in violation of the access policy.

To avoid such attacks, it seems necessary to have independent random exponents s_1, s_2, and s_3, so the attacker cannot take advantage of *don't care*'s and polynomial interpolation. For example, one can perform the secret sharing procedure on the message M to obtain shares M_1, M_2 and M_3, which are then encrypted to the three AND gates independently. This proposal, however, suffers from collusion attacks: if one user can recover M_1 and another can recover M_2, then they can collude to recover M.

We note that collusion attacks of this type pose no threat when we restrict to disjunctive normal form (DNF) policies, although a formal security proof remains elusive. The difficulty seems to be that, in the selective ID security game (cf. Section 2), the adversary must commit to a challenge access structure before the game starts. If multiple AND gates are used, the simulator in the reduction proof must choose one of the AND gates and use it to obtain the public key. The chosen AND gate is "good" if it corresponds to a non-negligible jump in success probability in the subsequent hybrid argument. It is unclear how the simulator can predict which AND gate is good before the game starts.

A possible solution is to consider adaptive ID security, which strengthens selective ID security by allowing the adversary to choose a challenge access structure in the Challenge phase. In that setting, we are able to prove the security of multiple encryptions [7]. Thus, we leave as important future work to find adaptive ID security proofs for CP-ABE.

7. APPLICATIONS

As observed in [12], ABE has much potential in providing data security in distributed environments, because it allows complex access policies to be specified and enforced without online interaction with trusted and/or centralized servers. In this section, we outline two applications: selective data sharing and group key management. Many other application scenarios have been mentioned, including audit log and targeted broadcast [10], distributed file systems for medical data and online social networks [12]. The areas of cryptographic file systems and content distribution are also of interest.

7.1 Selective Data Sharing

Consider a scenario in which a large corporation installs a standing committee to investigate any reports of improper conducts of employees. Members of this committee are drawn from different departments and locations, and are given three different clearance levels. Using a hierarchical CP-ABE instance similar to that in Section 5, these attributes can be categorized as in Figure 5.

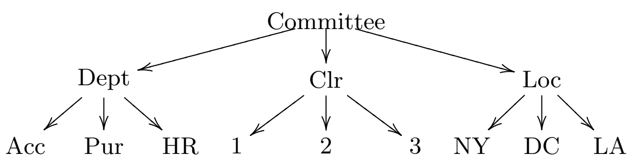

Figure 5: Attributes By Category.

Suppose there is an investigation regarding an accounting officer in New York, and company policy says that no committee member from the accounting department in New York may take part in this investigation. To encrypt a memo regarding this investigation, the content of the memo is first encrypted with a symmetric data key. The data key is then encrypted separately with the AND gates in Figures 6 and 7. (We write "Att:∗" to indicate that we don't care about the attribute "Att".) The two ciphertexts are placed in a header that accompanies the encrypted memo. Anyone not belonging to an accounting department can decrypt the first ciphertext, and anyone not working in New York can decrypt the second. This enforces the desire access policy.

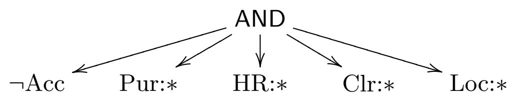

Figure 6: Excluding "Accounting".

7.2 Group Key Management

CP-ABE is well-suited for the problem of Group Key Management (GKM) in the context of secure multicast. This

$$\begin{tikzcd}
 & & \text{AND} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
\text{Dept:}* & \text{Clr:}* & \neg\text{NY} & \text{DC:}* & \text{LA:}*
\end{tikzcd}$$

Figure 7: Excluding "New York".

problem requires a Group Controller (GC) to maintain a shared data encryption key, which is used to encrypt multicast traffic and is known only to current group members (GMs). New GMs are given this data key through a secure unicast channel at the time of joining. The main challenge is membership revocation: excluding a subset of the GMs from future communications. This action requires the distribution of a new data key to all remaining GMs, so that revoked members no long have access to future messages.

It has been noted that CP-ABE can be used to solve the GKM problem. The main idea is to define attributes in such a way that any subset of users can be distinguished from the rest using a combination of attributes. More precisely, each GM is associated with a set of attributes and, for revocation, the GC computes an access policy that is (i) satisfied by the attribute sets of all remaining users and (ii) *not* satisfied by the attribute set of any revoked user. Therefore, remaining GMs can use their secret keys to recover the new data key, while revoked GMs gain no information even if they collude.

In [6], for example, we realized this intuition by constructing a collusion-resistant variant of the *flat table* GKM scheme. This construction uses the CP-ABE scheme of [1] and associates $\log(N)$ attributes with each GM, where N is the size of an ID space. Specifically, each attribute corresponds to one bit in the GM's ID, and the GM receives a CP-ABE secret key associated with his ID. For revocation, the GC runs a Boolean function minimization algorithm to obtain a sum of products expression that separates revoked users from the remaining membership. This expression can be viewed as a threshold access tree, and CP-ABE is used accordingly to distribute the new data key.

8. CONCLUSIONS AND FUTURE WORK

In this paper we present several related CP-ABE schemes. The basic scheme allows an encryptor to use any AND gate on positive and negative attributes as an access policy on the ciphertext. This scheme is proven to be CPA secure under the DBDH assumption. To obtain CCA security, we extend the basic scheme with strongly existentially unforgeable one-time signatures.

We also present a variant with substantially smaller ciphertexts and faster encryption/decryption operations. The main idea is to form a hierarchy of attributes, so that fewer group elements are needed to represent all attributes in the system. This efficient variant is proven to be CPA secure. We believe our CCA secure scheme can be optimized in a similar way.

For disjunctive normal form policies, we propose the use of *independent* instances of CP-ABE encryption. The security of this proposal remains as an open problem, as we look for adaptive ID security proofs for CP-ABE.

Acknowledgments. We thank Ran Canetti, Ron Rivest, Eran Tromer, and Brent Waters for many helpful suggestions. We also thank the anonymous referees for their comments.

9. REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 28th IEEE Symposium on Security and Privacy (Oakland)*, 2007.

[2] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Crytology – CRYPTO 2005*, volume 3621 of *LNCS*, 2005.

[3] D. Boneh, E. Shen, and B. Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Proceedings of PKC 2006*, volume 3958 of *LNCS*, pages 229–240, 2006.

[4] R. Canetti, S. Halevi, and J. Katz. Chosen ciphertext security from identity based encryption. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, 2004.

[5] M. Chase. Multi-authority attribute-based encryption. In *Proceedings of the 4th IACR Theory of Cryptography Conference (TCC 2007)*, 2007.

[6] L. Cheung, J. Cooley, R. Khazan, and C. Newport. Collusion-resistant group key management using attribute-based encryption. Cryptology ePrint Archive Report 2007/161, 2007. Presented at GOCP '07.

[7] L. Cheung and C. Newport. Provably secure ciphertext policy ABE. Cryptology ePrint Archive Report 2007/183, 2007. `http://eprint.iacr.org/`.

[8] A. Fiat and M. Noar. Broadcast encryption. In *Advances in Cryptology – CRYPTO '93*, volume 773 of *LNCS*, pages 480–491, 1993.

[9] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology – CRYTO '99*, pages 537–554, 1999.

[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006)*, pages 89–98, 2006.

[11] D. Naor, M. Naor, and J. Lotspiech. Recovation and tracing schemes for stateless receivers. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62, 2001.

[12] M. Piretti, P. Traynor, P. McDaniel, and B. Waters. Secure atribute-based systems. In *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006)*, 2006.

[13] A. Sahai and B. Waters. Fuzzy identity based encryption. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473, 2005.

[14] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO '84*, pages 47–53, 1985.