The Computer Journal Advance Access published November 12, 2015

© The British Computer Society 2015. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
doi:10.1093/comjnl/bxv100

A Key-Policy Attribute-Based Proxy Re-Encryption Without Random Oracles

Chunpeng Ge[1], Willy Susilo[2]*, Jiandong Wang[1], Zhiqiu Huang[1], Liming Fang[1] and Yongjun Ren[3]

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China*
[2]*Centre for Computer and Information Security Research (CCISR), School of Computing and Information Technology, University of Wollongong, Wollongong, Australia*
[3]*School of Computer and Software, Nanjing University of Information Science, Nanjing, China*
**Corresponding author: wsusilo@uow.edu.au*

A conditional proxy re-encryption (CPRE) scheme enables the proxy to convert a ciphertext from Alice to Bob, if the ciphertext satisfies one condition set by Alice. To improve the issue of more fine-grained on the condition set, Fang, Wang, Ge and Ren proposed a new primitive named Interactive conditional PRE with fine grain policy (ICPRE-FG) in 2011, and left an open problem on how to construct CCA-secure ICPRE-FG without random oracles. In this paper, we answer this open problem affirmatively by presenting a new construction of CCA-secure key-policy attribute-based PRE (KP-ABPRE) without random oracles. In this paper, we enhance the security model of Fang's ICPRE-FG scheme by allowing the adversary to make some extra queries, which do not help them win the game trivially. Finally, we present a CCA-secure KP-ABPRE without random oracles under the 3-weak decisional bilinear Diffie–Hellman inversion(3-wDBDHI) assumption.

Keywords: conditional proxy re-encryption; attribute-based proxy re-encryption; without random oracles

Received 15 March 2015; revised 9 October 2015
Handling editor: Pil Lee

1. INTRODUCTION

Attribute-based encryption (ABE) [1,2] was first introduced by Sahai and Waters. In an ABE, a user's key and ciphertext are labeled with a set of descriptive attributes and a particular key can decrypt a particular ciphertext only if there is a match between the attributes of ciphertext and the user's key. ABE is divided into two categories: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE). In a KP-ABE system, secret keys are associated with access policies and ciphertext are encrypted under attributes, while CP-ABE is complementary. In this paper, we deal with the case of KP-ABE.

We use the medical consultation system (MCS) to illustrate the application of our key-policy attribute-based proxy re-encryption (KP-ABPRE) scheme. Consider the following scenario in an MCS. In MCS, a patient prefers to find doctors with the following attributes to remedy his gastritis problem. The attributes are denote as $\mathsf{I}_1 =$ ('gastritis', 'Consultant', 'Registrar', 'Downtown of Hongkong'). The patient encrypts his medical record under I_1 and then uploaded to the system. The system then forwards the ciphertext to doctor Alice to in charge of this case. On receiving the ciphertext of the patient's medical record, Alice may need another doctor Bob, that satisfies the access tree $\mathcal{T} =$ ('gastritis' $\bigwedge$ ('Senior Registrar' $\bigvee$ 'Registrar') $\bigwedge$ Location: 'Hongkong'), together with him to diagnose the patient's illness. The ciphertext needs to be encrypted under Bob's public key, prior to sending it to him. In a traditional public key encryption system, Alice has to decrypt the ciphertext to recover the data first, and then encrypt the data using Bob's public key. If Alice needs to consult with N doctors, that satisfy different requirements $\mathcal{T}_2, \mathcal{T}_3, \ldots, \mathcal{T}_N$, Alice is required to perform N decryptions and N new encryptions. This might not be desirable as Alice's workload is linear in N. The situation becomes worse as she should be on-line during each update process.

To solve the above problem, we may leverage the notion of proxy re-encryption (PRE). PRE was first introduced by Blaze *et al.* [3], whereby a semi-trusted proxy can transform a ciphertext from one key to another of the same message

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

TABLE 1. Comparison with [6–8].

Schemes	Public/Priavte key size	Ciphertext/ Re-key size	Re-encryption cost	CCA security/ without RO
CP-ABPRE[6]	$\mathcal{O}(U)/\mathcal{O}(U)$	$\mathcal{O}(U)/\mathcal{O}(U)$	$\mathcal{O}(U)\cdot c_p$	$\times/\surd$
CP-ABPRE[7]	$\mathcal{O}(1)/\mathcal{O}(A)$	$\mathcal{O}(f)/\mathcal{O}(A)$	$\mathcal{O}(A)\cdot(c_p+c_e)$	$\surd/\times$
ICPRE-FG[8]	$\mathcal{O}(1)/\mathcal{O}(1)$	$\mathcal{O}(w)/\mathcal{O}(t)$	$\mathcal{O}(w)\cdot c_p+\mathcal{O}(t)\cdot c_e$	$\surd/\times$
Our KP-PRE	$\mathcal{O}(1)/\mathcal{O}(1)$	$\mathcal{O}(w)/\mathcal{O}(t)$	$\mathcal{O}(w)\cdot c_p+\mathcal{O}(t)\cdot c_e$	$\surd/\surd$

without relying on trusted paries. In a PRE scheme, the proxy only needs a re-encryption key to convert the ciphertext without learning any information of the underlying message. To facilitate flexible delegation, Weng *et al.* [4] introduced the notion of *conditional* PRE (CPRE), whereby only ciphertext satisfying one certain condition set by Alice can be transformed by the proxy. CPRE has found many applications, such as encrypted email forwarding [3], secure distributed file systems [5] and outsourced filtering of encrypted spam [5].

Furthermore, Liang *et al.* [6] introduced the notion of CP-ABPRE, which use PRE in the attribute-based setting. In their scheme, access policy is represented as AND gates on positive and negative attributes. Liang *et al.* [7] proposed a ciphertext-policy attribute-based proxy encryption, which support any monotonic access formula. Fang *et al.* [8] proposed a new notion called interactive PRE with fine grain policy, whereby ciphertext encrypted with conditions W can be re-encrypted by the proxy using the CPRE key under the access structure $\mathcal{T}$ if and only if $\mathcal{T}(W)=1$. However, the constructions in [7] and [8] are both proved in the random oracle model.

We here compare our scheme with previous ABPRE schemes in terms of public/private size, re-encryption key size, ciphertext size, re-encryption cost and security model in Table 1. Let t be the size of an access tree and w be the size of attributes used in the encryption algorithm in our scheme and Fang's scheme [8], f be the size of an access formula and A be the number of attributes on a user's private key in [7], U be the number of all attributes defined in the system. Besides, c_e and c_p denote the computational cost of an exponentiation and a bilinear pairing.

Remark 1. In our scheme, the ciphertext size and re-encryption key size are almost the same as the ciphertext size and private key size in the basic ABE scheme [9], which were influenced by the number of attributes and the size of the access tree, as the basic ABE scheme [9] is used in our conditional re-encryption key generation and ciphertext generation algorithm. However, in our scheme, the private key is a random element in Z_p^* and the public key is a couple of elements in G_1, G_2, Z_p. So the private key and public key size are only $\mathcal{O}(1)$, the re-encryption key size is $\mathcal{O}(t)$ and the ciphertext size is $\mathcal{O}(\omega)$.

Motivation. Although the existing notion of interactive PRE with fine grain policy (ICPRE-FG) scheme explores the application of CPRE, there are remaining interesting open problems left in [8]. The former PRE scheme with key policy achieves security due to [8] is unfortunately only secure in the random oracles. Unfortunately, a proof in the random oracle model can only serve as a heuristic argument and admittedly uses quite contrived constructions. As an evidence, it has been shown in [10] that the instantiation of a random oracle model could lead to an insecure scheme. Furthermore, some queries in the former security model were also neglected. As a result, the former security model fails to capture the adversary behaviour, as these queries do not help the adversary to win the game trivially. Thus, it is desirable to propose a KP-ABPRE without random oracles under a more rigorous security model.

1.1. Our contribution

This paper aims to resolving the aforementioned open problems. In this paper, we first review the security model of previous ICPRE-FG scheme, then we enhance the security model. Finally, we propose a KP-ABPRE scheme which is CCA-secure under the 3-weak decisional bilinear Diffie–Hellman inversion assumption without random oracles.

1.2. Paper organization

The rest of the paper is organized as follows. In Section 2, we provide the definitions and complexity assumption. In Section 3, we present our KP-ABPRE scheme without random oracles. Section 4 concludes the paper.

2. PRELIMINARIES

2.1. Negligible function

A function $\varepsilon(n): N \rightarrow R$ is said to be negligible if for all positive integer $c \in N$ there exits a $n_c \in N$ such that $\varepsilon(n) < n_c$ for all $n > n_c$.

2.2. Bilinear map

Let G_1 and G_2 be two multiplicative cyclic groups with the same prime order p, and g be a generator of G_1. A bilinear pairing is a

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

map $e : G_1 \times G_1 \to G_2$ with the following properties [11,12]:

(1) $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \xleftarrow{R} Z_p^*$ and $g_1, g_2 \in G_1$.
(2) $e(g, g) \neq 1$.
(3) There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in G_1$.

2.3. Complexity assumption

The security of our system is based on a complexity assumption called 3-week decisional bilinear Diffie–Hellman inversion assumption (3-wDBDHI), which was used by Weng [13] to construct their scheme.

Let $e : G_1 \times G_1 \to G_2$ be a bilinear map. For all probability polynomial time adversaries $\mathcal{A}$, the following probability is negligible:

$$\left|\Pr\left[a, b, r \xleftarrow{R} Z_p; \quad T_0 = e(g,g)^{b/a^2}; \quad T_1 = e(g,g)^r; \right.\right.$$
$$\left.\left. z \in \{0,1\}; \quad z' \leftarrow \mathcal{A}(g, g^{1/a}, g^a, g^{a^2}, g^b, T_z) : \quad z = z'\right]\right| - \frac{1}{2}.$$

2.4. Strongly unforgeable one-time signature

A one-time signature [14] consists of a triple of algorithms $Sig = (G, S, V)$ such that, on input of a security parameter λ, $\mathcal{G}$ generates a one-time key pair (ssk, svk) , while for any message M, $\mathcal{V}(svk, \sigma, M)$ outputs 1 whenever $\sigma = \mathcal{S}(ssk, M)$ and 0 otherwise. As in [15], we need strongly unforgeable one-time signatures, which means that no probabilistic polynomial time (PPT) adversary can create a new signature even for a previously signed message.

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

2.5. Pseudorandom function family

We here review the definition of pseudorandom function (PRF) family [13]. Let $F : \mathcal{K} \times D \to R$ be a function family, where $\mathcal{K}$ is the set of keys of F, D is the domain and R is the range. Let $G : D \to R$ be a true random function family. Let $\mathcal{F}$ be a PPT adversary which outputs a bit. We consider the following two experiments:

Experiment $Exp_F^{PRF-1}(\mathcal{F})$: $[K \xleftarrow{\$} \mathcal{K}, b \xleftarrow{\$} \mathcal{F}^{F(K,\cdot)}$, Return $b]$.

Experiment $Exp_F^{PRF-0}(\mathcal{F})$: $[g \xleftarrow{\$} G, b \xleftarrow{\$} \mathcal{F}^{G(\cdot)}$, Return $b]$.

We define $\mathcal{F}$'s advantage $Adv_{F,\mathcal{F}}^{P}RF$ in attacking the pseudorandom of the function F as

$$|\Pr[Exp_F^{PRF-1}(\mathcal{F}) = 1] - \Pr[Exp_F^{PRF-0}(\mathcal{F}) = 1]|.$$

If for any PPT adversary $\mathcal{F}$, its advantage in attacking the pseudorandomness of the function family F is negligible, the we say that F is a PRF family.

2.6. Key-policy attribute-based proxy re-encryption

In this section, we first define security notions for KP-ABPRE. We use the same approach as [8], where ciphertext are labeled with a set of descriptive conditions. A tree access structure, in which each interior node of the tree is a threshold gate and the leaves are associated with conditions, is used to identify re-encryption keys. Using this approach, we can present a tree with 'AND' and 'OR' gate by using n-out-of-n and 1-out-of-n threshold gates, respectively. A user will be able to re-encrypt a ciphertext with a re-encryption key provided if and only if there is an assignment of conditions from the ciphertexts to nodes of the tree such that the tree is satisfied.

DEFINITION 1 (Access tree [2]). *Access tree $\mathcal{T}$. We incorporate the definition of access tree as in [2]. Let $\mathcal{T}$ be a tree that represents an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children a threshold value. Let num_x be the number of children of a node x and k_x be its threshold value, then we have $0 < k_x \leq num_x$. When $k_x = num_x$, it is an AND gate, and when $k_x = 1$, the threshold is an OR gate. Each leaf node x of the tree is described by a conditional keyword and a threshold value $k_x = 1$. We fit the root of an access tree to be at depth 0. Let $\overline{LN_{\mathcal{T}}}$ be the set of all the non-leaf nodes, and $LN_{\mathcal{T}}$ be the set of all the leaf nodes. We denote the parent of node x in the tree $p(x)$. The function keyword(x) is defined only if $x \in LN_{\mathcal{T}}$ and it denotes the conditional keyword associated with the leaf node x in the tree. The access tree $\mathcal{T}$ also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to num. The function inex(z) returns a index associated with the node z, where the index values are uniquely assigned to nodes in the access tree.*

Satisfying an access tree. Let $\mathcal{T}$ be an access tree with root r. $\mathcal{T}_x$ is the subtree rooted at the node x. Hence $\mathcal{T}$ is the same as $\mathcal{T}_r$. $\mathcal{T}_x(W) = 1$ denotes a set of conditions W satisfies the access tree $\mathcal{T}_x$. If $x \in LN_{\mathcal{T}}$, then $\mathcal{T}_x(W) = 1$ if and only if keyword$(x) \in W$. If $x \in \overline{LN_{\mathcal{T}}}$, evaluate $\mathcal{T}_z(W)$ for all children z of node x, $\mathcal{T}_x(W)$ return 1 if and only if at least k_x children return 1.

DEFINITION 2 (KP-ABPRE). *A (single-use) KP-ABPRE [8] scheme consists the following algorithms*:

(i) *GlobalSetup(λ) : the GlobalSetup algorithm is run by a trusted party that, on input a security parameter λ, the global public parameters PP are outputted.*
(ii) *KeyGen(i) : the KeyGen algorithm generates the public key pk_i and the private key sk_i for user i.*
(iii) *RKeyGen$(sk_i, \mathcal{T}, sk_j)$: the RKeyGen algorithm takes as input the private key sk_i, an access tree $\mathcal{T}$, and the private key sk_j, it outputs the conditional re-encryption key $rk_{i,\mathcal{T},j}$.*

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

(iv) *Enc*1(pk, m) : *the level* 1 *encryption algorithm takes as input the public key pk and the message* $m \in M$, *it outputs the first level ciphertext CT under the public key pk. Here M denotes the message space.*

(v) *Enc*2(pk, m, W) : *the level* 2 *encryption algorithm takes as input the public key pk, the message* $m \in M$ *and the conditional keywords set W, it outputs the second level ciphertext CT associated with W under the public key pk.*

(vi) *REnc*$(CT_i, rk_{i,\mathcal{T},j})$: *the REnc algorithm takes as input the second level ciphertext* CT_i *associated with conditions W, and the conditional re-encryption key* $rk_{i,\mathcal{T},j}$ *for access tree* $\mathcal{T}$. *It outputs the first level ciphertext* CT_j *under* pk_j *if* $\mathcal{T}(W) = 1$, *or an error symbol* $\perp$ *otherwise.*

(vii) *Dec*1(CT_j, sk_j) : *the Dec*1 *algorithm takes as input the first level ciphertext* CT_j *under* pk_j *and the private key* sk_j. *It output the message* $m \in M$ *or an error symbol* $\perp$.

(viii) *Dec*2(CT_i, sk_i) : *the Dec*1 *algorithm takes as input the second level ciphertext* CT_i *under* pk_i *and the private key* sk_i. *It output the message* $m \in M$ *or an error symbol* $\perp$.

Note that we omit the global parameters *PP* as the other algorithms' input for simplicity. The correctness of KP-ABPRE means that, for any condition set W, any message $m \in M$, any $(pk_i, sk_i) \leftarrow KeyGen(i)$, $(pk_j, sk_j) \leftarrow KeyGen(j)$, any $rk_{i,\mathcal{T},j} \leftarrow REnc(CT_i, rk_{i,\mathcal{T},j})$, where $\mathcal{T}(W) = 1$ and $CT_i = Enc2(pk_i, m, W)$,

$$\Pr[Dec2(CT_i, sk_i) = m] = 1$$

and

$$\Pr[Dec1(ReEnc(CT_i, RKeyGen(sk_i, \mathcal{T}, sk_j)), sk_j) = m] = 1.$$

In the following, we provide the game-based security definition of KP-ABPRE. Our definition considers a challenger that produces a number of public keys. As [8], we let the target public key pk_{i^*}, corrupted users, the honest users and the target keywords set W^* be determined at the beginning of the game.

DEFINITION 3 (KP-ABPRE-IND-CCA game). *Next we consider the following two games, which correspond to the security of the first level ciphertext and the second level ciphertext individually.*
Game 1 : *IND-level* 2*-CCA game, which considers the security of level* 2 *ciphertext.*

(1) *Setup. The challenger* $\mathcal{C}$ *perform GlobalSetup*(λ) *to get the public parameter PP. Give the public parameter PP to the adversary* $\mathcal{A}$.

(2) *Query phase* 1. *The adversary* $\mathcal{A}$ *makes the following queries*:

(a) *Uncorrupted key generation* $\mathcal{O}_{HU}$: *on input an uncorrupted user i, the challenger* $\mathcal{C}$ *first runs algorithm KeyGen*(i) *to obtain a public/secret key pair* (pk_i, sk_i), *and then sends* pk_i *to the adversary* $\mathcal{A}$.

(b) *Corrupted key generation* $\mathcal{O}_{CU}$: *on input an uncorrupted user j, the challenger* $\mathcal{C}$ *first runs algorithm KeyGen*(j) *to obtain a public/secret key pair* (pk_j, sk_j), *and then sends* (pk_j, sk_j) *to the adversary* $\mathcal{A}$.

(c) *Re-encryption key generation* $\mathcal{O}_{rk}$: *on input* $(pk_i, \mathcal{T}, pk_j)$, *the challenger* $\mathcal{C}$ *runs algorithm* $RKeyGen(sk_i, \mathcal{T}, sk_j)$ *to generate a re-encryption key* $rk_{i,\mathcal{T},j}$ *and returns it to the adversary* $\mathcal{A}$. *Here,* sk_i *is the secret key responds to* pk_i. *It is required that* pk_i *and* pk_j *have been generated beforehand by algorithm KeyGen.*

(d) *Re-encryption* $\mathcal{O}_{re}$: *on input* $(pk_i, pk_j, (\mathcal{T}, CT_i))$, *the challenger*$\mathcal{C}$ *runs algorithm*

$$CT_j = ReEnc(CT_i, RKeyGen(sk_i, \mathcal{T}, sk_j))$$

and returns the resulting ciphertext CT_j *to the adversary* $\mathcal{A}$. *It is required that* pk_i *and* pk_j *have been generated beforehand by algorithm KeyGen.*

(e) *Decryption* $\mathcal{O}_{d_2}$: *on input* $(pk_i, (\mathcal{T}, CT_i))$, $(pk_i, (\mathcal{T}, CT_i))$ *denotes the queries on a second level ciphertext. Challenger* $\mathcal{C}$ *returns the result of* $Dec2(CT_i, sk_i)$ *to* $\mathcal{A}$. *It is required that* pk_i *has been generated beforehand by algorithm KeyGen.*

(f) *Decryption query* $\mathcal{O}_{d_1}$: *on input* (pk_j, CT_j), (pk_j, CT_j) *denotes the queries on a first level ciphertext. Challenger* $\mathcal{C}$ *returns the result of* $Dec1(CT_j, sk_j)$ *to* $\mathcal{A}$. *It is required that* pk_j *has been generated beforehand by algorithm KeyGen.*

(3) *Challenge. Once* $\mathcal{A}$ *decides that Phase* 1 *is over, it outputs two equal length message* (m_0, m_1). *Challenger* $\mathcal{C}$ *chooses a bit* $b \in \{0, 1\}$ *and sets the challenge ciphertext to be* $CT^* = Enc2(pk_{i^*}, m_b, W^*)$, *which is sent to* $\mathcal{A}$.

(4) *Query Phase* 2. $\mathcal{A}$ *continues making queries as in the query phase* 1.

(5) *Guess.* $\mathcal{A}$ *outputs the guess* b'. *The adversary wins if* $b' = b$.

During the above game, adversary $\mathcal{A}$ *is subject to the following restrictions*:

(i) $\mathcal{A}$ *cannot issue corrupted key generation oracles on* (i^*) *to obtain the target secret key* sk_{i^*}.

(ii) $\mathcal{A}$ *cannot issue decryption oracles on neither* $(pk_{i^*}, (W^*, CT^*))$ *nor* (pk_j, CT_j) *where* (pk_j, CT_j) *is a re-encryption of the challenge pair* $(pk_{i^*}, (W^*, CT^*))$.

(iii) $\mathcal{A}$ *cannot issue re-encryption oracles on* $(pk_{i^*}, pk_j, (\mathcal{T}, CT^*))$, *if* $\mathcal{T}(W^*) = 1$ *and* pk_j *appears in a previous corrupted key generation oracle.*

(iv) $\mathcal{A}$ *cannot obtain the re-encryption key* $rk_{i^*,\mathcal{T},j}$, *if* $\mathcal{T}(W^*) = 1$ *and* pk_j *appears in a previous corrupted key generation oracle.*

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

We refer to the above adversary $\mathcal{A}$ as an IND-L2-CCA adversary. Its advantage is defined as

$$Succ_{\mathcal{A}}^{Game_1}(\lambda) = |\Pr[b' = b] - 1/2|.$$

Game 2 : *IND-level1-CCA game, which considers the security of level1 ciphertext.*

The above definition provides adversaries with a second level ciphertext in the challenge phase. A complementary definition of security captures their inability to distinguish first level ciphertext as well. For single-use schemes, the adversary is provided with access to all re-encryption keys in this definition. Also, the level 2 *decryption oracle is unnecessary. The re-encryption oracle becomes useless since all re-encryption keys are available to $\mathcal{A}$. Furthermore, since first level ciphertext cannot be re-encrypted, their is no need to keep attackers from obtaining all honest-to-corrupt re-encryption keys.*

(1) *Setup. The challenger $\mathcal{C}$ perform GlobalSetup(λ) to get the public parameter PP. Give the public parameter PP to the adversary $\mathcal{A}$.*
(2) *Query Phase* 1. *The adversary $\mathcal{A}$ makes the following queries*:
   (a) *Uncorrupted key generation $\mathcal{O}_{HU}$: on input an uncorrupted user i, the challenger $\mathcal{C}$ first runs algorithm KeyGen(i) to obtain a public/secret key pair (pk_i, sk_i), and then sends pk_i to the adversary $\mathcal{A}$.*
   (b) *Corrupted key generation $\mathcal{O}_{CU}$: on input an uncorrupted user j, the challenger $\mathcal{C}$ first runs algorithm KeyGen(j) to obtain a public/secret key pair (pk_j, sk_j), and then sends (pk_j, sk_j) to the adversary $\mathcal{A}$.*
   (c) *Re-encryption key generation $\mathcal{O}_{rk}$: on input $(pk_i, \mathcal{T}, pk_j)$, the challenger $\mathcal{C}$ runs algorithm RKeyGen $(sk_i, \mathcal{T}, sk_j)$ to generate a re-encryption key $rk_{i,\mathcal{T},j}$ and returns it to the adversary $\mathcal{A}$. Here, sk_i is the secret key responds to pk_i. It is required that pk_i and pk_j have been generated beforehand by algorithm KeyGen.*
   (e) *Decryption query $\mathcal{O}_{d_1}$: On input (pk_j, CT_j), (pk_j, CT_j) denotes the queries on a first level ciphertext. Challenger $\mathcal{C}$ returns the result of $Dec1(CT_j, sk_j)$ to $\mathcal{A}$. It is required that pk_j has been generated beforehand by algorithm KeyGen.*
(3) *Challenge. Once $\mathcal{A}$ decides that Phase* 1 *is over, it outputs two equal length message (m_0, m_1). Challenger $\mathcal{C}$ chooses a bit $b \in \{0, 1\}$ and sets the challenge ciphertext to be $CT^* = Enc1(pk_{i^*}, m_b)$, which is sent to $\mathcal{A}$.*
(4) *Query Phase* 2. *$\mathcal{A}$ continues making queries as in Query Phase* 1.
(5) *Guess. $\mathcal{A}$ outputs the guess b'. The adversary wins if $b' = b$.*

During the above game, adversary $\mathcal{A}$ is subject to the following restrictions:

(i) *$\mathcal{A}$ cannot issue corrupted key generation queries on (i^*) to obtain the target secret key sk_{i^*}.*
(ii) *$\mathcal{A}$ cannot issue decryption queries on a the challenge pair (pk_{i^*}, CT^*).*

We refer to the above adversary $\mathcal{A}$ as an IND-L1-CCA adversary. Its advantage is defined as

$$Succ_{\mathcal{A}}^{Game_2}(\lambda) = |\Pr[b' = b] - 1/2|.$$

The KP-ABPRE scheme is said to be KP-ABPRE-IND-CCA secure if both $Succ_{\mathcal{A}}^{Game_1}(\lambda)$ and $Succ_{\mathcal{A}}^{Game_2}(\lambda)$ are negligible.

Remark 2. Let HU be the set of honest parties, including the target user i^*, and CU be the set of corrupt parties. As described in the security model, the following queries are allowed:

(1) $i \in HU \setminus \{i^*\}$ and $j \in CU$, for all $\mathcal{T}(W^*) = 1$ during the re-encryption key query;
(2) $i \in CU$ and $j \in HU$, for all $\mathcal{T}(W^*) = 1$ during the re-encryption key query;
(3) $i \in HU \setminus \{i^*\}, j \in CU$, for all $\mathcal{T}(W^*) = 1$ during the re-encryption query.

Unfortunately, the above three legal queries were not consider in Fang *et al.*'s scheme [8]. In this paper, we allow the adversary making the three queries, thus enhanced the adversary's ability. Hence, it is clear that our model is more rigorous than the existing scheme in the literature.

Master secret security. Ateniese *et al.* [5] defined another important security requirement, called the master secret security. This suggests that even if the dishonest proxy colludes with delegatees, it is still impossible for them to expose the private key of their common delegators in its entirety. As discussed in [16], the notion of CCA security of level 1 ciphertext implies the master secret security.

3. OUR CCA-SECURE KP-ABPRE WITHOUT RANDOM ORACLES

Inspired by ABE scheme [2], we present our constructions of CCA-secure KP-ABPRE scheme. Similarly, in our scheme, ciphertexts are encrypted using a condition set W, and the re-encryption key is labeled with an access structure $\mathcal{T}$. Thus, a ciphertext created under conditional set W can be re-encrypted by a re-encryption key for access tree $\mathcal{T}$ if $\mathcal{T}(W) = 1$.

3.1. Technical difficulties and our approach

Unfortunately, the above key-policy ABE only achieves chosen plaintext secure. Fang *et al.*'s ICPRE-FG scheme [8] uses the

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

Fujisaki–Okamoto conversion [17] to provide the validity check of both levels 1 and 2 ciphertext for the decryptor. Inherently, the scheme achieves CCA-security only in the random oracle model. The difficulty is to add validity check on the ciphertexts without relying on random oracles.

In Canetti–Hohenberger's scheme [15], it appends the ciphertext a checksum value consisting of an element of G_1 raised to the random encryption exponent r. However, it cannot be directly used in the unidirectional case. As pointed out in [16], after re-encryption, the component $e(g,g)^{x_j r}$ in target group G_2 cannot be used any longer to check the equality of two discrete logarithms in group G_1 and G_2. Therefore, the simulator cannot decide whether is has been modified. Inspired by Weng *et al.* [13], we use PRF to add validity check on the first level ciphertext.

3.2. Our construction

Let G_1 be bilinear group of prime order p, and g be a generator of G_1. Additionally, let $e : G_1 \times G_1 \to G_2$ denote the bilinear map.

We also define the Lagrange coefficient $\Delta_{\omega,S}(x)$ for $\omega \in Z_p$ and a set S, of elements in Z_p:

$$\Delta_{\omega,S}(x) = \prod_{i \in S, i \neq \omega} \frac{x-i}{\omega - i}$$

Our proposed scheme consists of the following algorithms:

(i) GlobalSetup(λ): let λ be the security parameter, and (p, g, G_1, G_2, e) be the bilinear map parameters. $M = \{0,1\}^k$ denotes the message space. Let $Sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a strongly unforgeable one-time signature scheme, svk denotes the verification key output by $\mathcal{G}(\lambda)$, $\mathcal{S}$ denotes the sign algorithm and $\mathcal{V}$ denotes the verify algorithm. Let $H_1 : \{0,1\}^* \to G_1^*$ be collision–resistent hash function. Choose a PRF family $F : G_2 \times G_1 \to \{0,1\}^{l-k} \parallel \{0,1\}^k$. Pick generators $g, h, g_2, g_3 \xleftarrow{\$} G_1$.

Output the public parameters $(p, g, h, g_2, g_3, G_1, G_2, e, H_1, F, l, k, \mathcal{G}, \mathcal{S}, \mathcal{V})$.

(ii) KeyGen(i): user i picks a random value $x_i \xleftarrow{R} Z_p^*$, and sets his public key as $pk_i = g^{x_i}$, and the private key as $sk_i = x_i$.

(iii) RKeyGen($sk_i, \mathcal{T}, sk_j$): on input user $i's$ private $sk_i = x_i$, an access tree $\mathcal{T}$ and user $j's$ private $sk_j = x_j$. The ReKeyGen algorithms outputs a re-encryption key $rk_{i,\mathcal{T},j}$ which enables the proxy to re-encrypted a ciphertext under a condition set W, if and only if $\mathcal{T}(W) = 1$. The algorithms proceeds as follows. First chooses a polynomial q_x for each non-leaf node x in the tree $\mathcal{T}$. These polynomials are chosen in the following way in a top down manner, starting from the root node r. It calls the procedure $RKG(\mathcal{T}, x_j/x_i)$, where procedure $RKG(\mathcal{T}, x_j/x_i)$ is defined as follows.

For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of the note, that is, $d_x = k_x - 1$. Now for the root r, set $q_r(0) = x_j/x_i$ and d_r other points of the polynomial q_r randomly to define it completely. For any other node x, set $q_x(0) = q_{p(x)}(index(x))$ and choose d_x other points randomly to completely define q_x, once the polynomials have been decided, for each leaf node x, set $\omega = keyword(x)$. Thus the re-encryption key given to the proxy are calculated as follows:

Select a random value $r_x \xleftarrow{R} Z_p^*$, computes

$$a_x = g^{q_x(0)} H_1(pk_i, \omega)^{r_x} \quad b_x = pk_i^{r_x}$$

Let $a = \{a_x : x \in LN_{\mathcal{T}}\}$ and $b = \{b_x : x \in LN_{\mathcal{T}}\}$. Return $rk_{i,\mathcal{T},j} = (a, b, \mathcal{T})$ as the re-encryption key.

On generating the re-encryption keys. As both the private key of user i and user j are required in the re-encryption key generation algorithm, user i needs to interact with user j when generating the re-encryption key. The process of re-encryption key generation is outlined as follows:

User j runs $RKG(\mathcal{T}, x_j)$ to get $(a', b', \mathcal{T})$ where $a' = \{a'_x : x \in LN_{\mathcal{T}}\}$ and $b' = \{b'_x : x \in LN_{\mathcal{T}}\}$, and sends the results to user i.

On receiving a', b', user i calculates $a = \{a_x = {a'_x}^{1/x_i} : x \in LN_{\mathcal{T}}\}$, $b = \{b_x = {b'_x}^{1/x_i} : x \in LN_{\mathcal{T}}\}$.

Note that, $a'_x = g^{q'_x(0)} H(pk_i, x)^{r'_x}$, $b'_x = pk_i^{r'_x}$. Let $r_x = r'_x/x_i$, we have

$$a_x = {a'_x}^{1/x_i} = g^{q'_x(0)/x_i} H(pk_i, x)^{r'_x/x_i} = g^{q'_x(0)/x_i} H(pk_i, x)^{r_x}$$

$$b_x = {b'_x}^{1/x_i} = pk_i^{r'_x/x_i} = pk_i^{r_x}.$$

Thus, $a = \{a_x = a_x : x \in LN_{\mathcal{T}}\}$, $b = \{b_x = b_x : x \in LN_{\mathcal{T}}\}$ is identical to the above re-encryption key generation algorithm $RKG(\mathcal{T}, x_j/x_i)$.

(iv) Enc2(pk_i, W, m): to encrypt a message $m \in M$, under a public key pk_i and a keyword set W at the second level, the sender proceeds as follows:

(1) Set $C_1 = W$.

(2) Select a one-time signature key pair as (svk, ssk). Set $C_2 = svk$.

(3) Select a random value $s \xleftarrow{R} Z_p^*$ and compute

$$C_3 = pk_i^s, \quad C_4 = h^s, \quad C_6 = (g_2^{svk} g_3)^s,$$
$$C_7 = H_1(pk_i, \omega)^s_{\omega \in W}.$$

(4) Compute $K = e(g,g)^s$, and set

$$C_5 = [F(K, C_4)]_{l-k} \parallel ([F(K, C_4)]^k \oplus m).$$

(5) Run the signing algorithm $\mathcal{S}(ssk, (C_4, C_5, C_6))$ and denote the signature S.

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

(6) Output the ciphertext $CT_i = (C_1, C_2, C_3, C_4, C_5, C_6, S, C_7)$.

(v) Enc1(pk_j, m): to encrypt a message $m \in M$ under the public key pk_j at the first level, the sender proceeds as follows:

(1) Select a one-time signature key pair as (svk, ssk). Set $C_2 = svk$.

(2) Select a random value $s \xleftarrow{R} Z_p^*$ and compute

$$C_3' = e(pk_j, g)^s, \quad C_4 = h^s, \quad C_6 = (g_2^{svk} g_3)^s.$$

(3) Compute $K = e(g, g)^s$, and set

$$C_5 = [F(K, C_4)]_{l-k} \parallel ([F(K, C_4)]^k \oplus m).$$

(4) Run the signing algorithm $\mathcal{S}(ssk, (C_4, C_5, C_6))$ and denote the signature S.

(5) Output the ciphertext $CT_j = (C_2, C_3', C_4, C_5, C_6, S)$.

(vi) ReEnc$(rk_{i,\mathcal{T},j}, CT_i)$: on input a re-encryption key $rk_{i,\mathcal{T},j} = (a, b, \mathcal{T})$ and a second level ciphertext $CT_i = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, S)$ under the key word set W. For all $\omega \in W$ it check whether the following equalities hold:

$$\mathcal{V}(C_2, S, (C_4, C_5, C_6)) \stackrel{?}{=} 1, \tag{1}$$

$$e(C_3, g_2^{C_2} g_3) \stackrel{?}{=} e(pk_i, C_6), \tag{2}$$

$$e(C_3, h) \stackrel{?}{=} e(pk_i, C_4), \tag{3}$$

$$e(C_3, H_1(pk_i, \omega))_{\omega \in W} \stackrel{?}{=} e(pk_i, C_7). \tag{4}$$

If not, output $\perp$. Otherwise, we define a recursive algorithms $NodeReEnc(CT_i, rk_{i,\mathcal{T},j}, x)$ that takes as input the ciphertext CT_i, the re-encryption key $rk_{i,\mathcal{T},j}$, and a note x in the tree.

(1) For leaf node x, let $\omega = keyword(X)$, if $\omega \in W$, then

$$\begin{aligned} &NodeReEnc(CT_i, rk_{i,\mathcal{T},j}, x) \\ &= \frac{e(C_3, a_x)}{e(b_x, C_7)} = \frac{e(pk_i^s, g^{q_x(0)} H_1(pk_i, \omega)^{r_x})}{e(pk_i^{r_x}, H_1(pk_i, \omega)^s)} \\ &= e(pk_i, g)^{sq_x(0)} \end{aligned}$$

Otherwise, outputs $NodeReEnc(CT_i, rk_{i,\mathcal{T},j}, x) = \perp$.

(2) If x is a non-leaf node, consider the recursive case, for all nodes z that are children of x, it calls $NodeReEnc(CT_i, rk_{i,\mathcal{T},j}, z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child notes z, such that $F_z \neq \perp$. If no such set exists, then the node was not satisfied and the function $NodeReEnc(CT_i, rk_{i,\mathcal{T},j}, x)$ returns $\perp$. Otherwise, let $S_x' = \{index(z) : z \in S_x\}$, and computes

$$\begin{aligned} F_x &= \prod_{z \in S_x, i = index(z)} (F_z)^{\Delta_{i,S_x'}(0)} \\ &= \prod_{z \in S_x, i = index(z)} \left(e(pk_i, g)^{sq_z(0)}\right)^{\Delta_{i,S_x'}(0)} \\ &= \prod_{z \in S_x, i = index(z)} \left(e(pk_i, g)^{sq_{p(z)}(index(z))}\right)^{\Delta_{i,S_x'}(0)} \\ &= \prod_{z \in S_x, i = index(z)} \left(e(pk_i, g)^{sq_x(i)}\right)^{\Delta_{i,S_x'}(0)} \\ &= e(pk_i, g)^{sq_x(0)}. \end{aligned}$$

and return the result.

Finally, we can compute

$$\begin{aligned} C_3' &= NodeReEnc(CT_i, rk_{i,\mathcal{T},j}, r) \\ &= e(pk_i, g)^{sq_r(0)} \\ &= e(pk_i, g)^{sx_j/x_i} \\ &= e(pk_j, g)^s. \end{aligned}$$

Output the re-encrypted ciphertext (level1) is $CT_j = (C_2, C_3', C_4, C_5, C_6, S)$.

(vii) Dec2(sk_i, CT_i): On input a private key sk_i and a second level ciphertext $CT_i = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, S)$, he proceeds as follows:

(1) First, check the validity of the ciphertexts as in equations (1–4). If the verification fails, output $\perp$ and abort.

(2) Compute $K = e(C_3, g)^{1/sk_i}$. If $F[K, C_4]_{l-k} = [C_5]_{l-k}$ holds, output $m = F[K, C_4]^k \oplus [C_5]^k$; else output $\perp$ and abort.

(viii) Dec1(sk_j, CT_j): on input a private key sk_j and a first level ciphertext $CT_j = (C_2, C_3', C_4, C_5, C_6, S)$, he proceeds as follows:

(1) First check the validity of the ciphertexts as in equations (1). If the verification fails, output $\perp$ and abort.

(2) Next, check whether $e(C_4, g_2^{C_2} g_3) = e(h, C_6)$, if the check fails, output $\perp$ and abort.

(3) Compute $K = {C_3'}^{1/sk_j}$. Output $m = F[K, C_4]^k \oplus [C_5]^k$, if the following equality holds:

$$F[K, C_4]_{l-k} = [C_5]_{l-k}. \tag{5}$$

Otherwise, output $\perp$.

Correctness. We now explain the correctness of our scheme:

(1) For an original ciphertext $CT_i = (C_1, C_2, C_3, C_4, C_5, C_6, S, C_7)$, we have $K = e(C_3, g)^{1/sk_i} = e(pk_i^s, g)^{1/sk_i} = e(g, g)^s$.

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

(2) For a re-encrypted ciphertext $CT_j = (C_2, C_3', C_4, C_5, C_6, S)$, we have

$$\begin{aligned} C_3' &= NodeReEnc(CT_i, rk_{i,\mathcal{T},j}, r) \\ &= e(pk_i, g)^{sq_r(0)} \\ &= e(pk_j, g)^s. \end{aligned}$$

Then we have $K = C_3'^{1/sk_j} = e(pk_j, g)^{s/sk_j} = e(g, g)^s$.

Remark 3. In Shao's scheme [18], $C_3 = pk_i^r$ is re-encrypted to $C_3' = e(C_3, g^{x_j/x_i}) = e(g, g)^{x_j r}$ using the re-encryption key $rk = g^{x_j/x_i}$. They parse $C_5 = e(g, g)^r \cdot m$ into $K = e(g, g)^r$ and m, where K is the used as the symmetric encryption secret key to encrypt the m.

Unfortunately, as pointed by Weng *et al.* [13], their scheme failed to check the validity of C_3' and violate the principle proposed by them [13]: for a first level ciphertext CT_j re-encrypted from a second ciphertext CT_i, CT_j should not contain all the components of CT_i. Inspired by their work, we use K as a PRF seed to encrypt m: $[F(K, C_4)]_{l-k} \parallel ([F(K, C_4)]^k \oplus m)$.

3.3. Security of our KP-ABPRE scheme

In this subsection, we prove the KP-ABPRE-IND-CCA security for our scheme without random oracles. The analysis of Games 1 and 2 are as follows.

THEOREM 1. *If H_1 is a target collision resistant hash function, F is a PRF family and $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature, then the above scheme is KP-ABPRE-IND-CCA secure without random oracles under the* 3*-wDBDHI assumption.*

LEMMA 1. *Our scheme is IND-L2-CCA secure under the* 3*-wDBDHI assumption, assuming H_1 is a target collision resistant hash function, F is a PRF family and $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature.*

Proof. Our approach to proving Lemma 1 closely follows the security proof for Libert and Vergnaud's scheme [16] and the key-policy ABE scheme [2]. Suppose there exits a polynomial-time adversary $\mathcal{A}$ that can attack our scheme in the standard model with probability ε', we built a simulator $\mathcal{B}$ that can play a 3-wDBDHI game with probability ε, where

$$\varepsilon' \geqslant \frac{\epsilon}{\dot{e}(1 + q_{rk})} - \frac{q_O}{p} - Adv^{OTS} - Adv^{TCR}_{H_1,\mathcal{A}} - Adv^{PRF}_{F,\mathcal{F}}.$$

The adversary arbitrarily select a user i^* from the set of honest users HU and commit to the user to be attacked. In the following, we call HU be the set of honest parties, including the target user i^*, and CU be the set of corrupt parties. $\mathcal{A}$ first generates an keyword set W^* that it intends to attack.

We let the challenger set the group G_1 and G_2 with an efficient bilinear map e and a generator g of G_1. $\mathcal{B}$ inputs a 3-wDBDHI instance $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, T)$ and has to distinguish $T = e(g, g)^{b/a^2}$ from a random element in G_2.

Next, $\mathcal{B}$ sets up the global parameters as follows: $\mathcal{B}$ picks a strongly unforgeable one-time signature $\mathcal{G}, \mathcal{S}, \mathcal{V}$ and runs $\mathcal{G}(1^k) \to (ssk^*, svk^*)$, and sets the generators $h = A_2^{\alpha_0}, g_2 = A_1^{\alpha_1}, g_3 = A_1^{-\alpha_1 svk^*} \cdot A_2^{\alpha_2}$ for a randomly chosen $\alpha_0, \alpha_1, \alpha_2 \xleftarrow{R} Z_p^*$. Finally, $\mathcal{B}$ samples a pairwise independent hash functions H_1, such that $H_1(pk_i, \omega) = pk_i^{H_1(\omega)}$. Without loose of generality, assume that H_1 is target collision resistant and F is a PRF family. The system parameters are $(p, g, G_1, G_2, e, h, g_2, g_3, H_1, F, l, k, \mathcal{G}, \mathcal{S}, \mathcal{V})$.

(i) Uncorrupted key: on input i to $\mathcal{O}_{HU}$, public keys of honest users $i \in HU \setminus \{i^*\}$ are defined as follows: $\mathcal{B}$ select a random $x_i \xleftarrow{R} Z_p^*$, then $\mathcal{B}$ computes $pk_i = A_1^{x_i}$. The target user's public key is set as $pk_{i^*} = A_2^{x_{i^*}}$ for randomly chosen $x_{i^*} \xleftarrow{R} Z_p^*$. Sends public key pk_i, pk_{i^*} to $\mathcal{A}$.

(ii) Corrupted key: on input j to $\mathcal{O}_{CU}$, public keys of corrupted user $j \in CU$ are defined as follows: $\mathcal{B}$ select a random $x_j \xleftarrow{R} Z_p^*$, then $\mathcal{B}$ computes $sk_j = x_j, pk_i = g^{x_j}$. Sends public and private key (sk_j, pk_j) to $\mathcal{A}$.

(iii) Re-encryption key: on input $(pk_i, \mathcal{T}, pk_j)$ to $\mathcal{O}_{rk}$, consider a query for a re-encryption key corresponding to an access tree $\mathcal{T}$, the only restriction is that $\mathcal{T}(W^*) = 0$ when $pk_i = pk_i^*$ and $j \in CU$, otherwise $\mathcal{B}$ abort and output $\perp$.

If the above condition is met, $\mathcal{B}$ first define the following three procedures: *PolySat*, *PolyUnsat* and $\widetilde{PolySat}$.

(a) $PolySat(\mathcal{T}_x, W, \lambda_x)$: this procedures sets up the polynomials for the nodes of an access sub-tree with satisfied root node, that is $\mathcal{T}_x(W) = 1$. The procedure takes an access tree $\mathcal{T}_x$(with root node x) as input along with a condition set and an integer $\lambda_x \in Z_p$. It first sets up a polynomial q_x of degree d_x for the root node x. It sets $q_x(0) = \lambda_x$ and sets the rest d_x points randomly to completely fix q_x. Now it sets polynomials for each child note x' of x by calling the procedure $PolySat(\mathcal{T}_{x'}, W, q_x(index(x')))$. Notice in this way, $q_{x'}(0) = q_x(index(x))$ for each child note x' of x.

(b) $PolyUnsat(\mathcal{T}_x, W, g^{\lambda_x})$: this procedures sets up the polynomials for the nodes of an access sub-tree with unsatisfied root node, that is $\mathcal{T}_x(W) = 0$. The procedures takes an access tree $\mathcal{T}_x$ (with root node x) as input along with a condition set and an element $g^{\lambda_x} \in G_1$. It first sets up a polynomial q_x of degree d_x for the root node x. It sets $q_x(0) = \lambda_x$ for the unknown valueλ_x. Because $\mathcal{T}_x(W) = 0$ no more than d_x children of x are satisfied. Let $h_x \leqslant d_x$ be the

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

number of satisfied children of x. For each satisfied children x' of node x, the procedure chooses a random point $\lambda_{x'} \xleftarrow{R} Z_p$ and sets $q_x(index(x')) = \lambda_{x'}$. It then fixes the remaining $d_x - h_x$ points of q_x randomly to completely define q_x. Now the algorithm recursively defines polynomials for the rest of the nodes in the tree as follows. For each child node x' of x, the algorithm calls:

* $PolySat(\mathcal{T}_{x'}, W, q_x(index(x')))$: if x' is a satisfied node. Note that $q_x(index(x'))$ is known in this case.
* $PolyUnsat(\mathcal{T}_{x'}, W, g^{q_x(index(x'))})$: if x' is not a satisfied node. Note that only $g^{q_x(index(x'))}$ can be obtained by interpolation as only $g^{q_x(0)}$ in this case. Note that in this case also $q_{x'}(0) = q_x(index(x'))$ for each child node x' of x.

We have

$$g^{q_x(\cdot)} = \prod_{x' \in S_x, i = index(x')} g^{q_x(i)\Delta_{i,S'_x}(\cdot)} \cdot g^{\lambda_x \Delta_{0,S'_x}(\cdot)}.$$

(c) $\widetilde{PolySat}(\mathcal{T}_x, W, g^{\lambda_x})$: this procedures sets up the polynomials for the nodes of an access sub-tree with satisfied root node, that is $\mathcal{T}_x(W) = 1$. The procedures takes an access tree $\mathcal{T}_x$(with root node x) as input along with a condition set and an element $g^{\lambda_x} \in G_1$. It first sets up a polynomial q_x of degree d_x for the root node x. It sets $q_x(0) = \lambda_x$ for the unknown value λ_x. Because $\mathcal{T}_x(W) = 1$ there are at least $d_x + 1$ children of x are satisfied. Chooses any d_x satisfied children x' of node x, the procedure chooses a random point $\lambda_{x'} \xleftarrow{R} Z_p$ and sets $q_x(index(x')) = \lambda_{x'}$. Thus it fixes the polynomial q_x completely. Now the algorithm recursively defines polynomials for the rest of the nodes in the tree as follows. For each child node x' of x, the algorithm calls:

* $PolySat(\mathcal{T}_{x'}, W, q_x(index(x')))$: if x' is a satisfied node. Note that $q_x(index(x'))$ is known in this case.
* $PolyUnsat(\mathcal{T}_{x'}, W, g^{q_x(index(x'))})$: if x' is not a satisfied node. Note that only $g^{q_x(index(x'))}$ can be obtained by interpolation as only $g^{q_x(0)}$ in this case. Note that in this case also $q_{x'}(0) = q_x(index(x'))$ for each child node x' of x.

Let $\Phi = sk_j/sk_i$, $g^{\Phi} = g^{sk_j/sk_i}$, to answer the re-encryption key query$(pk_i, \mathcal{T}, pk_j)$, $\mathcal{B}$ has to distinguish several situations:

(1) If $i \in CU$ and $j \in CU$, that means $\Phi = x_j/x_i$. Since $\mathcal{B}$ knows the private key x_i for user i and the private key x_j for user j, $\mathcal{B}$ can compute the conditional re-encryption $rk_{i,W,j}$ as the re-encryption key generation algorithm.

(2) If

$$\begin{cases} i \in HU\backslash\{i^*\},\ j = i^* & \Phi = sk_j/sk_i = ax_{i^*}/x_i,\ g^{\Phi} = A_1^{x_{i^*}/x_i} \\ i \in HU\backslash\{i^*\},\ j \in CU & \Phi = sk_j/sk_i = x_j/ax_i,\ g^{\Phi} = A_{-1}^{x_j/x_i} \\ i = i^*,\ j \in HU\backslash\{i^*\} & \Phi = sk_j/sk_i = ax_j/a^2x_{i^*},\ g^{\Phi} = A_{-1}^{x_j/x_{i^*}} \\ i \in CU,\ j \in HU\backslash\{i^*\} & \Phi = sk_j/sk_i = ax_j/x_i = ax_j/x_i,\ g^{\Phi} = A_1^{x_j/x_i} \\ i \in CU,\ j = i^* & \Phi = sk_j/sk_i = a^2x_{i^*}/x_i,\ g^{\Phi} = A_2^{x_{i^*}/x_i}. \end{cases}$$

$\mathcal{B}$ first runs $\widetilde{PolyUnsat}(\mathcal{T}, W^*, g^{\Phi})$ if $\mathcal{T}(W^*) = 0$, and $\widetilde{PolySat}(\mathcal{T}, W^*, g^{\Phi})$ if $\mathcal{T}(W^*) = 1$ to define a polynomial q_x for each node x of $\mathcal{T}$. Note that for each leaf node x of $\mathcal{T}$, $\mathcal{B}$ know q_x completely if x is satisfied, if x is not satisfied, then at least $g^{q_x(0)}$ is known. Let $\omega = keyword(x)$, the re-encryption key corresponding to each leaf node is given using its polynomial as follows.

If $\omega \in W^*$:

$$a_x = g^{q_x(0)} \cdot H_1(pk_i, \omega)^{r_x}, \quad b_x = pk_i^{r_x},$$

where r_x is chosen randomly in Z_p.

If $\omega \notin W^*$:

$$\begin{aligned} a_x &= \prod_{x' \in S_x, i = index(x')} g^{q_x(i)\Delta_{i,S'_x}(0)} \\ &\quad \cdot (A_1^{x_{i^*}/x_i})^{\Delta_{0,S'_x}(0)} \cdot H_1(pk_i, \omega)^{r_x} \\ &= \prod_{x' \in S_x, i = index(x')} g^{q_x(i)\Delta_{i,S'_x}(0)} \\ &\quad \cdot (g^{\Phi})^{\Delta_{0,S'_x}(0)} \cdot H_1(pk_i, \omega)^{r_x} \\ &= g^{q_x(0)} \cdot H_1(pk_i, \omega)^{r_x}, \\ b_x &= pk_i^{r_x}. \end{aligned}$$

Where r_x is randomly chosen in Z_p^*.

(3) If $i = i^*, j \in CU$ and $\mathcal{T}(W^*) = 0$. $\mathcal{B}$ outputs a random bit in $\{0, 1\}$ and abort.

(4) If $i \in HU \setminus \{i^*\}$ and $j \in HU \setminus \{i^*\}$, $\mathcal{B}$ chooses a d_r degree polynomial q_r such that $q_r(0) = sk_j/sk_i = (ax_j)/(ax_i) = x_j/x_i$, since $\mathcal{B}$ knows x_j, x_i, he can compute $q_r(0)$ correctly. $\mathcal{B}$ can compute the re-encryption $rk_{i,\mathcal{T},j}$ as the re-encryption key generation algorithm.

Thus, $\mathcal{B}$ can derive a valid re-encryption key $rk_{i,\mathcal{T},j}$ for $\mathcal{T}$.

(iv) First level decryption: on input (pk_j, CT_j) to $\mathcal{O}_{d_1}$, where $CT_j = (C_2, C'_3, C_4, C_5, C_6,)$, $\mathcal{B}$ first verifies if $\mathcal{V}(C_2, (C_4, C_5, C_6), \mathcal{S}) \stackrel{?}{=} 1$ and $e(C_4, g_2^{C_2} g_3) \stackrel{?}{=} e(h, C_6)$, $\mathcal{B}$ abort and output '$\perp$' if the verifies fail. Otherwise, if $j \in CU$, it means that $sk_j = x_j$, and $\mathcal{B}$ return $Dec1(sk_j, CT_j)$ to $\mathcal{A}$. Otherwise, if $j \in HU$, proceeds as follows:

(1) $\mathcal{B}$ checks $C_2 \neq svk^*$, and abort if this check fails. Output '$\perp$'.

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

(2) Compute $A_1^s = (C_6/C_4^{\alpha_2/\alpha_0})^{1/\alpha_1(svk-svk^*)}$, and then $K = e(A_1^s, A_{-1}) = e(g,g)^s$.
(3) If $F[K, C_4]_{l-k} \neq [C_5]_{l-k}$, output '$\perp$' indicating an invalid ciphertext. Otherwise, output $m = F[K, C_4]^k \oplus [C_5]^k$.

(v) Second level decryption: on input (pk_i, CT_i) to $\mathcal{O}_{d_2}$, where $CT_i = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, S)$, $\mathcal{B}$ first checks whether the equations (1–4) hold, if not, $\mathcal{B}$ abort and output '$\perp$'. Otherwise, if $i \in CU$, it means that $sk_i = x_i$, and $\mathcal{B}$ return $Dec1(sk_i, CT_i)$ to $\mathcal{A}$. Otherwise, if $i \in HU$, $\mathcal{B}$ proceeds as follows:
(1) $\mathcal{B}$ checks $C_2 \neq svk^*$, and abort if this check fails. Output '$\perp$'.
(2) Compute $A_1^s = (C_6/C_4^{\alpha_2/\alpha_0})^{1/\alpha_1(svk-svk^*)}$, and then $K = e(A_1^s, A_{-1}) = e(g,g)^s$.
(3) If $F[K, C_4]_{l-k} \neq [C_5]_{l-k}$, output '$\perp$' indicating an invalid ciphertext. Otherwise, output $m = F[K, C_4]^k \oplus [C_5]^k$.

(vi) Re-encryption query $(pk_i, pk_j, \mathcal{T}, CT_i)$: on input a second level ciphertext $CT_i = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, S, C_8)$ under the keyword set W, $\mathcal{B}$ first make sure that $(pk_i, pk_j, \mathcal{T}, CT_i) \neq (pk_{i^*}, pk_j, \mathcal{T}, CT_{i^*})$ if $\mathcal{T}(W^*) = 1$ and $j \in CU$; otherwise, $\mathcal{B}$ abort and output $\perp$.

Next $\mathcal{B}$ first check whether equations (1–4) hold, if not, output '$\perp$'. If the above condition is met, $\mathcal{B}$ has to distinguish the following several situations:

(1) If $i = i^*, j \in CU$ and $\mathcal{T}(W^*) = 0$. $\mathcal{B}$ checks $C_2 \neq svk^*$, and abort if this is false. Otherwise, computes $A_2^s = C_4^{1/\alpha_0}$, and $C_3' = e(g, A_2^s)^{x_{i^*}} = e(pk_{i^*}, g)^s$. Output $CT_j = (C_2, C_3', C_4, C_5, C_6, S)$ as the re-encrypted ciphertext.
(2) Else, since $\mathcal{B}$ can compute the re-encryption key $rk_{i,W,j}$, so $\mathcal{B}$ can re-encrypted it correctly.

(vii) Challenge. Once $\mathcal{A}$ decided that Phase 1 is over, it outputs two equal length plaintext (m_0, m_1). Let $s^* = b/a^2$, $\mathcal{B}$ picks a random $\beta \in \{0,1\}$, and responds as follows:

$$
\begin{aligned}
C_1^* &= W^*,\\
C_2^* &= svk^*,\\
C_3^* &= (B)^{x_{i^*}}, = A_2 x_{i^*})^{b/a^2} = pk_i^{s^*}\\
C_4^* &= (B)^{\alpha_0} = A_2\alpha_0)^{b/a^2} = h^{s^*},\\
C_5^* &= [F(T, C_4^*)]_{l-k} \parallel ([F(T, C_4^*)]^k \oplus m_\beta),\\
C_6^* &= B^{\alpha_2} = (A_2^{svk^*} g_3)^{s^*},\\
C_7^* &= B^{x_{i^*} H_1(\omega)} = (A_2^{x_{i^*} H_1(\omega)})^{b/a^2} = H_1(pk_{i^*}, \omega)^{s^*},\\
S^* &= \mathcal{S}(ssk^*, (C_4^*, C_5^* C_6^*)).
\end{aligned}
$$

Observe that, if $T = e(g,g)^{b/a^2}$, CT_{i^*} is indeed a valid challenge ciphertext under public key pk_{i^*}. On the other hand, when T is uniform in G_2, the challenge ciphertext CT_{i^*} is independent of β in the adversary's view.

(viii) Query Phase 2. $\mathcal{A}$ continues making queries as in the Query Phase 1 with the restrictions described in the IND-L2-CCA game.

(ix) Guess. $\mathcal{A}$ outputs the guess β', if $\beta = b'$, then output 1 meaning $T = e(g,g)^{b/a^2}$; else output 0 meaning $T = e(g,g)^r$.

Probability analysis. First, we define an event F_{OTS} and bound its probability to occur. Let $CT_{i^*} = (C_1^*, svk^*, C_3^*, C_4^*, C_5^*, C_6^*, C_7^*, S^*)$ denote the challenge ciphertext given to $\mathcal{A}$ in the game. Let F_{OTS} be the event that, at some point, $\mathcal{A}$ issues a first level decryption $CT_i = (svk^*, C_3', C_4, C_5, C_6, S)$, second level decryption and re-encryption query $CT_i = (C_1, svk^*, C_3, C_4, C_5, C_6, C_7, S)$, where $(C_4, C_5, C_6, S) \neq (C_4^*, C_5^*, C_6^*, S^*)$ but $\mathcal{V}(S, svk, (C_4, C_5, C_6)) = 1$. Before the challenge is given, $\mathcal{A}$ has a $q_O \cdot \delta(\delta$ does not exceed $1/p$, q_O denotes the total number of such queries) chance of querying each oracle on a ciphertext with $C_2 = svk^*$, After the challenge is give, F_{OTS} clearly gives rise to an algorithm breaking the strong unforgeability of the one-time signature. Therefore we have $\Pr[F_{OTS}] \leq q_O/p + Adv^{OTS}$.

It is clear that the simulation of oracles $\mathcal{O}_{HU}, \mathcal{O}_{CU}$ are perfect. We define abort be the event of $\mathcal{B}'s$ aborting during the simulation of oracle $\mathcal{O}_{rk}$. Let $\Pr[i \in CU] = \theta$, we have $\Pr[\neg Abort] \geq \theta^{q_{rk}}(1-\theta)$, which is maximized at $\theta_{opt} = q_{rk}/(1+q_{rk})$. Using θ_{opt}, the probability $\Pr[\neg Abort]$ is at least $1/\dot{e}(1+q_{rk})$.

Let ε denotes the advantage of $\mathcal{A}$ win in the IND-leve2-CCA game, and ε' denotes the advantage of $\mathcal{A}$ break the 3-wDBDHI assumption. Combining the above results and counting for the target collision resistant of hash functions H_1 and the pseudorandomness of F, we have

$$\varepsilon' \geqslant \frac{\epsilon}{\dot{e}(1+q_{rk})} - \frac{q_O}{p} - Adv^{OTS} - Adv_{H_1,\mathcal{A}}^{TCR} - Adv_{F,\mathcal{F}}^{PRF}.$$

This completes the proof of Lemma 1. □

LEMMA 2. *Our scheme is IND-L1-CCA secure under the 3-wDBDHI assumption, assuming H_1 is a target collision resistant hash function, F is a PRF family and $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature.*

Proof. Suppose there exits a polynomial-time adversary $\mathcal{A}$ that can attack our scheme in the standard model. We built a simulator $\mathcal{B}$ that can play a 3-wDBDHI game. The adversary arbitrarily select a user i^* from the set of honest users HU and commit to the user to be attacked. In the following, we call HU be the set of honest parties, i^* be the target user, and CU be the set of corrupt parties. $\mathcal{A}$ first generates an keyword set W^* that it intends to attack.

We let the challenger set the group G_1 and G_2 with an efficient bilinear map e and a generator g of G_1. Simulator $\mathcal{B}$ inputs

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

a 3-wDBDHI instance $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, T)$ and has to distinguish $T = e(g,g)^{b/a^2}$ from a random element in G_2.

Next, $\mathcal{B}$ sets up the global parameters as follows: $\mathcal{B}$ picks a strongly unforgeable one-time signature $\mathcal{G}, \mathcal{S}, \mathcal{V}$ and runs $\mathcal{G}(1^k) \to (ssk^*, svk^*)$, and sets the generators $h = A_2^{\alpha_0}, g_2 = A_1^{\alpha_1}, g_3 = A_1^{-\alpha_1 svk^*} \cdot A_2^{\alpha_2}$ for a randomly chosen $\alpha_0, \alpha_1, \alpha_2 \xleftarrow{R} Z_p^*$. Finally, $\mathcal{B}$ samples a pairwise independent hash functions H_1, such that $H_1(pk_i, \omega) = pk_i^{H_1(\omega)}$. Without loose of generality, assume that H_1 is target collision resistant and F is a PRF family. The system parameters are $(p, g, G_1, G_2, e, h, g_2, g_3, H_1, F, l, k, \mathcal{G}, \mathcal{S}, \mathcal{V})$.

(i) Uncorrupted key: on input i to $\mathcal{O}_{HU}$, public keys of honest users $i \in HU \setminus \{i^*\}$ are defined as follows: $\mathcal{B}$ select a random $x_i \xleftarrow{R} Z_p^*$, then $\mathcal{B}$ computes $pk_i = g^{x_i}$. The target user's public key is set as $pk_{i^*} = A_1^{x_{i^*}}$ for randomly chosen $x_{i^*} \xleftarrow{R} Z_p^*$. Sends public key pk_i, pk_{i^*} to $\mathcal{A}$.

(ii) Corrupted key: on input j to $\mathcal{O}_{CU}$, public keys of corrupted user $j \in CU$ are defined as follows: $\mathcal{B}$ select a random $x_j \xleftarrow{R} Z_p$, then $\mathcal{B}$ computes $sk_j = x_j, pk_i = g^{x_j}$. Sends public and private key (sk_j, pk_j) to $\mathcal{A}$.

(iii) Re-encryption key: on input $(pk_i, \mathcal{T}, pk_j)$ to $\mathcal{O}_{rk}$, consider a query for a re-encryption key corresponding to an access tree $\mathcal{T}$, $\mathcal{B}$ first define the three procedures: *PolySat*, *PolyUnsat* and $\widetilde{PolySat}$ the same as in game 1.

Let $\Phi = sk_j/sk_i$, $g^\Phi = g^{sk_j/sk_i}$, to answer the re-encryption key query $(pk_i, \mathcal{T}, pk_j)$, $\mathcal{B}$ has to distinguish several situations:

(1) If $i \neq i^*$ and $j \neq i^*$ that means $\Phi = x_j/x_i$. Since $\mathcal{B}$ knows the private key x_i for user i and the private key x_j for user j, $\mathcal{B}$ can compute the conditional re-encryption $rk_{i,W,j}$ as the re-encryption key generation algorithm.

(2) If

$$\begin{cases} i \in HU\backslash\{i^*\},\ j = i^* & \Phi = sk_j/sk_i = ax_{i^*}/x_i,\ g^\Phi = A_1^{x_{i^*}/x_i} \\ i = i^*,\ j \in HU\backslash\{i^*\} & \Phi = sk_j/sk_i = x_j/ax_{i^*},\ g^\Phi = A_{-1}^{x_j/x_{i^*}} \\ i \in CU,\ j = i^* & \Phi = sk_j/sk_i = ax_{i^*}/x_i,\ g^\Phi = A_1^{x_{i^*}/x_i} \\ i = i^*,\ j \in CU & \Phi = sk_j/sk_i = x_j/ax_{i^*},\ g^\Phi = A_{-1}^{x_j/x_{i^*}}. \end{cases}$$

$\mathcal{B}$ first runs $\widetilde{PolyUnsat}(\mathcal{T}, W^*, g^\Phi)$ if $\mathcal{T}(W^*) = 0$, and $PolySat(\widetilde{\mathcal{T}, W^*, g^\Phi})$ if $\mathcal{T}(W^*) = 1$ to define a polynomial q_x for each node x of $\mathcal{T}$. Note that for each leaf node x of $\mathcal{T}$, $\mathcal{B}$ know q_x completely if x is satisfied, if x is not satisfied, then at least $g^{q_x(0)}$ is known. Let $\omega = keyword(x)$, the re-encryption key corresponding to each leaf node is given using its polynomial as follows.

If $\omega \in W^*$:

$$a_x = g^{q_x(0)} \cdot H_1(pk_i, \omega)^{r_x}, \quad b_x = pk_i^{r_x},$$

where r_x is chosen randomly in Z_p.

If $\omega \notin W^*$:

$$\begin{aligned} a_x &= \prod_{x' \in S_x, i = index(x')} g^{q_x(i)\Delta_{i,S_x'}(0)} \cdot (A_1^{x_{i^*}/x_i})^{\Delta_{0,S_x'}(0)} \cdot H_1(pk_i, \omega)^{r_x} \\ &= \prod_{x' \in S_x, i = index(x')} g^{q_x(i)\Delta_{i,S_x'}(0)} \cdot (g^\Phi)^{\Delta_{0,S_x'}(0)} \cdot H_1(pk_i, \omega)^{r_x} \\ &= g^{q_x(0)} \cdot H_1(pk_i, \omega)^{r_x}, \\ b_x &= pk_i^{r_x}, \end{aligned}$$

where r_x is randomly chosen in Z_p^*.

Thus, $\mathcal{B}$ can derive a valid re-encryption key $rk_{i,\mathcal{T},j}$ for $\mathcal{T}$.

(iv) First level decryption: on input (pk_j, CT_j) to $\mathcal{O}_{d_1}$, where $CT_j = (C_2, C_3', C_4, C_5, C_6, S)$, $\mathcal{B}$ first verifies if $\mathcal{V}(C_2, (C_4, C_5, C_6), \mathcal{S}) \stackrel{?}{=} 1$ and $e(C_4, g_2^{C_2} g_3) \stackrel{?}{=} e(h, C_6)$, $\mathcal{B}$ abort and output '$\perp$' if the verify fails. Otherwise, if $j \neq i^*$, it means that $sk_j = x_j$, and $\mathcal{B}$ return $Dec1(sk_j, CT_j)$ to $\mathcal{A}$. Otherwise, if $j = i^*$, it means that $pk_j = ax_{i^*}$, $\mathcal{B}$ proceeds as follows:

(1) $\mathcal{B}$ checks $C_2 \neq svk^*$, and abort if this check fails. Output '$\perp$'.

(2) Compute $A_1^s = (C_6/C_4^{\alpha_2/\alpha_0})^{1/\alpha_1(svk - svk^*)}$, and then $K = e(A_1^s, A_{-1}) = e(g,g)^s$.

(3) If $F[K, C_4]_{l-k} \neq [C_5]_{l-k}$, output '$\perp$' indicating an invalid ciphertext. Otherwise, output $m = F[K, C_4]^k \oplus [C_5]^k$.

(v) Challenge. Once $\mathcal{A}$ decided that Phase 1 is over, it outputs two equal length plaintexts (m_0, m_1). Let $s^* = b/a^2$, $\mathcal{B}$ picks a random $\beta \in \{0,1\}$, and responds as follows:

$$\begin{aligned} C_2^* &= svk^*, \\ C_3'^* &= e(A_{-1}, B)^{x_{i^*}} = e(A_1^{x_{i^*}}, g^b)^{1/a^2} = e(pk_{i^*}, g)^{s^*} \\ C_4^* &= B^{\alpha_0} = (A_2^{\alpha_0})^{b/a^2} = h^{s^*}, \\ C_5^* &= [F(T, C_4^*)]_{l-k} \parallel ([F(T, C_4^*)]^k \oplus m_\beta), \\ C_6^* &= (B)^{\alpha_2} = (g_2^{svk^*} g_3)^{s^*}, \\ S^* &= \mathcal{S}(ssk^*, (C_4^*, C_5^* C_6^*)), \end{aligned}$$

Observe that, if $T = e(g,g)^{b/a^2}$, CT_{i^*} is indeed a valid challenge ciphertext under public key pk_{i^*}. On the other hand, when T is uniform in G_2, the challenge ciphertext CT_{i^*} is independent of β in the adversary's view.

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

(vi) Query Phase 2. $\mathcal{A}$ continues making queries as in Query Phase 1 with the restrictions described in the IND-L2-CCA game.

(vii) Guess. $\mathcal{A}$ outputs the guess β', if $\beta = b'$, then output 1 meaning $T = e(g,g)^{b/a^2}$; else output 0 meaning $T = e(g,g)^r$.

Probability analysis. First, we define an event F'_{OTS} and bound its probability to occur. Let $CT_{i^*} = (svk^*, C_3'^*, C_4^*, C_5^*, C_6^*, S^*)$ denote the challenge ciphertext given to $\mathcal{A}$ in the game. Let F'_{OTS} be the event that, at some point, $\mathcal{A}$ issues a first level decryption $CT_i = (svk^*, C_3', C_4, C_5, C_6, S)$, where $(C_4, C_5, C_6, S) \neq (C_4^*, C_5^*, C_6^*, S^*)$ but $\mathcal{V}(S, svk, (C_4, C_5, C_6)) = 1$. Before the challenge is given, $\mathcal{A}$ has a $q_O \cdot \delta$ (δ does not exceed $1/p$, q_O denotes the total number of such queries) chance of querying each oracle on a ciphertext with $C_2 = svk^*$, After the challenge is give, F'_{OTS} clearly gives rise to an algorithm breaking the strong unforgeability of the one-time signature. Therefore, we have $\Pr[F'_{OTS}] \leq q_O/p + Adv^{OTS}$.

It is clear that the simulation of oracles $\mathcal{O}_{HU}$, $\mathcal{O}_{CU}$ and $\mathcal{O}_{rk}$ are perfect. Let ε denotes the advantage of $\mathcal{A}$ win in the IND-leve1-CCA game, and ε' denotes the advantage of $\mathcal{A}$ break the 3-wDBDHI assumption. Combining the above results and counting for the target collision resistant of hash functions H_1 and the pseudorandomness of F, we have

$$\varepsilon' \geqslant \epsilon - \frac{q_O}{p} - Adv^{OTS} - Adv^{TCR}_{H_1,\mathcal{A}} - Adv^{PRF}_{F,\mathcal{F}}.$$

This completes the proof of Lemma 2. □

4. CONCLUSIONS AND OPEN PROBLEMS

In this paper, we solve the problem left by Fang, Susilo, Ge and Wang by proposing a KP-ABPRE scheme without random oracles. Our scheme enhances the security model by making some improvements of the re-encryption key query and re-encryption query. Many interesting questions are still remaining to be solved, such as designing non-interactive KP-ABPRE schemes, and KP-ABPRE schemes without pairings.

FUNDING

This work was supported by the National Natural Science Foundation of China (Nos 61272083–61300236), the National Natural Science Foundation of Jiangsu (No. BK20130809), the National Science Foundation for Post-doctoral Scientists of China (No. 2013M530254), the National Science Foundation for Post-doctoral Scientists of Jiangsu (No. 1302137C) and the China Postdoctoral Science special Foundation (No. 2014T70518).

REFERENCES

[1] Sahai, A. and Waters, B. (2005) Fuzzy Identity-based Encryption. *Proc. Eurocrypt 2005*, Aarhus, Denmark, May 22–26, pp. 457–473. Springer, Berlin.

[2] Goyal, V., Pandey, O., Sahai, A. and Waters, B. (2006) Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. *Proc. ACM CCS 2006*, Alexandria, VA, USA, October 30–November 3, pp. 89–98. ACM, New York.

[3] Blaze, M., Bleumer, G. and Strauss, M. (1998) Divertible Protocols and Atomic Proxy Cryptography. *Proc. EUROCRYPT 1998*, Finland, May 31–June 4, pp. 127–144. Springer, Berlin.

[4] Weng, J., Deng, R.H. and Chu, C. (2009) Conditional Proxy Re-encryption Secure Against Chosen-ciphertext Attack. *Proc. 4th Int. ACM Symp. Information, Computer and Communications Security 2009*, Sydeny, Australia, March 10–12, pp. 322–332. ACM, New York.

[5] Ateniese, G., Fu, K., Green, M. and Hohenberger, S. (2005) Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. *Proc. 12th Annual Network and Distributed System Security Symp. 2005*, San Diego, CA, USA, February 3–4, pp. 29–44. ACM, New York.

[6] Liang, X., Cao, Z., Lin, H. and Shao, J. (2009) Attribute-Based Proxy Re-Encryption with Delegating Capabilities. *Proc. ASI-ACCS 2009*, Sydney, Australia, March 10–12, pp. 276–286. ACM, New York.

[7] Liang, K.T., Fang, L., Wong, D.S. and Susilo, W. (2013) A Ciphertext-Policy Attribute-Based Proxy Re-Encryption with Chosen-Ciphertext Security. *Proc. 5th Int. Conf. Intelligent Networking and Collaborative Systems 2013*, Xi'an, China, September 9–13, pp. 552–559. IEEE Computer Society, Washington, DC, USA.

[8] Fang, L.M., Susilo, W., Ge, C. and Wang, J.D. (2011) Interactive conditional proxy re-encryption with fine grain policy. *J. Syst. Softw.*, **84**, 2293–2302.

[9] Green, M. and Ateniese, G. (2007) Identity-based Proxy Re-encryption. *Proc. ACNS 2007*, Zhuhai, China, June 5–8, pp. 288–306. Springer, Berlin.

[10] Canetti, R., Goldreich, O. and Halevi, S. (1998) The Random Oracle Methodology, Revisited. *Proc. 30th Annual ACM Symp. Theory of Computing, STOC 1998*, Texas, USA, May 23–26, pp. 209–218. ACM, New York.

[11] Boneh, D. and Boyen, X. (2004) Efficient Selective-ID Based Encryption Without Random Oracles. *Proc. EUROCRYPT 2004 Interlaken*, Switzerland, May 2-6, pp. 223–238. Springer, Berlin.

[12] Boneh, D. and Franklin, M. (2001) Identity-based Encryption from the Weil Pairing. *Proc. CRYPTO 2001*, Santa Barbara, CA, USA, August 19–23, pp. 231–229. Springer, Berlin.

[13] Weng, J., Chen, M., Yang, Y., Deng, R., Chen, K. and Bao, F. (2010) CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. *J. Sci. China Inf. Sci.*, **53**, 593–606.

[14] Canetti, R., Halevi, S. and Katz, J. (1999) Chosen-ciphertext Security from Identity-based Encryption. *Proc. PKC 1999*, Kamakura, Japan, May 21–23, pp. 53–68. Springer, Berlin.

[15] Canetti, R. and Hohenberger, S. (2007) Chosen-ciphertext Secure Proxy Re-encryption. *Proc. 14th ACM Conf. Computer and Comuniation, Security 2007*, October 29-31–November 1, pp. 185–194. ACM, New York.

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016

[16] Libert, B. and Vergnaud, D. (2008) Unidirectional Chosen-ciphertext Secure Proxy Re-encryption. *Proc. PKC 2008*, Barcelona, Spain, March 9–12, pp. 360–379. Springer, Berlin.

[17] Fujisaki, E. and Okamota, T. (2004) How to Enhance the Security of Public-key Encryption at Minimum Cost. *Proc. Eurocrypt 2004*, Interlaken, Switzerland, May 2–6, pp. 207–222. Springer, Berlin.

[18] Shao, J. (2008) *Proxy Re-cryptography Revisited (in Chinese)*. Academic Journal Online Publication. Integrated Database.

Downloaded from http://comjnl.oxfordjournals.org/ at University of Electronic Science and Technology of China on September 17, 2016