Attribute Based Data Sharing with Attribute Revocation

Shucheng Yu
Department of ECE
Worcester Polytechnic Institute
Worcester, MA 01609
yscheng@wpi.edu

Cong Wang
Department of ECE
Illinois Institute of Technology
Chicago, Illinois 60616
cong@ece.iit.edu

Kui Ren
Department of ECE
Illinois Institute of Technology
Chicago, Illinois 60616
kren@ece.iit.edu

Wenjing Lou
Department of ECE
Worcester Polytechnic Institute
Worcester, MA 01609
wjlou@wpi.edu

ABSTRACT

Ciphertext-Policy Attribute Based Encryption (CP-ABE) is a promising cryptographic primitive for fine-grained access control of shared data. In CP-ABE, each user is associated with a set of attributes and data are encrypted with access structures on attributes. A user is able to decrypt a ciphertext if and only if his attributes satisfy the ciphertext access structure. Beside this basic property, practical applications usually have other requirements. In this paper we focus on an important issue of attribute revocation which is cumbersome for CP-ABE schemes. In particular, we resolve this challenging issue by considering more practical scenarios in which semi-trustable on-line proxy servers are available. As compared to existing schemes, our proposed solution enables the authority to revoke user attributes with minimal effort. We achieve this by uniquely integrating the technique of proxy re-encryption with CP-ABE, and enable the authority to delegate most of laborious tasks to proxy servers. Formal analysis shows that our proposed scheme is provably secure against chosen ciphertext attacks. In addition, we show that our technique can also be applicable to the Key-Policy Attribute Based Encryption (KP-ABE) counterpart.

Categories and Subject Descriptors

E.4 [**Data**]: Coding and Information Theory

General Terms

Security, Theory

Keywords

Attribute Based Encryption, Proxy Re-encryption, Revocation

ASIACCS'10 April 13–16, 2010, Beijing, China.
Copyright 2010 ACM 978-1-60558-936-7 ...$10.00.

1. INTRODUCTION

Today's computing technologies have attracted more and more people to store their private data on third-party servers either for ease of sharing or for cost saving. When people enjoy the advantages these new technologies and services bring about, their concerns about data security also arise. Naturally, people would like to make their private data only accessible to authorized users. In many cases, it is also desirable to provide differentiated access services such that data access policies are defined over user attributes/roles. We can easily foresee that these security concerns and requirements would become more urgent in the coming era of cloud computing wherein individuals, organizations, and businesses may outsource their various types of data, including the highly sensitive data, into the cloud. Traditional access control strategies, such as the reference monitor method [1], will not be as effective under this new setting because the service providers and the data owners now very possibly belong to different trusted domains, and the third-party storage servers themselves may not be fully trustworthy. To address this problem, in this paper we explore a feasible solution based on novel cryptographic methods.

Ciphertext-policy attribute based encryption (CP-ABE) [2] is a public-key cryptography primitive that was proposed to resolve the exact issue of fine-grained access control on shared data in one-to-many communications. In CP-ABE, each user is assigned a set of attributes which are embedded into the user's secret key. A public key component is defined for each user attribute. When encrypting the message, the encryptor chooses an access structure on attributes, and encrypts the message under the access structure via encrypting with the corresponding public key components. Users are able to decrypt a ciphertext if and only if their attributes satisfy the ciphertext access structure. The public key and ciphertext sizes in CP-ABE are just linear to the number of attributes and the complexity of the access structure, which is independent to the number of users. Moreover, CP-ABE is resistant to collusion attacks from unauthorized users. All these nice properties make CP-ABE extremely suitable for fine-grained data access control on untrusted storage.

As promising as it is, there also exist several issues when directly applying state-of-the-art CP-ABE schemes to practical applications. These issues can be summarized in two folds: firstly, existing CP-ABE schemes are not able to simultaneously achieve provable security, expressiveness of access structure, and efficient construction; secondly, user management, user revocation in particular, is extremely hard to realize in an efficient way. When current researches are mainly focusing on solving the former, the later has drawn less attention. In fact, user revocation is a challenge issue in many one-to-many communication systems. In attribute based systems, this issue is even more difficult since each attribute is conceivably shared by multiple users. Revocation of any single user would affect others who share his attributes. Moreover, user revocation in attribute based systems may be flexible and occur in different granularities. That is, it may require to revoke either the entire user access privilege, or just partial access right of the user, i.e., a subset of his/her attributes. Existing CP-ABE schemes [18, 2] suggest associating expiration time attributes to user secret keys. However, this type of solutions always have a trade-off between granularity of user revocation and the load placed on the system authority, and require interaction between users and the authority. In addition, the expiration method is not able to efficiently revoke user attributes on the fly. In [4], Boldyreva et al. proposed an efficient revocation scheme for IBE, which is also applicable to KP-ABE [12, 6] and fuzzy IBE [19]. However, it is not clear whether the proposed scheme is applicable to CP-ABE.

Towards building a full fledged CP-ABE system, this paper focuses on the important yet difficult problem of user revocation. Instead of addressing the issue in general settings, we particularly focus on practical application scenarios such as data sharing, as shown by Fig.1, in which semi-trustable proxy servers are always available for providing various types of content services. Similar to previous work [11], we can assume these servers to be curious-but-honest. That is, they will honestly execute the tasks assigned by legitimate parties in the system. However, they would like to learn information of encrypted contents as much as possible. Based on this assumption, our solution uniquely integrates the proxy re-encryption technique with CP-ABE, and enables the authority to delegate most laborious tasks of user revocation to proxy servers without leaking any confidential information to them. On each revocation event, the authority just generates several proxy re-encryption keys and transmits them to proxy servers. Proxy servers will update secret keys for all users but the one to be revoked. Unlike solutions suggested by existing CP-ABE schemes, our construction places minimal load on the authority upon each revocation event, and the authority is able to freely revoke any attribute of users at any time. The only requirement is that proxy servers should stay online and perform honestly. The former can easily be satisfied in many systems since servers provide various kinds of services and should stay online anyway. The later can be guaranteed by exploiting secure computing techniques such as auditing, which is out of the scope of this paper.

1.1 Challenges and Our Contributions

The main challenge of our construction is to formulate a reasonable security model and provide formal security proofs when combining CP-ABE with proxy re-encryption. Our contribution can be summarized as follows.

Firstly, we provide the definition for attribute revocation in CP-ABE with honest-but-curious severs, and formulate the security model to reflect possible attacks.

Secondly, the proposed scheme enables the authority to revoke any attribute of users at any time while placing a minimal load on him.

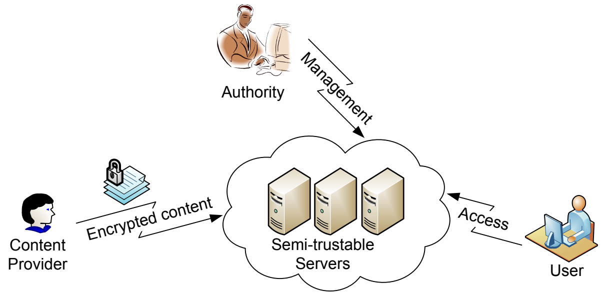

Figure 1: An example application scenario of data sharing.

Thirdly, the proposed scheme is provably secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption.

Last but not least, our method is applicable to the KP-ABE counterpart in which the authority is able to revoke any partial access privilege of users.

To the best of our knowledge, this paper is among the first formally addressing the issue of user/attribute revocation in ABE although it focuses on a practical setting.

1.2 Related Work

Sahai and Waters [19] first introduced attribute based encryption (ABE) for encrypted access control. In an ABE system, both the user secret key and the ciphertext are associated with a set of attributes. Only if at least a threshold number of attributes overlap between the ciphertext and his secret key, can the user decrypt the ciphertext. Goyal et al. [12] first introduced the concept of CP-ABE based on [19]. The idea of a CP-ABE scheme is as follows: the user secret key is associated with a set of attributes and each ciphertext is embedded with an access structure. A user is able to decrypt a ciphertext if and only if his attributes satisfy the access structure of the ciphertext.The access structure is generalized as any boolean formula over threshold gates on positive attributes and negative attributes. Bethencourt et al. [2] proposed the first CP-ABE construction under the generic group model. Cheung et al. [9] proposed the first provably secure CP-ABE under a standard assumption (the DBDH assumption) while only permitting AND gates in the access structure. Goyal et al. recently proposed a bounded CP-ABE scheme with expressive access structure and provable security under the standard model. However, complexity of the construction is extremely high and can just serve as a theoretical feasibility. Further improvements on CP-ABE can be found in [16, 13, 15, 17] etc.

The issue of attribute revocation, a.k.a. key revocation, in CP-ABE was first addressed in [18] as a rough idea. This paper suggests extending each user attribute with an expiration date. This idea, as the authors pointed out, requires the users to periodically go to the authority for key reissuing and thus is inefficient. [2] enhances this solution by associating the user secret key with a single expiration date. As is compared to [18], this solution places a lower load on the authority as users need to update their keys less frequently. However, this method is not able to realize user attribute change in a timely fashion. These solutions can just disable a user secret key at a designated time, but are not able to revoke a user attribute/key on the ad hoc basis. In [4], Boldyreva et al. proposed an efficient revocation scheme for IBE, and the proposed scheme is also applicable to KP-ABE and fuzzy IBE. However, its applicability to CP-ABE is not clear.

In 1998, Blaze et al. [3] proposed a cryptographic primitive in which a semi-trustable proxy is able to convert a ciphertext encrypted under Alice's public key into another ciphertext that can be opened by Bob's private key without seeing the underlying plaintext. This cryptographic primitive is called Proxy Re-Encryption (PRE). A PRE scheme allows the proxy, given the proxy re-encryption key $rk_{a\leftrightarrow b}$, to translate ciphertexts under public key pk_a into ciphertexts under public key pk_b and vise versa. We refer to [3] for more details on proxy re-encryption schemes. Enhancements to proxy re-encryption scheme can be found in [10] etc.

The rest of this paper is organized as follows. Section 2 presents formal definitions and models of our proposed scheme. Section 3 reviews some technique preliminaries pertaining to our construction. In section 4, we describe our construction in detail together with its security proof. Section 5 gives a CCA secure construction. In section 6, we discuss applicability of our method to KP-ABE and some application considerations. We conclude this paper in Section 7.

2. DEFINITIONS AND MODELS

In this section, we first give an overview of our solution to the issue of attribute revocation. Then, we present our definition of the proposed scheme and its security model.

2.1 Scheme Overview

Our scheme is proposed to resolve the issue of attribute revocation for applications such as data sharing as shown in Fig.1. For example, in a campus data system, each student is associated with an attribute set such as (department, courses, club memberships, ...). When a student drops a class or quits from a club, the system needs to remove the corresponding attribute from the student's attribute set. Recall that in CP-ABE [2], the system (the authority) defines a master key component for each attribute in the system. With these master key components, the system defines the public key and user secret key components each of which corresponds to one of the user's attributes. Based on this observation, we propose to resolve the attribute revocation issue as follows:

Whenever an attribute revocation event occurs, the authority redefines the master key components for involved attributes. Corresponding public key components are then updated accordingly. From then on, data will be encrypted with the new public key. Apparently, user secret keys should be updated accordingly for data access. For this purpose, the authority generates proxy re-key's for updated master key components. With these proxy re-key's, the proxy servers are able to securely update user secret keys to the latest version for all but the user for revocation[1]. This removes the involved attributes from that user's attribute set since their corresponding secret key components no longer comply with the new master key. The proxy re-key's also allow the proxy servers to re-encrypt existing ciphertexts stored on them[2] to the latest version without disclosing any plaintext information as we will discuss later. As compared to previous work, this solution places minimal load on the authority since most of the laborious tasks are delegated to proxy servers.

2.2 Algorithm Definition

Our proposed scheme is composed of 7 algorithms: *Setup*, *Enc*, *KeyGen*, *ReKeyGen*, *ReEnc*, *ReKey*, and *Dec*. *Setup*, *KeyGen*, and *ReKeyGen* are performed by the authority while *ReEnc* and *ReKey* are executed by proxy servers. *Enc* and *Dec* are called by encryptors and decryptors respectively. Note that, in our scheme we define a system wide version information *ver* indicating the evolution of the system master key as follows: initially it is set to 1; whenever an attribute revocation event occurs and the system master key is redefined, it increases by 1. The system public key, ciphertexts, user secret keys, and proxy re-key's are all tagged with the version information indicating which version of system master key they comply with.

Setup(1^λ) It takes as input the security parameter 1^λ and outputs the system master key MK and public parameters PK. *ver* is initialized as 1.

Enc(M, AS, PK) It takes as input a message M, an access structure AS, and current public parameters PK, and outputs a ciphertext CT.

KeyGen(MK, S) It takes as input current system master key MK and a set of attributes S that describes the key. It outputs a user secret key SK in the form of (*ver*, S, D, $\bar{D} = \{D_i, F_i\}_{i\in S}$).

ReKeyGen(γ, MK) It takes as input an attribute set γ that includes attributes for update, and current master key MK. It outputs the new master key MK', the new public key PK' (computation of PK' can be delegated to proxy servers), and a set of proxy re-key's rk for all the attributes in the attribute universe U. *ver* is increased by 1. Note that, for attributes in set $U-\gamma$, their proxy re-key's are set as 1 in rk.

ReEnc(CT, rk, β) It takes as input a ciphertext CT, the set of proxy re-key's rk having the same version with CT, a set of attributes β which includes all the attributes in CT's access structure with proxy re-key not being 1 in rk. It outputs a re-encrypted ciphertext CT' with the same access structure as CT.

ReKey$(\bar{D}, rk, \theta)$ It takes as input the component $\bar{D}$ of a user secret key SK, the set of proxy re-key's rk having the same version with SK, and a set of attributes θ which includes all the attributes in SK with proxy re-key not being 1 in rk. It outputs updated user secret key components $\bar{D}'$.

Dec(CT, PK, SK) It takes as input a ciphertext CT, public parameters PK, and the user secret key SK having the same version with CT. It outputs the message M if the attribute set of SK satisfies the ciphertext access structure.

[1] A user secret key is updated when the user accesses proxy servers. Aggregate update for successive events is possible when a user has not accessed the system for a long time.

[2] In practice, this can be done efficiently using the technique of lazy re-encryption[14] as we will discuss later.

Otherwise, it returns $\perp$ with an overwhelming probability.

2.3 Security Definition

We first present the requirements of correctness of our proposed scheme by the following conditions:

(1) $Dec(Enc(M, AS, PK), PK, SK) = M$, if the attribute set S of SK satisfies AS.

(2) Let $CT' = ReEnc(Enc(M, AS, PK), rk, \beta)$, and $SK' = (ver + 1, S, D, \bar{D}' = ReKey(\bar{D}, rk, \theta))$, where ver is the version number of PK and rk. $Dec(CT', PK', SK') = M$, if $S' = S\backslash(\beta\backslash\theta)$ satisfies AS.

(3) Let $CT'' = ReEnc(CT', rk', \beta')$, and $SK'' = (ver + 2, S', D, ReKey(\bar{D}', rk', \theta'))$. If $Dec(CT', PK', SK') = M$ and $S'' = S'\backslash(\beta'\backslash\theta')$ satisfies AS, $Dec(CT'', PK'', SK'') = M$.

(4) Inductively we get the statement for $(CT^{(n)}, PK^{(n)}, SK^{(n)})$ of any n.

CPA security of our proposed scheme under the selective-structure model [9] can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

CPA Security Game Let λ be a security parameter. We say that our scheme is secure against chosen plaintext attacks under selective-structure model if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible advantage.

Init The adversary $\mathcal{A}$ chooses the challenge access structure AS^*, a version number ver^*, and $ver^* - 1$ attribute sets $\{\gamma^{(1)}, \gamma^{(2)}, \cdots, \gamma^{(ver^*-1)}\}$, and submits them to the challenger $\mathcal{B}$.

Setup The challenger $\mathcal{B}$ fist runs Setup(1^λ) to obtain MK and PK for version 1. He then runs $ReKeyGen(\gamma_i, MK)$ from $i = 1$ to ver^*-1. Finally, $\mathcal{B}$ gives $(PK, \{rk^{(i)}\}_{2\leq i\leq ver^*})$ to $\mathcal{A}$, where $rk^{(i)}$ denotes the proxy re-key set for version i[3]. Note that, $\mathcal{A}$ is able to derive PK for all the versions with $rk^{(i)}$'s.

Phase 1 The adversary $\mathcal{A}$ is allowed to issue polynomial times (in λ) of queries on generation of secret keys of any version within the range of $[1, ver^*]$. The only restrict is that the attribute set that $\mathcal{A}$ submits for each secret key query does not satisfy AS^*.

Challenge The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b by executing $CT^* \leftarrow Enc(M, AS^*, PK)$, where PK is the public parameter for version ver^*. The challenge ciphertext CT^* is passed to the adversary.

Phase 2 Phase 1 is repeated.

Guess The adversary $\mathcal{A}$ outputs his guess b_0 of b.

The adversary $\mathcal{A}$ is advantage in winning this CPA security game is defined as $ADV_{CPA} = Pr[b_0 = b] - \frac{1}{2}$.

Note that, In Phase 1, the adversary is also permitted to issue queries on re-encryption of ciphertexts and on update of secret keys. In our security game, however, the adversary has been given all the proxy re-key's. This means that he is able to answer the two queries by himself. For this sake, we do not include the two corresponding oracles in Phase 1. In fact, the adversary $\mathcal{A}$ has at least the same capability as proxy servers who passively collect secret keys of unauthorized users. Since we assume proxy servers are honest, we do not consider active attacks from proxy servers by colluding with revoked authorized users.

Definition 1. (CPA SECURITY) We say that our scheme is CPA secure if ADV_{CPA} is negligible (in λ) for any polynomial time adversary.

3. PRELIMINARIES

3.1 Bilinear Maps

Our design is based on some facts about groups with efficiently computable bilinear maps.

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$. A bilinear map is an injective function $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ with the following properties:

1. *Bilinearity*: for all $u, v \in \mathbb{G}_0$ and a, b $\in \mathbb{Z}_p$, we have e(u^a, v^b) $= e(u, v)^{ab}$.

2. *Non-degeneracy*: $e(g, g) \neq 1$.

3. *Computability*: There is an efficient algorithm to compute $e(u, v)$ for $\forall$ $u, v \in \mathbb{G}_0$.

3.2 Complexity Assumptions

Decisional Bilinear Diffie-Hellman (DBDH) Assumption Let $a, b, c, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}_0$. The DBDH assumption [5] states that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuples $(A = g^a, B = g^b, C = g^c, e(g, g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^z)$ with non-negligible advantage.

4. OUR CONSTRUCTION

4.1 Overview

As described previously, our scheme is to enhance CP-ABE for achieving efficient attribute revocation in terms of loads placed on the authority and users. The basic idea of our construction is to combine the proxy re-encryption technique with CP-ABE. Instead of building a new CP-ABE scheme from scratch, we intend to enhance an existing construction by extending it with abilities of proxy update of secret key and proxy re-encryption of ciphertext. Our construction is partially based on but not limited to Cheung et al's construction of CP-ABE [9].

Attribute and Access Structure In our construction, attributes are represented by their index values and the attribute universe is $U = \{1, 2, \cdots, n\}$ for a certain natural number n. Each attribute would have three occurrences: *positive*, *negative*, and "*don't care*". We just consider access structures consisting of a single AND gate, i.e., the gate $\bigwedge_{\tilde{i}\in I} \tilde{i}$, where I denotes the set of attributes of interest and $\tilde{i}$ is the literal of an attribute i, which can be positive (denoted by $+i$) or negative (denoted by $-i$). If an attribute does not appear in the AND gate, its occurrence is "don't care".

[3] In this paper, the superscript (i) means that the component is of version i. When there is no confusion, we always remove the superscript for brevity. For example, we may just use rk or γ.

4.2 The Detailed Construction

As is defined in section 2.2, there are 7 algorithms in our construction: *Setup*, *Enc*, *KeyGen*, *ReKeyGen*, *ReEnc*, *ReKey*, and *Dec*. Now we present the construction for each of them as follows.

Setup(1^λ) First choose a bilinear group $\mathbb{G}_0$ of prime order p with a generator g, and a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. Next, select random numbers $y, t_1, \cdots, t_{3n} \in \mathbb{Z}_p$. Then, generate the public parameter as: $PK = (e, g, Y, T_1, \cdots, T_{3n})$, where $Y = e(g, g)^y$ and $T_i = g^{t_i}$ for $1 \leq i \leq 3n$. T_i, T_{n+i}, and T_{2n+i} are for the three occurrences of i, i.e., positive, negative, and "don't care", respectively. The system master key MK is: $MK = (y, t_1, \cdots, t_{3n})$. Finally, initialize version number as $ver = 1$ and publish (ver, PK). (ver, MK) is witheld by the authority.

Enc(M, AS, PK) Note that AS is a single AND gate of form $AS = \bigwedge_{\tilde{i} \in I} \tilde{i}$, and assume $M \in \mathbb{G}_1$. The algorithm chooses a random number $s \in \mathbb{Z}_p$ and outputs the ciphertext CT as: $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i \in U})$, where ver is current version number, $\tilde{C} = MY^s$, $\hat{C} = g^s$. For each $i \in I$, C_i is T_i^s if $\tilde{i} = +i$; or T_{n+i}^s if $\tilde{i} = -i$. If $i \in U \backslash I$, $C_i = T_{2n+i}^s$.

KeyGen(MK, S) First choose a random number $r_i \in \mathbb{Z}_p$ for each $i \in U$. Let $r = \sum_{i=1}^n r_i$. User secret key is defined as $SK = (ver, S, D, \bar{D} = \{D_i, F_i\}_{i \in U})$, where ver is current version number, $D = g^{y-r}$. For each $i \in U$, $F_i = g^{\frac{r_i}{t_{2n+i}}}$, and $D_i = g^{\frac{r_i}{t_i}}$ if $i \in S$, or $D_i = g^{\frac{r_i}{t_{n+i}}}$ otherwise. Note that $i \notin S$ means negative occurrence of attribute i in S.

ReKeyGen(γ, MK) Each item $i \in \gamma$ is defined to be within the range of [1,2n]. Value less or equal to n means positive occurrence of the attribute, while value greater than n represents the negative occurrence of attribute $i - n$. The proxy re-key is computed as follows. For each $i \in \gamma$, randomly choose $t'_i \in \mathbb{Z}_p$ and compute $rk_i = \frac{t'_i}{t_i}$. For each $i \in \{1, \cdots, 2n\} \backslash \gamma$, $rk_i = 1$. Output proxy re-key as $rk = (ver, \{rk_i\}_{1 \leq i \leq 2n})$ where ver is current version number. Increase the system version number ver by 1 when everything is done.

ReEnc(CT, rk, β) Denote the access structure of CT as $AS = \bigwedge_{\tilde{i} \in I} \tilde{i}$. Similar to γ, each item in β is also defined to be within the range of [1,2n]. This algorithm directly outputs CT if CT and rk contain different version numbers. Otherwise, re-encrypt CT as follows. For each $i \in \beta$, $C'_i = C_i^{rk_i}$ if $1 \leq i \leq n$, or $C'_{i-n} = (C_{i-n})^{rk_i}$ if $n < i \leq 2n$. For each $i \in U$, $C'_i = C_i$ if $i \notin \beta$ and $i + n \notin \beta$, or $i \notin I$. Ciphertext is output as follows: $CT' = (ver + 1, AS, \tilde{C}, \hat{C}, \{C'_i\}_{i \in U})$, where ver is the version number in CT.

ReKey$(\bar{D}, rk, \theta)$ Each item in θ is defined to be within the range of [1,2n]. This algorithm returns with $\bar{D}$ immediately if $\bar{D}$ and rk contain different version numbers. Otherwise, update $\bar{D}$ as follows. For each $i \in \theta$, $D'_i = D_i^{rk_i^{-1}}$ if $1 \leq i \leq n$, or $D'_{i-n} = (D_{i-n})^{rk_i^{-1}}$ if $n < i \leq 2n$. For each $i \in U$, $D'_i = D_i$ if $i \notin \theta$ and $i + n \notin \theta$. It outputs as follows: $\bar{D}' = \{D'_i, F_i\}_{i \in U}$. ver in the corresponding user secret key SK is increased by 1.

Dec(CT, PK, SK) If any two of CT, PK, and SK have different version numbers, return $\perp$. Otherwise, continue to decrypt as follows. Suppose $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i \in U})$, $SK = (ver, S, D, \bar{D} = \{D_i, F_i\}_{i \in U})$, and denote AS by $AS = \bigwedge_{\tilde{i} \in I} \tilde{i}$. For each $\tilde{i} \in I$, if $\tilde{i} = +i$ and $i \in S$,

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

if $\tilde{i} = -i$ and $i \notin S$,

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

For each $\tilde{i} \notin I$,

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

Ciphertext is decrypted as follows:

$$M = \bar{C} / (e(\hat{C}, \hat{D}) \textstyle\prod_{i=1}^n e(g, g)^{r_i s}).$$

Its correctness can be verified easily.

4.3 CPA Security Proof

Now we prove the CPA security of our scheme. We show the CPA security of our scheme by a theorem.

THEOREM 1. *If a PPT algorithm (the adversary $\mathcal{A}$) wins our CPA security game with non-negligible advantage ADV_{CPA}, we can use this algorithm to construct another PPT algorithm $\mathcal{B}$ to solve the DBDH problem with advantage $\frac{1}{2}ADV_{CPA}$.*

PROOF. In the DBDH game, the challenger chooses random numbers a, b, c from $\mathbb{Z}_p$ and flips a fair coin μ. If $\mu = 0$, set $z = abc$; If $\mu = 1$, set z as a random value in $\mathbb{Z}_p$. $\mathcal{B}$ is given (A, B, C, Z)=$(g^a, g^b, g^c, e(g, g)^z)$ and asked to output μ. To answer this challenge, $\mathcal{B}$ then simulates our CPA security game as follows.

Init The adversary $\mathcal{A}$ chooses the challenge access structure $AS^* = \bigwedge_{\tilde{i} \in I} \tilde{i}$, a version number ver^*, and $ver^* - 1$ attribute sets $\{\gamma^{(1)}, \gamma^{(2)}, \cdots, \gamma^{(ver^*-1)}\}$, and submits them to the challenger.

Setup The challenger $\mathcal{B}$ first generates the public key of version 1 for $\mathcal{A}$ as follows. Y is defined as $e(A, B) = e(g, g)^{ab}$. For each $i \in U$, $\mathcal{B}$ randomly chooses δ_i, ζ_i, and η_i from $\mathbb{Z}_p$. It outputs public parameters as follows.

For $\tilde{i} \in I$, $T_i = g^{\delta_i}$, $T_{n+i} = B^{\zeta_i}$, and $T_{2n+i} = B^{\eta_i}$, if $\tilde{i} = +i$;
if $\tilde{i} = -i$, $T_i = B^{\delta_i}$, $T_{n+i} = g^{\zeta_i}$, and $T_{2n+i} = B^{\eta_i}$;
For $\tilde{i} \notin I$, $T_i = B^{\delta_i}$, $T_{n+i} = B^{\zeta_i}$, and $T_{2n+i} = g^{\eta_i}$.

Then, $\mathcal{B}$ generates ver^* versions and answers $ver^* - 1$ proxy re-key generation requests. Specifically, for each attribute set $\gamma^{(k)}$, $1 \leq k \leq ver^* - 1$, generate a PK for that version as follows:

for each element $j \in \gamma^{(k)}$, where $1 \leq j \leq 2n$, randomly choose $rk_j^{(k)}$ from $\mathbb{Z}_p$. if $1 \leq j \leq n$,

$$T_j^{(k+1)} = (T_j^{(k)})^{rk_j^{(k)}}, T_{n+j}^{(k+1)} = T_{n+j}^{(k)}, T_{2n+j}^{(k+1)} = T_{2n+j}^{(k)},$$

if $n < j \leq 2n$,

$$T_{j-n}^{(k+1)} = T_{j-n}^{(k)}, T_j^{(k+1)} = (T_j^{(k)})^{rk_j^{(k)}}, T_{n+j}^{(k+1)} = T_{n+j}^{(k)},$$

where superscripts (k) and $(k+1)$ denote the version number of each attribute set, re-key, and public key parameter.

For each element $1 \leq j \leq 2n$, if $j \notin \gamma^{(k)}$, set $rk_j^{(k)} = 1$, and calculate public key components in the same way as above. Finally, $\mathcal{B}$ returns $rk^{(k)} = (k, rk_1^{(k)}, rk_2^{(k)}, \cdots, rk_{2n}^{(k)})$ to $\mathcal{A}$.

Phase 1 Without loss of generality, we assume the adversary $\mathcal{A}$ submits secret key query on a set $S \subseteq U$ for version k, $1 \leq k \leq ver^*$. Since S does not satisfy the challenge access structure AS^*, we know there is a witness attribute $i \in I$ that either $i \in S$ and $\tilde{i} = -i$, or $i \notin S$ and $\tilde{i} = +i$. Without loss of generality, we assume $i \notin S$ and $\tilde{i} = +i$. $\mathcal{B}$ first chooses a random number $r'_j \in \mathbb{Z}_p$ for each $j \in U$. Then, it sets $r_j = r'_j \cdot b$ for every $j \neq i$ (non-witness attribute), and $r_j = ab + r'_j \cdot b$. Finally, it calculates $r = \Sigma_{j \in U} r_j = ab + \Sigma_{j \in U} r'_j \cdot b$. Secret key components are then returned as follows:

$D = \Pi_{j=1}^{n} B^{-r'_j} = g^{-\Sigma_{j=1}^{n} r'_j \cdot b} = g^{ab-r}$.

Consider that for any $j \in U$,

$$T_j^{(k)} = (T_j^{(1)})^{rk_j^{(2)} \cdot rk_j^{(3)} \cdots rk_j^{(k)}} = T_j^{\Pi_{i=2}^{k} rk_j^{(i)}}, \text{ and}$$

$$T_{n+j}^{(k)} = (T_{n+j}^{(1)})^{rk_{n+j}^{(2)} \cdot rk_{n+j}^{(3)} \cdots rk_{n+j}^{(k)}} = (T_{n+j})^{\Pi_{i=2}^{k} rk_{n+j}^{(i)}},$$

we denote $R_j^{(k)} = \Pi_{i=2}^{k} rk_j^{(i)}$ and $R_{n+j}^{(k)} = \Pi_{i=2}^{k} rk_{n+j}^{(i)}$. For each $j \in U$ and $j \neq i$, D_j is calculated as follows.

`Case 1.` $j \in S$.

1) $D_j = B^{\frac{r'_j}{\delta_j \cdot R_j^{(k)}}} = g^{\frac{r_j}{\delta_j \cdot R_j^{(k)}}}$, if $j \in I$ and $\tilde{j} = +j$;

2) $D_j = B^{\frac{r'_j}{\delta_j \cdot R_j^{(k)}}} = g^{\frac{r_j}{\delta_j \cdot R_j^{(k)} \cdot b}}$, if $j \in I$ and $\tilde{j} = -j$, or $j \notin I$;

`Case 2.` $j \notin S$.

1) $D_j = g^{\frac{r'_j}{\zeta_j \cdot R_{n+j}^{(k)}}} = g^{\frac{r_j}{\zeta_j \cdot R_{n+j}^{(k)} \cdot b}}$, if $j \in I$ and $\tilde{j} = j$, or $j \notin I$;

2) $D_j = B^{\frac{r'_j}{\zeta_j \cdot R_{n+j}^{(k)}}} = g^{\frac{r_j}{\zeta_j \cdot R_{n+j}^{(k)}}}$, if $j \in I$ and $\tilde{j} = -j$.

D_i is calculated as:

$$D_i = A^{\frac{1}{\zeta_i \cdot R_i^{(k)}}} \cdot g^{\frac{r'_i}{\zeta_i \cdot R_i^{(k)}}} = g^{\frac{ab + r'_i \cdot b}{\zeta_i \cdot R_i^{k} \cdot b}} = g^{\frac{r_i}{\zeta_i \cdot R_i^{(k)} \cdot b}}.$$

For each attribute $j \in U$, F_j is calculated as follows. If $j \neq i$, then

1) $F_j = g^{\frac{r'_j}{\eta_j}} = g^{\frac{r_j}{\eta_j \cdot b}}$, if $j \in I$;

2) $F_j = B^{\frac{r'_j}{\eta_j}} = g^{\frac{r_j}{\eta_j}}$, if $j \notin I$;

F_i is calculated as follows:

$$F_i = A^{\frac{1}{\eta_i}} \cdot g^{\frac{r'_i}{\eta_i}} = g^{\frac{ab + r'_i \cdot b}{\eta_i \cdot b}} = g^{\frac{r_i}{\eta_i \cdot b}}.$$

Challenge. The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, sets $\tilde{C} = M_b \cdot Z$, and outputs the ciphertext CT^* as follows.

$$CT^* = (ver^*, AS^*, \tilde{C}, C, \{C^{\delta_i \cdot R_i^{(ver^*)}}\}_{i \in I \wedge \tilde{i} = +i}, \{C^{\zeta_i \cdot R_{n+i}^{(ver^*)}}\}_{i \in I \wedge \tilde{i} = -i}, \{C^{\eta_i}\}_{i \notin I}).$$

Phase 2. Phase 1 is repeated.

Guess. $\mathcal{A}$ submits a guess b_0 of b. If $b_0 = b$, $\mathcal{B}$ will output $\mu' = 0$, meaning that (A, B, C, Z) is a valid DBDH-tuple; otherwise, $\mathcal{B}$ outputs $\mu' = 1$, indicating that (A, B, C, Z) is just a random 4-tuple. In the case of μ= 1, the adversary obtains no information about b. We thus have $Pr[b_0 \neq b | \mu = 1] = \frac{1}{2}$. $\mathcal{B}$ just randomly guesses μ'= 1 when $b \neq b_0$, we have $Pr[\mu' = \mu] = \frac{1}{2}$. In the case of μ= 0, the adversary obtains an encryption of m_b, and his advantage is ADV_{CPA} by definition. We thus have $Pr[b = b_0 | \mu = 0] = \frac{1}{2} + ADV_{CPA}$. Since $\mathcal{B}$ guesses $\mu' = 0$ whenever $b = b_0$, we have $Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + ADV_{CPA}$. The overall advantage of $\mathcal{B}$ in the DBDH game is $\frac{1}{2} Pr[\mu' = \mu | \mu = 0] + \frac{1}{2} Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2}(\frac{1}{2} + ADV_{CPA}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} = \frac{1}{2} ADV_{CPA}$.

□

5. CCA SECURITY CONSTRUCTION

We now proceed to discuss the construction of the chosen ciphertext secure scheme. For IBE schemes, a common practice of constructing a CCA secure scheme from a CPA secure one is to generate one-time signature keys (K_v, K_s) and sign the ciphertext with K_s with a strongly existentially unforgeable signature scheme, while K_v is viewed as the message receiver's identity. This technique was proposed by Canetti, Halvei, and Katz [7]. In [9], Cheung and Newport applied the similar technique to CP-ABE and constructed a CCA secure CP-ABE scheme from the CPA secure one. Their construction defines an attribute for each bit in the key space of K_v, each attribute having two occurrences for its binary values. Each user secret key contains two components for the both occurrences of each bit. Thereafter, these attributes are treated similarly as other normal attributes. For encryption, the encryptor chooses a pair (K_v, K_s) and encrypts the message with the attributes for K_v in addition to other normal attributes. The whole ciphertext is then signed with K_s. The ciphertext along with the signature is sent to receiver(s), who will verify the signature before decryption.

In our work, it seems to be a contradiction to construct a CCA secure scheme since we on one hand require the ciphertext to be non-malleable, and on the other hand give the proxy re-key's to proxy servers and allow them to re-encrypt ciphertexts. However, in our scheme ciphertext re-encryption is just limited to updating partial ciphertext components to the latest version. Modification of the underlying message or the access structure is not permitted. In terms of non-malleability, we just need to prevent adversaries from modifying the message or the access structure. Based on this observation, we adopt the same technique as [9] but just sign on partial ciphertext components.

5.1 CCA Secure Construction

The seven algorithms in the CCA secure construction are defined as follows, assuming that the signature verification key K_v has w bits. Denote the set $\{1, 2, \cdots, w\}$ as W.

Setup(1^λ) The same as the CPA secure construction except that, here $2w$ extra attributes are defined for K_v. Now the system master key is: $MK = (y, t_1, \cdots, t_{3n}, t_{3n+1}, \cdots, t_{3n+2w})$, and the public parameters are: PK=$(e, g, Y, T_1, \cdots, T_{3n}, T_{3n+1}, \cdots, T_{3n+2w})$. Initialize the system wide version number ver as 1 and publish (ver, PK). (ver, MK) is kept by the authority.

Enc(M, AS, PK) AS is defined to be an AND gate as before. The encryptor first chooses one-time signature key pair (K_v, K_s), and a random number $s \in \mathbb{Z}_p$. M is encrypted as: $(ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i \in U}, \{K_i\}_{i \in W}, K_v)$, where ver is current version number, $\tilde{C} = MY^s$, $\hat{C} = g^s$. For each $i \in I$, $C_i = T_i^s$ if $\tilde{i} = +i$; or $C_i = T_{n+i}^s$ if $\tilde{i} = -i$. If $i \in U \backslash I$, $C_i = T_{2n+i}^s$. For each $i \in W$, $K_i = T_{3n+i}^s$ if the ith bit of K_v is 0, otherwise, $K_i = T_{3n+w+i}^s$. The encryptor then signs on tuple $(AS, \tilde{C}, \hat{C}, \{K_i\}_{i \in W}, K_v)$ with K_s, and obtains a signature δ. Finally, the ciphertext of M is output as $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i \in U}, \{K_i\}_{i \in W}, K_v, \delta)$.

KeyGen(MK, S) First choose a random numbers $r_i \in \mathbb{Z}_p$ for each $i \in U \bigcup W$. Let $r = \sum_{i=1}^{w+n} r_i$. The secret key is defined as $SK = (ver, S, D, \bar{D} = \{D_i, F_i\}_{i \in U}, \hat{D} = \{\hat{D}_{i,0}, \hat{D}_{i,1}\}_{i \in W})$, where D and $\bar{D}$ are the same as the CPA secure construction. $\hat{D}$ is defined as: $\hat{D}_{i,0} = g^{\frac{r_{n+i}}{t_{3n+i}}}$ and $\hat{D}_{i,1} = g^{\frac{r_{n+i}}{t_{3n+w+i}}}$ for each $i \in W$. Note that this definition extends the one defined in section 2.2.

ReKeyGen(γ, MK) The same as the CPA secure construction.

ReEnc(CT, rk, β) Let CT be $(ver,\ AS,\ \tilde{C},\ \hat{C},\ \{C_i\}_{i \in U},\ \{K_i\}_{i \in W},\ K_v,\ \delta)$. The re-encrypted ciphertext is output as $CT' = (ver + 1, AS, \tilde{C}, \hat{C}, \{C'_i\}_{i \in U}, \{K_i\}_{i \in W}, K_v, \delta)$, where each C'_i is generated in the same way as the CPA secure construction.

ReKey$(\bar{D}, rk, \theta)$ The same as the CPA secure construction.

Dec(CT, PK, SK) The decryptor first verifies the signature δ. On failure, return $\perp$; otherwise, proceed as in the CPA secure construction.

5.2 CCA Security Proof

We first give a definition on CCA security of our scheme. Then, we sketch the security proof.

CCA Game Let λ be a security parameter. We say that our scheme is secure against chosen ciphertext attacks under selective-structure model if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible advantage.

Init and **Setup** Same as the CPA security game.

Phase 1 The adversary is allowed to adaptively make polynomial times (in λ) of any combination of secret key and decryption queries.

Query for Secret Key The adversary submits an attribute set S. The challenger returns a secret key SK for S, given that S does not satisfies AS^*.

Query for Decryption The adversary $\mathcal{A}$ submits a ciphertext CT. If CT is not a valid ciphertext, $\mathcal{A}$ loses the game; otherwise, the challenger $\mathcal{B}$ returns the plaintext M.

Challenge Same as the CPA security game.

Phase 2 Same as Phase 1. Similar to [8], ciphertexts submitted for decryption are not allowed to be derivatives of CT^*. A derivative of CT^* is defined as any CT that can be used to derive CT^* by repeatedly executing algorithm *ReEnc* on proxy re-key's $rk^{(2)}, rk^{(3)}, \cdots, rk^{(ver^*)}$.

Guess The adversary $\mathcal{A}$ outputs his guess b_0 of b.

The adversary $\mathcal{A}$ is advantage in winning this CCA security game is defined as $ADV_{CCA} = Pr[b_0 = b] - \frac{1}{2}$.

Definition 2. (CCA SECURITY) We say that our scheme is CCA secure if ADV_{CCA} is negligible (in λ) for any polynomial time adversary.

CCA security of our scheme can be shown by the following theorem.

THEOREM 2. *If a PPT algorithm (the adversary $\mathcal{A}$) wins our CCA security game with non-negligible advantage ADV_{CCA}, we can use this algorithm to construct another PPT algorithm $\mathcal{B}$ to solve the DBDH problem with advantage $\frac{1}{2}ADV_{CCA}$, assuming that the signature scheme is strongly existentially unforgeable.*

PROOF. The challenger of the DBDH game generates the tuple (A, B, C, Z) exactly as in the CPA security proof, and then sends it to $\mathcal{B}$. To answer this challenge, first choose a signature key pair (K_v^*, K_s^*) and then simulates our CPA security game as follows.

Init The same as the CPA security proof.

Setup In this phase, $\mathcal{B}$ generates $(Y, T_1, \cdots, T_{3n})$ and $(rk^{(2)}, rk^{(3)}, \cdots, rk^{(ver^*)})$ exactly the same as the CPA security proof. $\mathcal{B}$ generates $(T_{3n+1}, \cdots, T_{3n+2w})$ as follows. For each $i \in W$, select random numbers $\phi_i, \psi_i \in \mathbb{Z}_p$, and set $T_{3n+i} = g^{\phi_i}$ and $T_{3n+w+i} = B^{\psi_i}$ if the ith bit of K_v^* is 0, denoted by $K_{v,i}^* = 0$; otherwise, set $T_{3n+i} = B^{\phi_i}$ and $T_{3n+w+i} = g^{\psi_i}$.

Phase 1. $\mathcal{B}$ answers queries for secret key and for decryption.

Case 1. Query for secret key. $\mathcal{B}$ executes in the same way as Phase 1 of the CPA security game. When generating $(D_{j,0}, D_{j,1})$ for each $j \in W$, $\mathcal{B}$ deals in the same way as non-witness attributes in U except that, R_j^k or R_{n+j}^k are no longer needed when computing $D_{j,0}$ and $D_{j,1}$ since $\hat{D}$ part of a user secret key never needs update.

Case 2. Query for decryption. $\mathcal{A}$ submits a ciphertext $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i \in U}, \{K_i\}_{i \in W}, K_v, \delta)$ for decryption. $\mathcal{B}$ first verifies the signature δ with K_v. If the signature is not valid, $\mathcal{B}$ terminates the DBDH simulation game without answering the DBDH challenger and start a new game. Otherwise, proceed. In this case, we know $K_v \neq K_v^*$ with overwhelming probability. Otherwise, K_v can be used to successfully verify δ and the signature contained in the challenge ciphertext, which is assumed to happen with negligible probability since the signature scheme is strongly existentially unforgeable. In case of $K_v \neq K_v^*$, we can assume the jth bits of them are different. Without loss of generality, we assume that the bit of K_v^* is 0. Therefore, $K_j = T_{3n+w+j}^s = g^{b \cdot \psi_j \cdot s}$. $\mathcal{B}$ then calculates $e(K_j, A) = e(g, g)^{abs\psi_j} = Y^{s\psi_j}$. Since ψ_j is known to $\mathcal{B}$, he gives $\tilde{C}/(e(K_j, A)^{\frac{1}{\psi_j}})$ to $\mathcal{A}$ as the message M.

Challenge. The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, sets $\tilde{C} = M_b \cdot Z$, and outputs the ciphertext CT^* as follows.

$$CT^* = (ver^*, AS^*, \tilde{C}, C, \{C^{\delta_i \cdot R_i^{(ver^*)}}\}_{i \in I \wedge \tilde{i} = +i}, \{C^{\eta_i}\}_{i \notin I},$$
$$\{C^{\zeta_i \cdot R_{n+i}^{(ver^*)}}\}_{i \in I \wedge \tilde{i} = -i}, \{C^{\phi_i}\}_{i \in W \wedge K_{v,i}^* = 0}, \{C^{\psi_i}\}_{i \in W \wedge K_{v,i}^* = 1}).$$

Phase 2. Repeat Phase 1. The only restriction is that, ciphertexts submitted for decryption are not allowed to be derivatives of CT^*. $\mathcal{B}$ is able to verify this by running algorithm $ReEnc$ on proxy re-key's and CT^*, and compare the results with the ciphertexts he received from $\mathcal{A}$.

Guess. The same as the CPA security proof.

□

6. DISCUSSION

6.1 Applicability to KP-ABE

Key Policy Attribute-Based Encryption (KP-ABE) [12] is a sister technique of CP-ABE but the situation is reversed: In KP-ABE, ciphertexts are associated with attributes, while user secret keys are defined with access structures on attributes. If only the ciphertext attributes satisfy a user's access structure, can he decrypt. When CP-ABE is applicable in Role-Based Access Control like scenarios, KP-ABE is suitable for applications such as pay-per-view TV systems, in which user access privileges are defined over content attributes and could be based on the prices they paid. In these application scenarios, the issue of key revocation also exists. Fig. 2. shows such an example, in which a user currently is allowed to access any series with name "Hero", "Lost", or "Dexter"' provided by channel 4. The system administrator now wants to disable the user's access privilege on series with name "Lost" for some reason (maybe late payment). For this purpose, it is necessary to revoke the corresponding component of the user's secret key.

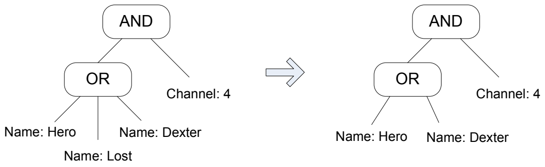

Figure 2: An example application scenario of KP-ABE.

Similar to CP-ABE, the basic construction of current KP-ABE scheme [12] also defines a system master key component t_i for each attribute i. The corresponding public key component is defined as $T_i = g^{t_i}$. Encrypting a message with attribute i means including a component T_i^s into the ciphertext, where s is a random number for this ciphertext. In user secret key, the component for attribute i has the form of $g^{\frac{q_x(0)}{t_i}}$, where $q_x(\cdot)$ is a polynomial uniquely defined for the user. Therefore, we can revoke a secret key component in the same way as we did for CP-ABE, i.e., the authority redefines the master key component as t_i' and give $\frac{t_i'}{t_i}$ to proxy servers as the proxy re-key. In the same way as our CP-ABE scheme, proxy servers, which are honest by our assumption, will use these proxy re-key's to re-encrypt ciphertexts stored on them and update secret keys for all but the user for revocation. Proof of the new KP-ABE scheme is similar to that of our CP-ABE scheme.

Large Universe Construction In addition to the basic construction, [12] also provides a KP-ABE construction for large universe cases. One significant advantage of this construction is that the number of public parameter components is constant, no matter how many attributes the system may have. In this construction, however, our technique of generating proxy re-key's for CP-ABE and the basic KP-ABE scheme is not applicable any more. This is because the definitions of public parameter components and user secret key components are no longer in the same format as before. In this construction, it selects $n+1$ random points from $\mathbb{G}_1$ [4] and defines a function T to calculate the public key component for attribute X [5] as:

$$T(X) = g_2^{X^n} \cdot \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(X)},$$

where g_2 is another group element in $\mathbb{G}_1$, $N = \{1, 2, \cdots, n+1\}$, and $\Delta_{i,N}(X)$ is the Lagrange coefficient. Ciphertext component for each attribute i is still in the form of T_i^s as before. Each attribute has two components in user secret key:

$$D_i = g_2^{q_x(0)} \cdot T(i)^{r_i}, \text{ and } R_i = g^{r_i},$$

where $r_i \in \mathbb{Z}_p$ is a random number of attribute i in the user secret key. We refer to [12] for the detailed construction.

In this construction, we can not simply redefine the system master key component t_i to update the attribute i for key revocation as before. Instead, we need to change the construction of the $Setup()$ algorithm of the original scheme as follows. We assume the bit string for each attribute is defined in a fixed format "(attribute description, version j)". The version number j of each attribute in the universe will be published.

Setup(n) Choose a random number $y \in \mathbb{Z}_p$ and let $g_1 = g^y$. Next, choose a random element g_2 from $\mathbb{G}_1$. Then, select random numbers $w_1, w_2, \cdots, w_{n+1}$ from $\mathbb{Z}_p$. Define $t_i = g_2^{w_i}$ for $1 \le i \le n+1$. Define function T as:

$$T(X) = g_2^{X^n} \cdot \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(X)} = g_2^{X^n + p(X)},$$

where $p(\cdot)$ is a n degree polynomial defined by points $(1, w_1)$, $(2, w_2)$, $\cdots$, $(n+1, w_{n+1})$. The public parameters are output as: $PK = (g_1, g_2, t_1, \cdots, t_{n+1})$. The master key $MK = (y, w_1, w_2, \cdots, w_{n+1})$.

Algorithms $Encryption$, $Key\ Generation$, and $Decryption$ are defined exactly the same as the original scheme. To enable the authority to generate proxy re-key's, we define algorithm $ReKeyGen$ as follows.

ReKeyGen(γ, MK) γ is the set of attributes needing redefinition. For each attribute $X \in \gamma$, assuming its pre-image is a bit string "(attribute description, version j)", now redefine it as bit string "(attribute description, version $j+1$)". We hence obtain H(attribute name, version $j+1$) $= X'$ [6], where $H()$ is a cryptographic hash function. Since hash function $H(\cdot)$ is collision free, X' can not be used for any other attributes. The proxy re-key for attribute X will be output as $rk_X = \frac{(X')^n + p(X')}{X^n + p(X)}$. The set of proxy re-key's rk for attributes in γ are sent to proxy servers.

Proxy servers, on receiving the proxy key's, re-encrypt existing ciphertexts stored on them as follows.

[4] Following the definition of [12], we assume the bilinear map is from $\mathbb{G}_1$ to $\mathbb{G}_2$, i.e., $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

[5] X is generated by applying a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ on the bit string representation of the attribute.

[6] When the new version $j+1$ is published, encryptors will use $T(X')$ as the public component for encryption thereafter.

ReEnc(E, rk, β) For each attribute $i \in \beta$, update E_i as $E_i^{rk_i}$.

To update user secret key, proxy servers update R_i components for users as follows.

ReKey(R_i, rk_i) Update R_i as $R_i^{(rk_i)^{-1}}$.

It is easy to verify that the updated user secret key will be able to decrypt ciphertexts encrypted with updated attributes if the ciphertext attributes satisfy the access structure of the secret key. Formal security reduction of this scheme will be presented in the extended version of this paper.

As compared to [4], our scheme places a minimal computation overhead on the authority, and is more suitable for application scenarios in which the content provider, e.g., an individual person who himself/herself serves as the authority, outsources contents on powerful but semi-trustable servers such as cloud servers, and would like to manage his/her resources via resource constrained devices, e.g., iPhone, anywhere and at any time. On the other hand, [4] is more efficient in terms of the overall system overhead and is more suitable for traditional applications in which most computational tasks are executed locally.

6.2 Application Considerations

Our scheme considers application scenarios of data sharing in which data are encrypted and stored on semi-trustable servers for sharing. In this scheme, the authority generates proxy re-key's whenever an attribute revocation event occurs. Proxy re-key's are then transmitted to proxy servers, who will re-encrypt existing ciphertexts stored on them and update user secret key components if necessary. For simplicity of description, our scheme just considers one revocation event. Multiple revocation events are assumed to be handled by repeatedly executing these operations. When this assumption is convenient for theoretical analysis of the scheme, it will cause efficiency issue in practice since proxy servers have to re-encrypt ciphertexts stored upon each revocation event. In practical systems, there could be a huge number of files stored on servers, and the computation load for re-encrypting them could be extremely heavy. On the other hand, users are not necessary available for key update upon each revocation event. In practical scenarios, users may have missed many revocation events before they come back to access the servers. To deal with attribute revocation efficiently, we propose to enable proxy servers to handle revocation events in an aggregative way, which further makes lazy re-encryption [14] possible. For this purpose, proxy servers keep a copy of a table for proxy re-key's of historical events as shown in Fig.3.

Version #	**att 1**	**att 2**	$\cdots$	**att** n
k	$rk_1^{(k)}$	$rk_2^{(k)}$	$\cdots$	$rk_n^{(k)}$
$k-1$	$rk_1^{(k-1)}$	$rk_2^{(k-1)}$	$\cdots$	$rk_n^{(k-1)}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$k-N$	$rk_1^{(k-N)}$	$rk_2^{(k-N)}$	$\cdots$	$rk_n^{(k-N)}$

Figure 3: Proxy re-key list

With this table, proxy servers do not need to re-encrypt data files upon each revocation event. Instead, they can just re-encrypt the data files when the files are accessed by some user, i.e., in the fashion of lazy re-encryption. Assuming the version number associated with the ciphertext of a data file is $k - i$, updating it to the latest version k just needs to call the *ReEnc* algorithm once with proxy re-key $(\prod_{j=0}^{i-1} rk_1^{(k-j)}, \prod_{j=0}^{i-1} rk_2^{(k-j)}, \cdots, \prod_{j=0}^{i-1} rk_n^{(k-j)})$. Unaccessed data files will never get re-encrypted. This modification can aggregate operations for multiple revocation events into one and save a lot of computation overload for proxy servers from statistical point of view. One issue with this method is that the storage overhead could be high if proxy servers keep all the proxy re-key's. In practical systems, we can just keep the list for a reasonable time period and release the storage burden on proxy servers.

User secret key update can be addressed in the same way. However, proxy servers need to keep a revoked user identity list for each attribute since a user may come back asking for update of secret key components of attributes that were previously revoked. Having these lists, proxy servers will just update secret key components of attributes that are still associated with the user. Recovering an attribute for a user can also be realized by removing the user's ID from the list.

Att i	ID_i	ID_j	$\cdots$	ID_k

Figure 4: Revoked user ID list of attribute i

7. CONCLUSION AND FUTURE WORK

In this paper we addressed an important issue of attribute revocation for attribute based systems. In particular, we considered practical application scenarios in which semi-trustable proxy servers are available, and proposed a scheme supporting attribute revocation. One nice property of our proposed scheme is that it places minimal load on authority upon attribute revocation events. We achieved this by uniquely combining the proxy re-encryption technique with CP-ABE and enabled the authority to delegate most laborious tasks to proxy servers. Our proposed scheme is provably secure against chosen ciphertext attacks. In addition, we also showed the applicability of our method to the KP-ABE scheme. One interesting future work is to combine a secure computation technique with our construction to guarantee the honesty of proxy servers. Another direction for future work is to allow proxy servers to update user secret key without disclosing user attribute information.

8. ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grants CNS-0716306, CNS-0831628, CNS-0746977, and CNS-0831963.

9. REFERENCES

[1] J. Anderson. Computer Security Technology Planning Study. Air Force Electronic Systems Division, Report ESD-TR-73-51, 1972. http://seclab.cs.ucdavis.edu/projects/history/.

[2] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *Proc. of SP'07*, Washington, DC, USA, 2007.

[3] M. Blaze, G. Bleumer, and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *Proc. of EUROCRYPT '98*, Espoo, Finland, 1998.

[4] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based Encryption with Efficient Revocation. In *Proc. of CCS'08*, Alexandria, Virginia, USA, 2008.

[5] D. Boneh and M. Franklin. Identity-Based Encryption from The Weil Pairing. In *Proc. of CRYPTO'01*, Santa Barbara, California, USA, 2001.

[6] S. Yu, K. Ren, W. Lou, and J. Li. Defending Against Key Abuse Attacks in KP-ABE Enabled Broadcast Systems. In *Proc. of Securecomm'09*, Athens, Greece, 2009.

[7] R. Canetti, S. Halevi, and J. Katz. Chosen Ciphertext Security from Identity Based Encryption. In *Proc. of EUROCRYPT'04*, Interlaken, Switzerland, 2004.

[8] R. Canetti and S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In *Proc. of CCS'07*, New York, NY, USA, 2007.

[9] L. Cheung and C. Newport. Provably Secure Ciphertext Policy ABE. In *Proc. of CCS'07*, New York, NY, USA, 2007.

[10] R. H. Deng, J. Weng, S. Liu, and K. Chen. Chosen-Ciphertext Secure Proxy Re-encryption without Pairings. In *Proc. of CANS'08*, Berlin, Heidelberg, 2008.

[11] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Over-encryption: Management of Access Control Evolution on Outsourced Data. In *Proc. of VLDB'07*, Vienna, Austria, 2007.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data. In *Proc. of CCS'06*, Alexandria, Virginia, USA, 2006.

[13] S. Yu, K. Ren, and W. Lou. Attribute-Based On-Demand Multicast Group Setup with Membership Anonymity. In *Proc. of SecureComm'08*, Istanbul, Turkey, 2008.

[14] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable Secure File Sharing on Untrusted Storage. In *Proc. of FAST'03*, Berkeley, California, USA, 2003.

[15] J. Li, K. Ren, B. Zhu, and Z. Wan. Privacy-Aware Attribute-Based Encryption with User Accountability. In *Proc. of ISC'09*, Pisa, Italy, 2009.

[16] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute Based Proxy Re-encryption with Delegating Capabilities. In *Proc. of ASIACCS'09*, Sydney, Australia, 2009.

[17] S. Yu, K. Ren, and W. Lou. Attribute-Based Content Distribution with Hidden Policy. In *Proc. of NPSEC'08*, Orlando, Florida, USA, 2008.

[18] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Atrribute-Based Systems. In *Proc. of CCS'06*, New York, NY, USA, 2006.

[19] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *Proc. of EUROCRYPT'05*, Aarhus, Denmark, 2005.