Attribute-Based Conditional Proxy Re-Encryption with Chosen-Ciphertext Security

Conference Paper · December 2010

DOI: 10.1109/GLOCOM.2010.5684045 · Source: DBLP

Zhenfeng Zhang
Chinese Academy of Sciences
46 PUBLICATIONS **715** CITATIONS

Attribute-Based Conditional Proxy Re-Encryption with Chosen-Ciphertext Security

Jing Zhao
State Key Laboratory of Information Security
Graduate School of Chinese Academy of Sciences
Beijing, China
zhaojingsctmnp@hotmail.com

Dengguo Feng, Zhenfeng Zhang
State Key Laboratory of Information Security
Institute of Software, Chinese Academy of Sciences
Beijing, China
feng@is.iscas.ac.cn, zfzhang@is.iscas.ac.cn

Abstract—**Proxy re-encryption is a cryptographic primitive which enables a ciphertext encrypted under a delegator's public key to be translated into a ciphertext of a delegatee by a semi-trusted proxy. Conditional proxy re-encryption (CPRE) is a variant of proxy re-encryption which allows the delegator to control the delegation of decryption rights with certain conditional value. The existing CPRE schemes left an open problem about how to construct CCA-secure CPRE schemes supporting Boolean predicates over conditions. In this paper, we propose attribute-based CPRE (AB-CPRE) in which the delegator could implement attributed-based control on the delegation of decryption rights by setting conditions in the form of access structure and attribute set. AB-CPRE is suitable for applications where fine-grained control of the decryption delegation is necessary. We formalize definitions and security notions for AB-CPRE and prove that the proposed scheme is chosen-ciphertext secure under the 3-Quotient Decision Bilinear Diffie-Hellman (3-QDBDH) assumption.**

Keywords-conditional proxy re-encryption; attribute-based encryption; chosen-ciphertext security; bilinear pairing

I. INTRODUCTION

The notion of proxy re-encryption (PRE) was first introduced by Blaze et.al in 1998[11]. In a PRE scheme a ciphertext encrypted under Alice's public key could be translated into a ciphertext encrypted under Bob's public key by a semi-trusted proxy with a re-encryption key Alice assigned for Bob. The procedure could be denoted as $C(pk_A)\xrightarrow{rk_{A\to B}}C'(pk_B)$, while the plaintext and secret keys remain secure from the proxy. PRE has been proven useful in many applications such as access control in encrypted email forwarding[11], distributed file storage[12], digital rights management (DRM)[13][14], privacy preserving in anonymity revocation[15] and so on.

However, once the re-encryption key is settled on the proxy, Bob the delegatee is able to translate all the ciphertexts that he could access of Alice, putting delegation of her decryption right under the risk of abuse. For example, when Alice wants to forward to Bob one of her encrypted email stored in the distributed network storage, all the other personal files stored there should be kept secret from Bob as usual. In traditional PRE solution, the re-encryption key $rk_{A\to B}$ allows Bob to decrypt all the encrypted data. In such applications fine-grained control of the delegation of decryption right is indispensable.

To address this issue, Libert and Vergnaud proposed warrant-based PRE scheme with replayable chosen-ciphertext security[21], Tang introduced CCA-secure type-based PRE[18] and Weng et al. proposed conditional PRE[10]. The first three researches were independently presented. Names of the notions are different while the spirits are similar. Later the authors of the latter two papers proposed an efficient and more secure scheme[9] together and named the variant of PRE as conditional proxy re-encryption (CPRE). The main idea of CPRE could be simplified as $C(pk_A,c)\xrightarrow{rk_{A\xrightarrow{c}B}}C'(pk_B)$, where c denotes a condition value. Besides the above works, Chu et.al introduced their work on conditional proxy broadcast re-encryption[19]. Fang et al. did some interesting research on CCA-secure anonymous CPRE[20], which answering an open question about condition anonymity[10].

Nevertheless, another open problem[9,10] is left unsolved in the existing CPRE schemes: the flexible condition expression, which determines the fine-grained level of the delegation control. As long as the condition is expressed as one single value, Alice would have to set n re-encryption keys for Bob when she shares n types of encrypted data with him. When she wants to share a file with Bob and Catherine in the encrypted form, whose assigned re-encryption keys are of different condition, she also has to generate re-encryption keys with a new condition for them and encrypt the file in the same condition. In such situations, the improvement in the expressiveness and flexibility of the condition construction will remarkably improve the efficiency and practicability of the CPRE system.

Our Contribution

In this paper, we propose attribute-based CPRE (AB-CPRE) in which the conditions are expressed in the form of access structure[8][1] and attribute set. In AB-CPRE, the re-encryption process could be simplified as $C(pk_A,S)\xrightarrow{rk_{A\xrightarrow{AS}B}}C'(pk_B)$, where S is a set of descriptive attributes and AS is an access structure that specifies the access policy the delegator assigned to the delegatee. The re-encryption performs successfully if and only if S satisfies AS.

Supported by the National Natural Science Foundation of China under Grant No. 60803129; The National High-Tech Research and Development Plan of China under Grant Nos. 2007AA120404, 2007AA120405; The Next Generation Internet Business and Equipment Industrialization Program under Grant No. CNGI-09-03-03.

AB-CPRE achieves the condition expressive power by applying key-policy attribute-based encryption (KP-ABE) techniques[1] to implant the access policy into re-encryption key. The reason we choose KP-ABE rather than ciphertext policy ABE (CP-ABE)[1] is that specifying access policy in re-encryption key is more natural than in ciphertext, since the main idea of CPRE is for the delegator to control his decryption delegation. Ciphertext could be encrypted by another sender while the re-encryption key could be generated only by the delegator. Our scheme achieves CCA security by applying CHK transform[7] to the CPA-secure construction inspiring by Libert and Vergnaud's work[6]. By this means we answer the open problem about how to support "OR" and "AND" gates over conditions in an CCA-secure CPRE scheme[9,10].

We notice that Liang et al. presented researches on attribute-based PRE (ABPRE)[5]. The difference between ABPRE and AB-CPRE is that ABPRE enables re-encryption of ABE ciphertexts, simplified as $C(AS_A)\xrightarrow{rk_{S_A\to S_B}}C'(AS_B)$, while AB-CPRE enables re-encryption of traditional public key encryption ciphertexts. AB-CPRE focuses on introducing KP-ABE technique to express the conditions of re-encryption.

We prove the AB-CPRE scheme selective-set CCA-secure under the 3-Quotient Decision Bilinear Diffie-Hellman (3-QDBDH) assumption without random oracle.

II. Preliminaries

A. AB-CPRE Model

In a PRE scheme, a 2nd-level ciphertext is an original ciphertext and a 1st-level ciphertext is a transformed ciphertext. In an AB-CPRE scheme, the 2nd-level ciphertext is a public key encryption (PKE) ciphertext labeled with a set of attributes S and 1st-level ciphertext is a common PKE ciphertext.

An AB-CPRE scheme consists of nine algorithms as follows.

1) **SetUp**(κ) Given a security parameter κ as input, outputs the public parameters par.

2) **KeyGen**(par) Given the public parameters par, generates a public/private key pair (pk_i, sk_i) for some user U_i.

3) **AKeyGen**(par) The attribute key could be defined by an authority or by a user who wants to place attribute-based control on the delegation of his decryption rights. Therefore the attribute key could be published or kept secret according to the specific applications. The algorithm takes par as input to generate attribute key ak to enable attribute-based re-encryption.

4) **ReKeyGen**$(sk_i, ak, pk_j, AS_{ij})$ Given attribute key ak, U_i's secret key sk_i, U_j's public key pk_j, and an access structure AS_{ij} that U_i sets for user U_j, the algorithm generates a re-encryption key $rk_{i\to j}$ associated with AS_{ij}.

5) **Enc**$_2(pk_i, ak, m, S)$ On input of a message m, pk_i and ak, together with a set of attributes S, outputs a 2nd-level ciphertext U_i.

6) **Enc**$_1(pk_i, m)$ On input of a message m and pk_i, outputs a 1st-level ciphertext C_i.

7) **ReEnc**$(rk_{i\to j}, C_i)$ Given U_i's 2nd-level ciphertext C_i labeled with S, the re-encryption algorithm uses the re-encryption key $rk_{i\to j}$ associated with AS_{ij} to compute a valid 1st-level ciphertext C_j under pk_j if $S\in AS_{ij}$.

8) **Dec**$_2(C_i, sk_i)$ Given a 2nd-level ciphertext C_i and the secret key sk_i, outputs the message m or the error symbol $\perp$.

9) **Dec**$_1(C_i, sk_i)$ Given a 1st-level ciphertext C_i and the secret key sk_i, outputs the message m or the error symbol $\perp$.

B. Security Notions

Now we define the selective-set chosen-ciphertext security for AB-CPRE system, following the security definition for CPRE[6,9] and the selective-set model for KP-ABE[1]. We let the target public key and the target attribute set determined at the beginning of the game, as in [6] and [1]. The adversary is allowed to query oracles of **KeyGen**, **ReKeyGen**, **ReEnc**, **Dec**$_1$ with constraints except explicitly providing the oracle of **Dec**$_2$, which has been proved useless[6,9].

Definition 1. An AB-CPRE scheme is IND-ABCPRE-CCA secure at the second level if the probability

$$Adv_{\varepsilon,A}^{IND-2ABCPRE-CCA}(\kappa)$$

$$=\left|\Pr\left[\delta=\delta'\left|\begin{array}{l} par\leftarrow SetUp(\kappa), S^*\leftarrow AKeyGen(par)\\ (pk^*,sk^*)\leftarrow KeyGen(par)\\ (m_0,m_1,St)\leftarrow A_{find}^{\mathcal{O}_h,\mathcal{O}_c,\mathcal{O}_{rk},\mathcal{O}_{re},\mathcal{O}_{1d}}(par)\\ \delta\in_R\{0,1\}, C^*\leftarrow Enc_2(pk^*,m_\delta,S^*)\\ \delta'\leftarrow A_{guess}^{\mathcal{O}_h,\mathcal{O}_c,\mathcal{O}_{rk},\mathcal{O}_{re},\mathcal{O}_{1d}}(par,C^*,St)\end{array}\right.\right]-\frac{1}{2}\right|$$

is negligibly for any PPT adversary $\mathcal{A}$. In the notion, St is state information maintained by $\mathcal{A}$. It is required that $|m_0|=|m_1|$, and the following constraints are simultaneously satisfied:

1) For a public key pk_j generated by oracle $\mathcal{O}_C$, $\mathcal{A}$ cannot issue the query $\mathcal{O}_{rk}(pk^*, AS, pk_j)$ if $AS(S^*)=1$;

2) For a public key pk_j generated by oracle $\mathcal{O}_C$, $\mathcal{A}$ cannot issue the query $\mathcal{O}_{re}(pk^*, C^*, pk_j, AS)$ if $AS(S^*)=1$;

3) For a public key pk_j and a first level ciphertext C', $\mathcal{A}$ cannot query $\mathcal{O}_{1d}(C', pk_j)$ if $C'=\mathrm{Re}Enc(C^*, rk_{i^*\to j})$ and $AS(S^*)=1$ for AS that embedded in $rk_{i^*\to j}$.

C. Complexity Assumptions

The security of our second level ciphertext is based on 3-Quotient Decision Bilinear Diffie-Hellman (3-QDBDH) assumption which is proven hard in [17].

Definition 2. The 3-QDBDH problem in group (G, G_T) is, given a tuple $(g, g^a, g^{a^2}, g^{a^3}, g^b, Z)\in G^5\times G_T$ with unknown $a, b\ {}_R\!\in Z_p^*$, to decide whether $Z=e(g,g)^{b/a}$.

Definition 3. A polynomial-time adversary $\mathcal{A}$'s advantage in solving the 3-QDBDH problem in (G, G_T) is defined as

$$Adv_B^{3-\mathrm{QDBDH}}\triangleq\Pr[B(g,g^a,g^{a^2},g^{a^3},g^b)=e(g,g)^{b/a}]$$

where $a,b \in_R Z_p^*$ and the random bits consumed by $\mathcal{A}$. The $(t,\varepsilon)-3-$QDBDH assumption holds in (G,G_T) if no t-time adversary $\mathcal{A}$ has advantage at least ε in solving the 3-QDBDH problem in (G,G_T).

D. One-Time Signatures

A one-time signature[7] Sig=$(\mathcal{G},\mathcal{S},\mathcal{V})$ consists of three algorithms. $\mathcal{G}$ generates a one-time key pair (ssk,svk). For any message m, $\mathcal{V}(svk,\sigma,m)$ equals to 1 iff $\sigma=\mathcal{S}(ssk,m)$ and equals to 0 otherwise. As in [6], we need strongly unforgeable one-time signature which makes sure that no PPT adversary can create a new signature for a previously signed message.

Definition 4. Sig=$(\mathcal{G},\mathcal{S},\mathcal{V})$ is a strong one-time signature if for any PPT forger $\mathcal{F}$ the probability $Adv^{OTS}=\Pr[(ssk,svk) \leftarrow \mathcal{G}(\kappa);(M,St)\leftarrow F(svk);\sigma\leftarrow \mathcal{S}(ssk,M);(M',\sigma')\leftarrow F(M,\sigma, svk,St):\mathcal{V}(svk,M',\sigma')=1\wedge(M',\sigma')\neq(M,\sigma)]$ is negligible, where St denotes the state information maintained by $\mathcal{F}$ between stages.

III. PROPOSED AB-CPRE SCHEME

In this section we first describe our AB-CPRE scheme with CPA security and then present the CCA-secure one. The fine-grained expression of condition is based on the access structure developed in the KP-ABE scheme[1]. We briefly review it in the first subsection. Our construction uses token-based encryption paradigm as a tool which was introduced into PRE construction by Green and Ateniese[2] and proved quite effective in several recent PRE schemes[3-5]. Our main idea is to combine the token-based technique with the large universe construction of KP-ABE to construct a CPA-secure AB-CPRE scheme. Then we will apply CHK transform[7] to achieve CCA security, inspired by Libert and Vergnaud's work in [6].

A. Construction for Access structure

We introduce the construction for access structure in KP-ABE[1] into AB-CPRE to construct re-encryption key. The re-encryption key is constructed by an access tree $\mathcal{T}$ in which each interior node x is a threshold gate and the leaves are associated with attributes. The threshold value is $k_x \in (0,num_x]$ where num_x is x's children number. For each leaf node a function $att(x)$ is defined to denote the attribute associated with the leaf. If a set of attributes S satisfies the access tree $\mathcal{T}$, denote it as $\mathcal{T}_x(S)=1$. $\mathcal{T}_x(S)$ is a recursive algorithm which returns 1 iff at least k_x children return 1. When x is a leaf node, $\mathcal{T}_x(S)$ returns 1 iff $att(x)\in S$. More details of the access tree can be found in [1].

B. Construction of CPA-Secure AB-CPRE

We describe the CPA-secure AB-CPRE scheme as follows.

SetUp(κ) : Let κ be the security parameter, the algorithm outputs $par=(p,G,G_1,e,g,g_1)$ as the public parameters. G is a bilinear group of prime order p. g is a generator and g_1 is a random element of G. $e:G\times G\rightarrow G_1$ denotes the bilinear map. The Lagrange coefficient $\Delta_{i,S}$ for $i\in Z_p$ and a set of elements S in Z_p is defined as $\Delta_{i,S}(x)=\prod_{j\in S,j\neq i}\frac{x-j}{i-j}$.

KeyGen(par) : User U_i randomly picks $x_i\in Z_p$, sets his public key as $pk_i=g^{x_i}$ and secret key as $sk_i=x_i$.

AKeyGen(par) : Let $N=\{1,2,...,n+1\}$. Choose $t_k \in_R G$, $k\in N$. Define function T: $T(X)=g_1^{X^n}\prod_{k=1}^{n+1}t_k^{\Delta_{k,N}(X)}$. Function T can be viewed as a function $g_1^{X^n}g^{h(X)}$ for some n degree polynomial h. The attribute key $ak=\{t_k\}_{k\in N}$ is published.

ReKeyGen$(sk_i,ak,pk_j,\mathcal{T}_{ij})$: The access tree $\mathcal{T}_{ij}$ following the access policy U_i sets for U_j is constructed as described in subsection III.A. The algorithm proceeds in three phases.

At first we choose a polynomial q_x for each node x in $\mathcal{T}_{ij}$ in a top-down manner starting from the root node r. q_x is constructed as follows.

Set the degree d_x of q_x as $d_x=k_x-1$. For the root node r, picks $w_{ij} \in_R Z_p$ to set $q_r(0)=w_{ij}/x_i$ and d_r other points of q_r are also randomly chosen to define q_r completely. For any other node x, $q_x(0)=q_{parent(x)}(index(x))$, d_x other points of q_x are randomly chosen to complete the definition.

In the second stage we compute a pair of values (D_x,R_x) for each leaf node x of the access tree $\mathcal{T}_{ij}$ as:

$$D_x=g_1^{q_x(0)}\bullet T(k)^{r_x},k=att(x);\ R_x=pk_i^{r_x}.$$

In the last phase we set the re-encryption key as:

$$rk_{i\rightarrow j}=(rk_{i\rightarrow j}^1=(\{D_x\},\{R_x\}),rk_{i\rightarrow j}^2=C_{pk_j}(w_{ij})).$$

$C_{pk_j}(w_{ij})$ is an encryption of message w_{ij} under pk_j.

Enc$_2$ (pk_i,ak,m,S): To encrypt a message m under pk_i with a set of attributes S, choose a random value $s\in Z_p$ and output the 2nd-level ciphertext as

$$C_i=(S,\ C_1=M\bullet e(g,g_1)^s,C_2=pk_i^s,\{E_k=T(k)^s\}_{k\in S}).$$

Enc$_1$ (pk_i,m): To encrypt a message m under pk_i, randomly choose values $s,w\in Z_p$ and compute the 1st-level ciphertext with token w:

$$C_i=(C_1=M\bullet e(g,g_1)^s,C_2=e(g,g_1)^{w\bullet s},C_{pk_i}(w)).$$

ReEnc$(rk_{i\rightarrow j},C_i)$: On input of a 2nd-level ciphertext under pk_i : $C_i=(S,\ C_1,C_2,\{E_k\}_{k\in S})$, the algorithm compute the 1st-level ciphertext under pk_j as follows.

First we invoke a recursive algorithm defined in [1] denoted as $\text{DecryptNode}(E,D,x)$ to process E_k, the attributes associated part of ciphertext, with (D_x, R_x), the access tree associated part of re-encryption key. See details of the adopted algorithm in [1].

If the node x is a leaf node and $k = att(x)$, then:

$$\text{DecryptNode}(E,D,x) = \begin{cases} \dfrac{e(D_x, C_2)}{e(R_x, E_k)} = e(g_1, pk_i)^{s \bullet q_x(0)} & \text{if } k \in S \\ \perp \text{otherwise} \end{cases}.$$

If the node x is a non-leaf node, then for all child nodes z of x, the algorithm calls $\text{DecryptNode}(E,D,z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of z such that $F_z \neq \perp$. If no such set exists then the node is not satisfied and the function returns $\perp$. Otherwise we obtain the following result using polynomial interpolation:

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x(0)}}, \text{ where}_{S'_x = \{index(z): z \in S_x\}}^{i = index(z)} \\ &= \prod_{z \in S_x} e(g, g_1)^{s \bullet q_x(i) \bullet \Delta_{i,S'_x(0)}} \\ &= e(g, g_1)^{s \bullet q_x(0)} \end{aligned}.$$

In our scheme the re-encryption algorithm first calls the recursive function $\text{DecryptNode}(E,D,x)$ on the root of the access tree as:

$$\begin{aligned} C_2' &= \text{DecryptNode}(\{E_k\}, (\{D_x\}, \{R_x\}), r) = e(pk_i, g_1)^{s \bullet q_r(0)} \\ &= e(g^{x_i}, g_1)^{s \bullet (w_{ij}/x_i)} = e(g, g_1)^{s \bullet w_{ij}} \end{aligned}.$$

Sets $C_1' = C_1$, $C_3' = rk^2_{i \to j}$. The algorithm outputs the re-encrypted ciphertext under pk_j as:

$$C_j = (C_1' = M \bullet e(g, g_1)^s, C_2' = e(g, g_1)^{s \bullet w_{ij}}, C_3' = C_{pk_j}(w_{ij})).$$

Dec₂(C_i, sk_i) **:** On input of a second level ciphertext

$$C_i = (S,\ C_1 = M \bullet e(g, g_1)^s, C_2 = pk_i^s, \{E_k = T(k)^s\}_{k \in S}),$$

the algorithm outputs $M = C_1 / e(g_1, C_2)^{1/x_i}$.

Dec₁(C_i, sk_i) **:** On input of a first level ciphertext

$$C_i = (C_1 = M \bullet e(g, g_1)^s, C_2 = e(g, g_1)^{w \bullet s}, C_{pk_i}(w)),$$

the algorithm decrypts $C_{pk_j}(w_{ij})$ using sk_i to obtain w and outputs the plaintext as $M = C_1 / C_2^{1/w}$.

C. Construction of CCA-Secure AB-CPRE

Through applying the CHK methodology[7] to the CPA-secure scheme (denoted by S_{CPA} in the following), we construct the CCA-secure AB-CPRE scheme as follows.

SetUp(κ) : Let κ be the security parameter, the algorithm outputs $par = (p, G, G_1, e, g, g_1, g_2, \text{Sig}, H)$ the same as in S_{CPA}, with the addition of a random element g_2 of G, a hash function $H: \{0,1\}^* \to G_1$ and Sig=$(\mathcal{G}, \mathcal{S}, \mathcal{V})$ denotes a strongly unforgeable one-time signature scheme.

KeyGen(par) and **AKeyGen**(par) : The same as in S_{CPA}.

ReKeyGen$(sk_i, ak, pk_j, \mathcal{T}_{ij})$:

First we choose a polynomial q_x for each node x in $\mathcal{T}_{ij}$ the same way as in S_{CPA}. Notice that for the root node r, picks $w_{ij} \in_R Z_p$, sets $q_r(0) = w_{ij} / x_i$.

Second we compute a tuple of values (D_x^1, D_x^2, R_x) for each leaf node x of the access tree $\mathcal{T}_{ij}$ as:

$$D_x^1 = g_1^{q_x(0)} \bullet T(k)^{r_x},\ D_x^2 = g_2^{q_x(0)} \bullet T(k)^{r_x},\ k = att(x), R_x = pk_i^{r_x}.$$

At last we set the re-encryption key as

$$rk_{i \to j} = (rk^1_{i \to j} = (\{D_x^1\}, \{D_x^2\}, \{R_x\}), rk^2_{i \to j} = C^{CCA}_{pk_j}(w_{ij})),$$

$C^{CCA}_{pk_j}(w_{ij})$ is a CCA-secure encryption of w_{ij} under pk_j.

Enc₂(pk_i, ak, m, S): To encrypt a message m under pk_i and a set of attributes S, it first selects a one-time signature keypair (ssk, svk) randomly from $\mathcal{G}$ and sets $C_3 = svk$.

Then it chooses a random value $s \in Z_p$ and computes

$$C_1 = M \bullet e(g, g_1)^s,\ C_2 = pk_i^s,\ \{E_k = T(k)^s\}_{k \in S},$$
$$C_4 = (g_1^{svk} \bullet g_2)^s,\ C_5 = H(C_2, \{E_k\}_{k \in S}).$$

At last it generates a one-time signature on (C_1, C_4, C_5): $\sigma = \mathcal{S}(ssk, (C_1, C_4, C_5))$.

The resulted second level ciphertext is

$$C_i = (S,\ C_1, C_2, \{E_k\}_{k \in S}, C_3, C_4, C_5, \sigma).$$

Enc₁(pk_i, m) : To encrypt a message m under pk_i, it first selects a one-time signature key pair (ssk, svk) randomly from $\mathcal{G}$ and sets $C_3 = svk$.

Then it chooses random values $s, w \in Z_p$ and computes

$$C_1 = M \bullet e(g, g_1)^s,\ C_2 = e(g, g_1)^{s \bullet w},\ C_2' = e(g, g_2)^{s \bullet w},$$
$$C_4 = (g_1^{svk} \bullet g_2)^s,\ C_5 = H(C_2, C_2'),\ C_6 = C^{CCA}_{pk_i}(w).$$

At last it generates a one-time signature on (C_1, C_4, C_5): $\sigma = \mathcal{S}(ssk, (C_1, C_4, C_5))$.

The resulted first level ciphertext is

$$C_i = (C_1, C_2, C_2', C_3, C_4, C_5, C_6, \sigma).$$

ReEnc$(rk_{i \to j}, C_i)$: On input of a second level ciphertext $C_i = (S,\ C_1, C_2, \{E_k\}_{k \in S}, C_3, C_4, C_5, \sigma)$, the algorithm first checks the validity of the ciphertext:

$$e(C_4, pk_i) = e(C_2, g_1^{C_3} \bullet g_2),\ \mathcal{V}(C_3, \sigma, (C_1, C_4, C_5)) = 1.$$

If the ciphertext is well-formed, calls the recursive function $\text{DecryptNode}(E, D, R, x)$ on the root of T_{ij} and compute:

$$C_2' = \text{DecryptNode}(\{E_k\},\{D_x^1\},\{R_x\},r) = e(pk_i,g_1)^{s\cdot q_r(0)} = e(g,g_1)^{s\cdot w_{ij}},$$
$$C_2'' = \text{DecryptNode}(\{E_k\},\{D_x^2\},\{R_x\},r) = e(pk_i,g_2)^{s\cdot q_r(0)} = e(g,g_2)^{s\cdot w_{ij}}.$$

Sets $C_1' = C_1$, $C_3' = C_3$, $C_4' = C_4$, $C_5' = C_5$, $C_6' = rk^2_{i\to j}$, it outputs the re-encrypted ciphertext under pk_j as:

$$C_j = (C_1', C_2', C_2'', C_3', C_4', C_5', C_6', \sigma).$$

$\mathbf{Dec_2}(C_i, sk_i)$: On input of a second level ciphertext $C_i = (S,\ C_1, C_2, \{E_k\}_{k\in S}, C_3, C_4, C_5, \sigma)$, the algorithm first checks the validity:

$$e(C_4, pk_i) = e(C_2, g_1^{C_3}\cdot g_2),\ \mathcal{V}(C_3,\sigma,(C_1,C_4,C_5)) = 1.$$

If the ciphertext is well-formed, the algorithm outputs

$$M = C_1 / e(g, C_2)^{1/x_i}.$$

$\mathbf{Dec_1}(C_i, sk_i)$: On input of a first level ciphertext $C_i = (C_1, C_2, C_2', C_3, C_4, C_5, C_6, \sigma)$, the algorithm first decrypts w from C_6 with sk_i. Then it checks the validity:

$$e(C_4, g^w) = C_2^{C_3}\cdot C_2',\ \mathcal{V}(C_3,\sigma,(C_1,C_4,C_5)) = 1.$$

If the ciphertext is well-formed, the algorithm outputs

$$M = C_1 / C_2^{1/w}.$$

IV. Security Analysis

Theorem 1. The AB-CPRE scheme is CCA-secure at the second level under the 3-QDBDH assumption, assuming the strong unforgeability of the one-time signature.

Proof. The proof in short can be found in appendix A for the lack of space. The details can be found in the full version.

V. Conclusion

We propose AB-CPRE to extend the expressiveness and flexibility of the conditions which expressed as a single value in existing CPRE schemes. By construction upon access tree and attribute set, the delegator is able to implement attribute-based access control over his encrypted data. Therefore AB-CPRE is more useful in broader applications. We prove our scheme secure against chosen-ciphertext attack under the 3-QDBDH assumption in the selective-set model for AB-CPRE.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments. The first author would like to thank Liwu Zhang and Rui Xue for helpful discussions on the topic.

References

[1] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*. pp. 89-98, 2006.

[2] M. Green and G. Ateniese. Identity-based proxy re-encryption. In *ACNS 2007*, volume 4521 of LNCS, pp. 288-306, 2007.

[3] J. Shao and Z. Cao. CCA-secure proxy re-encryption without pairings. In *PKC 2009*, volume 5443 of LNCS, pp. 357-376, 2009.

[4] R. H. Deng, J. Weng, S. Liu, and K. Chen. Chosen-ciphertext secure proxy re-encryption without pairings. In *CANS 2008*, volume 5339 of LNCS, pp. 1-17, 2008.

[5] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute-based proxy re-encryption with delegating capabilities. In *ACM ASIACCS 2009*, pp. 276-286, 2009.

[6] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *PKC 2008*, volume 4939 of LNCS, pp. 360-379, 2008.

[7] R. Canetti, S. Halevi, J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT 2004*, volume 3027 of LNCS, pp. 207-222, 2004.

[8] A. Beimel. Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[9] J. Weng, Y.Yang, Q.Tang, R. H. Deng, and F. Bao. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *ISC 2009*, volume 5735 of LNCS, pp. 151-166, 2009.

[10] J. Weng, R. H. Deng, C. Chu, X. Ding, and J. Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *ACM ASIACCS 2009*, pp. 322-332, 2009.

[11] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT 1998*, volume 1403 of LNCS, pp. 127-144, 1998.

[12] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *ACM NDSS 2005*, pp. 29-43, 2005.

[13] T. Smith. DVD Jon: buy DRM-less tracks from Apple iTunes, 2005. Available: http://www.theregister.co.uk/2005/03/18/itunes_pymusique/.

[14] G. Taban, A. A. Cirdenas and V. D. Gligor. Towards a secure and interoperable DRM architecture. In *ACM DRM 2006*, pp. 69-78, 2006.

[15] S. Suriadi, E. Foo, and J. Smith. Conditional privacy using re-encryption. In IFIP International Workshop on Network and System Security, 2008.

[16] D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*. volume 2139 of LNCS, pp. 229-231, 2001.

[17] Y. Dodis, A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC 2005*. volume 3386 of LNCS, pp. 416-431, 2005.

[18] Q. Tang. Type-based proxy re-encryption and its construction. In *INDOCRYPT 2008*. volume 5365 of LNCS, pp. 130-144, 2008.

[19] C.K. Chu, J. Weng, S.S.M. Chow, J. Zhou, R.H. Deng. Conditional proxy broadcast re-encryption. In *ACISP 2009*, volume 5594 of LNCS, pp. 327-342, 2009.

[20] L. Fang, W. Susilo, and J. Wang. Anonymous conditioinal proxy re-encryption without random oracle. In *PoveSec 2009*. volume 5848 of LNCS, pp. 47-60, 2009.

[21] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. Available: http://hal.inria.fr/inria-00339530/en/. This is the extended version of [6].

Appendix

A. *Proof for Theorem 1*

Proof. We construct a simulator $\mathcal{B}$ to play the 3-QDBDH game with a non-negligible advantage out of a successful CCA adversary $\mathcal{A}$ through the IND-ABCPRE-CCA game as follows.

At the beginning the challenger flips a fair binary coin μ outside of $\mathcal{B}$'s view. When $\mu = 0$, the challenger sets

$$(A_{-1}, A_1, A_2, B, Z) = (g^{1/a}, g^a, g^{(a^2)}, g^b, e(g,g)^{b/a^2}).$$

Otherwise the challenger sets $(A_{-1}, A_1, A_2, B, Z) = (g^{1/a}, g^{a}, g^{(a^2)}, g^{b}, e(g,g)^{z})$ for random $a, b, z \in Z_q$. We call HU the set of honest parties including the target user $i*$, and CU the set of corrupted parties.

Init. The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses the target user $i*$ and the set of attributes $S*$ it wishes to challenged upon. $\mathcal{B}$ runs **SetUp**(κ), generates a one-time signature key pair $(ssk*, svk*)$, let $g_1 = A_1^{\alpha}$ and $g_2 = A_1^{-\alpha \bullet svk*} \bullet A_2^{\beta}$ and gives the resulted public parameters par to $\mathcal{A}$.

Phase1. $\mathcal{A}$ makes the following queries:

1) uncorrupted key generation query $\langle i \rangle$*:* for $i \in HU$, when $i = i*$, $\mathcal{B}$ picks $x_i * {}_R\in Z_p$, sets $pk_i * = A_2^{x_i *}$. When $i \in HU \setminus \{i*\}$, $\mathcal{B}$ randomly chooses $x_i \in Z_p$, sets $pk_i = A_1^{x_i}$.

2) corrupted key generation query $\langle i \rangle$*:* for $i \in CU$, $\mathcal{B}$ picks $x_i \;{}_R\in Z_p$, sets $pk_i = g^{x_i}$ and sends (pk_i, x_i) to $\mathcal{A}$.

3) attribute key generation query: $\mathcal{B}$ randomly chooses a n degree polynomial $f(X)$ and computes an n degree polynomial $u(X)$ as: sets $u(X) = -X^n$ for all $X \in S*$ and $u(X) \neq -X^n$ for other X.

Then $\mathcal{B}$ sets $t_k = g_1^{u(k)} A_2^{f(k)}$, $k \in N$. Note that we have $T(k) = g_1^{k^n + u(k)} A_2^{f(k)}$, $k \in N$. $\mathcal{B}$ sends $ak = \{t_k\}_{k \in N}$ to $\mathcal{A}$.

4) re-encryption key generation query $\langle pk_{i*}, pk_j, \mathcal{T}_{i*j} \rangle$*:*

a) For a delegatee $j \in CU$, if $\mathcal{T}_{i*j}(S*) = 1$, $\mathcal{B}$ returns $\perp$ according to the constraints. Otherwise $\mathcal{B}$ invoke the procedure $PolyUnsat(\mathcal{T}_x, S, g^{\lambda_x})$ from [1] to assign a d_x degree polynomial Q_x for every node x in $\mathcal{T}_{i*j}$. In our simulation, $\mathcal{B}$ runs $PolyUnsat(\mathcal{T}_{i*j}, S*, A_{-1}^{w_{i*j}/x_{i*}})$. Note that $q_r(0) = w_{i*j} / a x_{i*}, w_{i*j} \;{}_R\in Z_p$. Let $Q_x(\bullet) = q_x(\bullet)/a$, there is $g_1^{Q_x(0)} = g^{\alpha q_x(0)}, g_2^{Q_x(0)} = (g^{-\alpha \bullet svk*} \bullet A_1^{\beta})^{q_x(0)}$. Then $\mathcal{B}$ computes $\{D_x^1, D_x^2, R_x\}$ corresponding to each leaf node x in $\mathcal{T}_{i*j}$ as follows: let $k = att(x)$. If $k \in S*$,

$$D_x^1 = g_1^{Q_x(0)} T(k)^{r_x}, \; D_x^2 = g_2^{Q_x(0)} T(k)^{r_x}, \; R_x = pk_{i*}^{r_x}.$$

If $k \notin S*$, let $g_3 = A_2^{Q_x(0)}, g_4 = pk_{i*}^{Q_x(0)}$,

$$D_x^1 = g_3^{-f(k)/(k^n + u(k))} (g_1^{k^n + u(k)} A_2^{f(k)})^{r_x'}$$

$$D_x^2 = g_3^{-f(k)/(k^n + u(k))} (g_2^{k^n + u(k)} A_2^{f(k)})^{r_x'}$$

$$R_x = g_4^{-1/(k^n + u(k))} pk_{i*}^{r_x'}.$$

Notice that if we set $r_x = r_x' - q_x(0) / (k^n + u(k))$, there is

$$D_x^1 = g_3^{-f(k)/(k^n + u(k))} (g_1^{k^n + u(k)} A_2^{f(k)})^{r_x'} = g_1^{Q_x(0)} T(k)^{r_x},$$

$$D_x^2 = g_3^{-f(k)/(k^n + u(k))} (g_2^{k^n + u(k)} A_2^{f(k)})^{r_x'} = g_2^{Q_x(0)} T(k)^{r_x},$$

$$R_x = g_4^{-1/(k^n + u(k))} pk_{i*}^{r_x'} = pk_{i*}^{r_x}.$$

$\mathcal{B}$ returns

$$rk_{i* \to j} = (rk^1_{i* \to j} = \{D_x^1, D_x^2, R_x\}, rk^2_{i* \to j} = C^{CCA}_{pk_j}(w_{i*j})).$$

b) For a delegatee $j \in HU \setminus \{i*\}$, if $\mathcal{T}_{i*j}(S*) = 1$, $\mathcal{B}$ runs $PolySat(T_{i*j}, S*, A_{-1}^{w_{i*j}/x_{i*}})$, $w_{i*j} \;{}_R\in Z_p$. Otherwise $\mathcal{B}$ runs $PolyUnsat(T_{i*j}, S*, A_{-1}^{w_{i*j}/x_{i*}})$, $w_{i*j} \;{}_R\in Z_p$. Note that $q_r(0) = w_{i*j} / a x_{i*}$. Let $Q_x(\bullet) = q_x(\bullet)$. $\mathcal{B}$ sets $\{D_x^1, D_x^2, R_x\}$ corresponding to each leaf node x in $\mathcal{T}_{i*j}$ as the procedures defined above. $\mathcal{B}$ returns

$$rk_{i* \to j} = (rk^1_{i* \to j} = \{D_x^1, D_x^2, R_x\}, rk^2_{i* \to j} = C^{CCA}_{pk_j}(w_{i*j})).$$

5) re-encryption query $\langle pk_{i*}, pk_j, C_{i*}, \mathcal{T}_{i*j} \rangle$ and 1st-level decryption query $\langle pk_j, C_j \rangle$ can be simulated with the above key simulation in a way similar as PRE security proofs in [6], [9],etc. For the lack of space the detailed proof can be found in the full version.

Challenge. When $\mathcal{A}$ decides that the first phase is over, it chooses messages (M_0, M_1). At this stage, $\mathcal{B}$ flips a coin $d \in \{0,1\}$ and sets the challenge ciphertext as:

$$C_i = (S, C_1 = M_d \bullet Z, C_2 = B^{x_i *}, \{E_k = B^{f(k)}\}_{k \in S}), C_3 = svk,$$
$$C_4 = B^{\beta}, C_5 = H(C_2, E_k), \sigma = S(ssk, (C_1, C_4, C_5))$$

If $\mu = 0$ then $Z = e(g,g)^{b/a^2}$. Let $s = b/a^2$, then there is

$$e(g,g)^{s} = e(g,g)^{b/a^2}, pk_{i*}^{s} = A_2^{x_i * \bullet (b/a^2)} = B^{x_i *},$$
$$\{E_k = T_i(k)^{s}\}_{k \in S} = A_2^{t_{ik} \bullet (b/a^2)} = B^{t_{ik}},$$
$$C_4 = A_1^{\alpha(svk - svk*)} A_2^{\beta} = A_2^{\beta \bullet (b/a^2)} = B^{\beta}.$$

Therefore, the ciphertext is a valid random encryption of message M_d.

If $\mu = 1$, then $Z = e(g,g)^{z}$, $C_1 = M_d \bullet e(g,g)^{z}$. In this case C_1 is a random element of G_1 from the adversary's view, leaking no information about M_d.

Phase 2. $\mathcal{B}$ acts exactly the same as it did in Phase 1.

Guess. $\mathcal{A}$ returns a guess $d' \in \{0,1\}$. If $d' = d$, $\mathcal{B}$ outputs $\mu = 0$ to indicate that it was given a valid 3-QDBDH tuple. Otherwise, $\mathcal{B}$ outputs $\mu = 1$ to indicate a random tuple.

When $\mu = 1$, $\mathcal{A}$ obtains no information about d from a random element, we have $\Pr[\mu' = \mu | \mu = 1] = 1/2$; when $\mu = 0$, $\mathcal{A}$ gets a valid encryption of M_d. Let $\mathcal{A}$'s advantage is ε, we have $\Pr[\mu' = \mu | \mu = 0] = 1/2 + \varepsilon$.

The overall advantage of the simulator in 3-QDBDH game is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 1] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 0] - \frac{1}{2} = \frac{1}{2}\varepsilon$. □