A Homomorphic Universal Re-encryptor for Identity-based Encryption

Liang Liu[1] and Jun Ye[2,3]

(Corresponding author: Jun Ye)

State Key Laboratory of Integrated Service Networks (ISN), Xidian University[1]

No. 2 South Taibai Road, Yanta District, Xi'an, Shaanxi 710071, China

Artificial Intelligence Key Laboratory of Sichuan Province, Sichuan University of Science & Engineering[2]

Sichuan Province University Key Laboratory of Bridge Non-destruction, Detecting and Engineering Computing[3]

No. 180, School Street, Huixing Road, Sichuan 643000, China

(Email: yejun@suse.edu.cn)

(Received Oct. 23, 2015; revised and accepted Dec. 7 & Dec. 30, 2015)

Abstract

Re-encryption (or proxy re-encryption) is a very useful cryptographic primitive which is able to transform a ciphertext under one public key into a new ciphertext encrypting the same message but under another different public key. It plays an important role in modern secure communication and information exchange via various kinds of network infrastructure. In addition to traditional public-key encryption scheme, re-encryption can also come into force in other cryptosystems like Identity-Based Encryption (IBE) and more advanced Functional Encryption (FE), making the enhanced schemes more powerful as well as easy-to-use. In this work, we have proposed a novel identity-based proxy re-encryption (IBPRE) scheme which to the maximum extent reduces the workloads in the user side by delivering the re-encryption key (RK) generation work to the proxy server. Besides, it is likewise able to prevent possible bottlenecks for the users, like re-encryption key management.

Keywords: Fully homomorphic encryption, identity-based encryption, proxy re-encryption, re-encryptor

1 Introduction

Cryptographic primitives supporting intermediate transformations from one object (ciphertext or signature) to another without leakage of sensitive information have found their irreplaceable places in modern Internet era. The most common application of these primitives may be proxy re-encryption in e-mail relay. A senior manager Alice wants to forward an encrypted e-mail from the executive level of the enterprise to her subordinate Bob. Of course, she is able to decrypt the encrypted e-mail by her private key SK_{Alice} and then encrypts the plaintext under Bob's public key PK_{Bob} to obtain the corresponding encrypted e-mail, which will then be sent to Bob. On the surface, it seems that this method actually achieves the goal. However, we argue that this trivial solution results in several shortcomings. The most obvious point is that the initial ciphertext owner Alice must execute all these computations herself - including decryption and encryption - to produce the ciphertext for Bob. In some scenarios, these computation workloads are awfully cumbersome.

Another way to solve the above mentioned issue may be to introduce a proxy to accomplish those transformation workloads on behalf of the manager Alice, who has usually been named as *delegator* in the proxy re-encryption/re-signature scheme. Accordingly, the term for the proxy is *delegatee*, the role of which is commonly played by a more powerful server. Then if the delegator Alice wants to largely reduce her computation workloads caused by decryption and encryption, she can generate a re-encryption key $RK_{\text{Alice}\to\text{Bob}}$ by which anyone is capable of transforming a ciphertext c_{Alice} under her public key to the corresponding ciphertext c_{Bob} under Bob's public key without any plaintext leakage or private key infringement. Nevertheless, this approach also suffers from some problems, which we will detail later.

Identity-based encryption is a useful as well as powerful primitive envisioned by Shamir [24] in 1984. But due to the lag of mathematical tools, the first scheme based on bilinear maps was proposed by Dan Boneh [2] after 17 years. IBE is more natural and convenient for the system users because when one user U_1 wishes to encrypt a message to another user U_2, she needs not to know the public key of U_2. In contrast, she can just encrypt the message she wishes to send under a publicly accessible identity corresponding to U_2. This tremendously cuts down the work from the management of user certificates. So extending the re-encryption techniques to identity-based encryption schemes and other advanced encryption schemes [3, 11] is reasonable and necessary.

1.1 Motivation

In the traditional research works on (proxy) re-encryption from former literatures, researchers use re-encryption in such a scenario: Alice and Bob are legal users within a cryptographic system with their own public-private key pairs. In the real-life scenario, this cryptography system may be used for an e-mail server or a cloud storage service provider. Alice wants to grant her e-mail server some privilege to forward e-mails to Bob for her convenience, otherwise every time she must decrypt the e-mail and then encrypt it under Bob's public key/identity. As we have stated above, re-encryption is essential in this scenario.

But unfortunately, in all of the former literatures, the authors only consider the scenarios in which re-encryptions happen between only two users. Although some schemes support multi-use (multi-hop) re-encryption, the inputs for generating re-encryption keys are not the same. For instance, the inputs for generating $RK_{U_1 \to U_2}$ (usually) including $\{SK_{U_1}, PK_{U_2}, id_{U_1}, id_{U_2}\}$ for user U_1 and U_2 are different from the inputs for generating $RK_{U_3 \to U_4}$ which usually includes $\{SK_{U_3}, PK_{U_4}, id_{U_3}, id_{U_4}\}$ for U_3 and U_4. This undoubtedly results in some inconvenience like re-encryption key management in the user or the RK generator side. Can we extract some common part(s) of those inputs for different RKs? In addition, inconvenience also exists in other aspects. For instance, the user may only trust her key generation server (key generator) but not the proxy; or a certain user U_1 may need to generate plenty of re-encryption keys for lots of other receivers $U_2, \ldots, U_k$, which may be a bottleneck for U_1 both in computation and in storage. All the above mentioned problems can be solved via such avenue: The key generator delegates his master secrete key MSK to the proxy without any privacy leakage about MSK. Then the proxy must have some special techniques to deal with the RK generation requirements between any pair of users. For the performance bottleneck of one specific user, since the RK generation workloads are all in the proxy server side, the possible busy user is then liberated from massive RK generation workloads.

In addition to email forwarding, there are a great many of other application scenarios for re-encryption in real life. To just name a few:

1) **Key revocation and key update.** Re-encryption is the mainstream technique and one of the most indispensable building blocks in schemes supporting key revocation (update). In such a typical scheme, after the key revocation procedure, the existing ciphertexts must be updated accordingly, involving tens of millions of existing ciphertexts. Without re-encryption, this ciphertext update procedure is definitely unbearable.

2) **Restricted law enforcement.** This is an example from [19] in which a law enforcement agency F wishes to scrutinize classified personal files of a set of suspect individuals G during a certain period of time. However, the legal court possessing all the keys cannot directly pass those keys to F, otherwise it will permanently obtain the privilege to infringe the privacy of these citizens. A plausible method is to let the court transform the ciphertext (with or without the help of a proxy) under a certain person's public key to the ciphertext under F's public key when F has already been granted a warrant. After the warrant loses its effect, F will lose its ability to probe into the classified files immediately.

3) **Fine-grained access control.** Suppose an accountant Alice at Department A before was a checker at Department B within the same corporation. Her private key corresponding to her identity is not only associated with her name "Alice" but also with her job title, *e.g.*, "Alice||Accountant" or "Alice||Checker". Then re-encryption allows for fine-grained access control so that she is able to deal with some of her final stage work after she leaved Department B.

1.2 Our Contribution and Main Technique

Our first contribution is to formalize the notion of homomorphic universal re-encryptor for identity-based encryption (HURE-IBE for short). This primitive is very useful in a scenario where the users only have limited computation or storage capability as we have explained above. Besides, the universality property features the advantage that a user can gain her re-encryption key quite easily by just issuing a re-encryption key query containing only two id's and the proxy does not need to deal with any public key or private key relevant information. This is due to the delegation of part of the master secret key s in the encrypted form under a fully homomorphic encryption (FHE) scheme.

Our scheme can not only provide solutions to practical issues, but also pave the way for follow-up research works.

Another contribution of this work is to put forward the first HURE-IBE scheme by combining the IBPRE scheme with an FHE scheme [4, 13]. In addition, although the proposed scheme in the construction part is instantiated via Boneh-Franklin IBE scheme, our construction is essentially generic. This means our general methodology can be applied to other IBPRE schemes, even other public-key proxy re-encryption schemes, subject to a condition that there must be a master secret key for the generation of all these user private keys.

To solve all those problems above mentioned, we have developed a novel technique to efficiently combine the possible IBPRE scheme and FHE scheme. A small shortcoming is that the introduction of FHE would decrease the efficiency of our scheme. But we argue that, on one hand, the scheme is more convenient than previous ones at the cost of a little more computation workloads; on the other hand, the state of the art FHE scheme is fast enough to bear the extra cost [23].

1.3 Related Works

Mambo *et al.* [21] first found the usefulness of re-encryption and suggested to use proxy cryptosystem to replace the trivial decrypt-and-encrypt method for sake of efficiency. Then proxy re-encryption and re-signature were conceptualized from a primitive named as atomic proxy function, which was coined by Blaze *et al.* [1]. Due to its wide range of application fields, proxy re-encryption has received much attention after its birth. For instance, since the earlier scheme for re-encryption proposed by Blaze *et al.* is inherently bidirectional, there are quite a number of works have focused on unidirectional schemes [19, 20]. For the security of the encryption schemes, chosen ciphertext attack (CCA) security is very important. Therefore there are also several works on CCA-secure PRE schemes [5, 16]. In addition, research on proxy signature schemes [18, 25] is also very active in cryptographic community.

Due to the significance, powerfulness and convenience of more advanced encryption schemes like identity-based encryption, attributed-based encryption [11], and functional encryption [3], re-encryption has also been developed along this line [6, 9, 14, 22]. Moreover, in recent years, since program obfuscation has a fast progress in several aspects, *e.g.*, extremely powerful constructions like indistinguishability obfuscation (iO) [10] has been developed, using obfuscation techniques to enhance re-encryption schemes is also a promising field. Besides those program obfuscation constructions for general programs/functions, special constructions also find their places due to their high efficiency. So the research field on re-encryption obfuscation is also very active. The first re-encryption obfuscation scheme is due to Hohenberger *et al.* [17], who had also proposed a new security definition framework for average-case secure obfuscation in order to bypass the limitations of obfuscating deterministic circuits. Then Hada [15] proposed obfuscation scheme for encrypted signatures. Chandran *et al.* [7] introduced collusion-resistant obfuscation to construct functional re-encryption scheme supporting function F's with a polynomial-size domain.

2 Preliminaries

Definition 1 (Identity-based proxy re-encryption)**.** *An identity-based proxy re-encryption (IBPRE) scheme* Π_{RE} *consists of six probabilistic polynomial time (PPT) algorithms:*

- **Setup**$(n) \rightarrow (params, MSK)$: On input a security parameter n, the algorithm outputs the public parameters $params$ and the master secret key MSK, which should be kept secret. This algorithm may also decide the maximum encryption level of the cryptosystem under some conditions. This algorithm is run by the trust authority.

- **KeyGen**$(params, MSK, id) \rightarrow SK_{id}$: On input an identity $id \in \{0,1\}^*$, the master secret key MSK, and the public parameters $params$, the algorithm outputs a user secret key (decryption key) SK_{id} corresponding to the identity id. This algorithm is run by the trust authority.

- **Enc**$(params, id, m) \rightarrow c_{id}$: On input a plaintext $m \in \mathcal{M}$, an identity id, and the public parameters $params$, the algorithm outputs a ciphertext c_{id} which corresponds to the plaintext m and the identity id. This algorithm is run by the users.

- **Dec**$(params, SK_{id}, c_{id}) = m$: On input a ciphertext c_{id} which is the encryption of the plaintext m and the identity id, a user secret key SK_{id}, and the public parameters $params$, the deterministic algorithm outputs m if decryption succeeds; otherwise, it outputs $\perp$. This algorithm is run by the users.

- **RKGen**$(params, SK_{id_1}, id_1, id_2) \rightarrow RK_{id_1 \rightarrow id_2}$: On input two id's id_1, id_2, a user secret key SK_{id}, and the public parameters $params$, the algorithm outputs a re-encryption key $RK_{id_1 \rightarrow id_2}$. This algorithm may be run by the users or the proxy (server).

- **ReEnc**$(params, RK_{id_1 \rightarrow id_2}, c_{id_1}) \rightarrow c_{id_2}$: On input a ciphertext c_{id_1} under identity id_1, a re-encryption key $RK_{id_1 \rightarrow id_2}$, and the public parameters $params$, the algorithm outputs a re-encrypted ciphertext c_{id_2} under identity id_2. This algorithm may be run by the users or the proxy (server).

Remark 1. *Our scheme is similar but not consistent with the traditional descriptions. Especially, we will sometimes omit* $params$ *and treat it as an implicit input.*

Definition 2 (Fully homomorphic encryption)**.** *A homomorphic encryption scheme* Π_{HE} *consists of four PPT algorithms:*

- **KeyGen**$_{\text{HE}}(n) \rightarrow (PK_{\text{HE}}, SK_{\text{HE}})$: This is a randomized algorithm which takes as input a security parameter n, and outputs a public key PK_{HE} and a secret key SK_{HE}.

- **Enc**$_{\text{HE}}(PK_{\text{HE}}, m) \rightarrow \text{CT}_{\text{HE}}(m)$: This is a randomized algorithm which takes as input a public key PK_{HE}, and a message $m \in \mathcal{M}$. It returns a ciphertext $\text{CT}_{\text{HE}}(m)$ as its output.

- **Dec**$_{\text{HE}}(SK_{\text{HE}}, \text{CT}_{\text{HE}}(m)) = m$: This is a deterministic algorithm which takes as input a secret key SK_{HE}, and a ciphertext $\text{CT}_{\text{HE}}(m)$. It returns the corresponding plaintext m if the decryption succeeds, and $\perp$ otherwise.

- $\mathbf{Eval}_{\mathrm{HE}}(PK_{\mathrm{HE}}, \Delta(\cdot), \mathrm{CT}_{\mathrm{HE}}(m_1), \ldots, \mathrm{CT}_{\mathrm{HE}}(m_k)) \to \mathrm{CT}_{\mathrm{HE}}(\Delta(m_1, \ldots, m_k))$: This is a randomized algorithm which takes as input a public key PK_{HE}, a circuit $\Delta(\cdot)$, and a bunch of ciphertexts $\{\mathrm{CT}_{\mathrm{HE}}(m_1), \ldots, \mathrm{CT}_{\mathrm{HE}}(m_k)\}$. It outputs a ciphertext $\mathrm{CT}_{\mathrm{HE}}(\Delta(m_1, \ldots, m_k))$ which is the encryption of the circuit output on inputs $m_1, \ldots, m_k$.

Remark 2. *In this work, we use fully homomorphic encryption as a black box, just like the way in some verifiable computation schemes [8, 12] for general circuits. For verifiable computation schemes for special functions like polynomial function [26], (proxy) signature scheme other than FHE might play a more significant role. Besides, for ease of description, we will sometimes omit the FHE public key PK_{HE} in some of the algorithms and notations, e.g., a ciphertext $\boldsymbol{Enc}_{HE}(PK_{HE}, m)$ of the homomorphic encryption scheme is equivalent to $CT_{HE}(m)$ in this work.*

Definition 3 (Bilinear maps)**.** *Let $\mathcal{G}$ be an algorithm which takes as input a security parameter n and outputs a tuple $(\hat{e}, q, g, \mathbb{G} = \langle g \rangle, \mathbb{G}_T = \langle \hat{e}(g, g) \rangle)$ where q is a large prime numbers, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order q. A bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has the following properties:*

1) **Bilinearity**: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
2) **Non-degeneracy**: If g generates $\mathbb{G}$, then $\hat{e}(g, g)$ generates $\mathbb{G}_T$.
3) **Efficiency**: Group operations in $\mathbb{G}$ and the bilinear map $\hat{e}$ are both computable in polynomial time.

Assumption 1 (Decisional Bilinear Diffie-Hellman (DBDH))**.** *The DBDH assumption says that the following two tuples are computationally indistinguishable.*

$$\{g, g^a, g^b, g^c, T = \hat{e}(g, g)^{abc}\} \overset{c}{\equiv} \{g, g^a, g^b, g^c, T \xleftarrow{\$} \mathbb{G}_T\}$$

3 IBPRE Constructions

3.1 Our Main IBPRE Scheme

Our main improvement on the traditional IBPRE scheme is to remove the possible burdensome RK generation workloads to the proxy whose role has usually been played by powerful cloud servers, and further cut down the total workloads. The main idea behind our scheme is that we use fully homomorphic encryption scheme to protect the master secret s and at the meantime to permit the required computations through the evaluation algorithm $\mathbf{Eval}_{\mathrm{HE}}$ of the FHE scheme. The construction methodology of ours is similar to the general verifiable computation protocol from Yao's Garbled Circuit and FHE proposed by Gennaro *et al.* [12]. More directly speaking, we use FHE to ensure the privacy and reusability of the master secret key. Then the proxy is able to securely and privately execute the RK generation procedures to respond to the RK generation requirements from the users without any privacy infringement of the master secret key.

We now provide a formal description of our scheme. Note that the common part of four algorithms are very similar to a common Boneh-Franklin IBE scheme except that there is an FHE system embedded in ours.

- $\mathbf{Setup}(n) \to (params, MSK)$: The **Setup** algorithm generates a bilinear map system $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T = \langle \hat{e}(g, g) \rangle$ are both cyclic groups of order q, which is a large prime implicitly decided by the security parameter n. The algorithm also chooses two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \{0,1\}^* \to \mathbb{G}$, as well as a fully homomorphic encryption scheme $\Pi_{\mathrm{HE}} = \{\mathbf{KeyGen}_{\mathrm{HE}}, \mathbf{Enc}_{\mathrm{HE}}, \mathbf{Dec}_{\mathrm{HE}}, \mathbf{Eval}_{\mathrm{HE}}\}$ with a pair of FHE keys $(PK_{\mathrm{HE}}, SK_{\mathrm{HE}})$. In addition, the algorithm chooses a secret number s uniformly at random from $\mathbb{Z}_q^*$ as a part of the master secret key MSK, namely $s \xleftarrow{\$} \mathbb{Z}_q^*$. The public parameters are
$$params = \left(\hat{e}, \mathbb{G}, \mathbb{G}_T, g, g^s, H_1, H_2, \Pi_{\mathrm{HE}}, PK_{\mathrm{HE}}\right)$$
and the master secret key is $MSK = (s, SK_{\mathrm{HE}})$. Besides, $\mathrm{CT}_{\mathrm{HE}}(s)$ and SK_{HE} will be sent to the proxy and the users, respectively.

- $\mathbf{KeyGen}(params, MSK, id) \to SK_{id}$: The **KeyGen** algorithm outputs a user secret key $SK_{id} = H_1(id)^s$ for the input identity $id \in \{0,1\}^*$.

- $\mathbf{Enc}(params, id, m) \to c_{id}$: In order to encrypt a message m under an identity id, the **Enc** algorithm selects $r \xleftarrow{\$} \mathbb{Z}_q^*$ uniformly at random and outputs the ciphertext
$$c_{id} = (C_1, C_2) = (g^r, m \cdot \hat{e}(g^s, H_1(id))^r)$$

- $\mathbf{RKGen}(params, \mathrm{CT}_{\mathrm{HE}}(s), id_1, id_2) \to RK_{id_1 \to id_2}$: To generate a RK from id_1 to id_2, the **RKGen** algorithm must generate three parts. It first homomorphically runs the FHE evaluation algorithm to conduct the following evaluation on circuit Δ_1 and the three FHE ciphertexts:
$$\begin{aligned}
&\mathbf{Eval}_{\mathrm{HE}}\Big(\Delta_1, \mathrm{CT}_{\mathrm{HE}}(id_1), \mathrm{CT}_{\mathrm{HE}}(s), \mathrm{CT}_{\mathrm{HE}}(id_2)\Big)\\
=&\mathbf{Enc}_{\mathrm{HE}}\Big(PK_{\mathrm{HE}}, SK_{id_1}^{-1} \cdot H_2(\mathbf{str}(K_{id_1 id_2})||id_1 \to id_2)\Big)\\
=&\mathbf{Enc}_{\mathrm{HE}}\Big(PK_{\mathrm{HE}}, SK_{id_1}^{-1} \cdot\\
&\qquad H_2(\underbrace{\mathbf{str}(\hat{e}(H_1(id_1), H_2(id_2))^s)||id_1 \to id_2}_{\mathrm{Const}_{id_1 \to id_2}})\Big)\\
=&\mathrm{CT}_{\mathrm{HE}}(SK_{id_1}^{-1} \cdot H_2(\underbrace{\mathrm{Const}_{id_1 \to id_2}}_{X}))
\end{aligned}$$

Here some notations in the above formula should be more detailedly explained. $\mathbf{str}(\cdot)$ is a function that outputs the bit-string representation of its input. And below we will use a simpler notation X to denote $\text{Const}_{id_1 \to id_2}$. Besides, we will illustrate circuit Δ_1 as the following 3-ary function.

$$\begin{aligned}\Delta_1(x_1, x_2, x_3) = (H_1(x_1)^{x_2})^{-1} \cdot \\ H_2(\mathbf{str}(\hat{e}(H_1(x_1), H_2(x_3))^{x_2} \| x_1 \to x_3))\end{aligned}$$

That is to say, let the three inputs x_1, x_2, x_3 be id_1, s, id_2 respectively, we have

$$\begin{aligned}\Delta_1(id_1, s, id_2) = (H_1(id_1)^s)^{-1} \cdot \\ H_2(\mathbf{str}(\hat{e}(H_1(id_1), H_2(id_2))^s \| id_1 \to id_2)) \\ = SK_{id_1}^{-1} \cdot H_2(\text{Const}_{id_1 \to id_2}) \\ = SK_{id_1}^{-1} \cdot H_2(X)\end{aligned}$$

The other two parts can be generated by simple IBE encryption operations on X and then the resulting two parts of the ciphertext are encrypted by the FHE algorithm $\mathbf{Enc}_{\text{HE}}$. So the final re-encryption key from id_1 to id_2 is

$$RK_{id_1 \to id_2} = \big(\text{CT}_{\text{HE}}(RK_1), \text{CT}_{\text{HE}}(RK_2), \text{CT}_{\text{HE}}(RK_3)\big)$$

where we have,

$$\begin{cases} \text{CT}_{\text{HE}}(RK_1) = \text{CT}_{\text{HE}}(g^{r'}) \\ \text{CT}_{\text{HE}}(RK_2) = \text{CT}_{\text{HE}}(X \cdot \hat{e}(g^s, H_1(id_2))^{r'}) \\ \text{CT}_{\text{HE}}(RK_3) = \text{CT}_{\text{HE}}(SK_{id_1}^{-1} \cdot H_2(X)) \end{cases}$$

Besides, we also denote the first two components of the RK, $\text{CT}_{\text{HE}}(RK_1)$ and $\text{CT}_{\text{HE}}(RK_2)$, as $\text{CT}_{\text{HE}}(\mathbf{Enc}(params, id_2, X))$ because they are actually an IBE ciphertext of X and randomness r' regardless of the FHE encryption layer.

- $\mathbf{ReEnc}(params, RK_{id_1 \to id_2}, c_{id_1}) \to c_{id_2}$: To produce a re-encrypted ciphertext for the input ciphertext c_{id_1} with the form of

$$c_{id_1} = (C_1, C_2) = (g^r, m \cdot \hat{e}(g^s, H_1(id_1))^r)$$

the **ReEnc** algorithm must generate four parts. It first runs the FHE evaluation algorithm $\mathbf{Eval}_{\text{HE}}$ on the homomorphic encryption of c_{id_1}, and the third part of the re-encryption key $RK_{id_1 \to id_2}$ to generate the second part of the re-encrypted ciphertext, namely,

$$\begin{aligned}&\mathbf{Eval}_{\text{HE}}\Big(\Delta_2, \text{CT}_{\text{HE}}(C_2), \text{CT}_{\text{HE}}(C_1), \text{CT}_{\text{HE}}(RK_3)\Big) \\ =&\mathbf{Eval}_{\text{HE}}\Big(\Delta_2, \text{CT}_{\text{HE}}(m \cdot \hat{e}(g^s, H_1(id_1))^r), \text{CT}_{\text{HE}}(g^r), \\ &\quad \text{CT}_{\text{HE}}(SK_{id_1}^{-1} \cdot H_2(X))\Big) \\ =&\mathbf{Enc}_{\text{HE}}\Big(PK_{\text{HE}}, m \cdot \hat{e}(g, H_2(X))^r\Big) \\ =&\text{CT}_{\text{HE}}\Big(m \cdot \hat{e}(g, H_2(X))^r\Big)\end{aligned}$$

The 3-ary circuit Δ_2 does the following computation:

$$\Delta_2(x_1, x_2, x_3) = x_1 \cdot \hat{e}(x_2, x_3)$$

Therefore, when the three inputs x_1, x_2, x_3 are C_2, C_1, RK_3 respectively

$$\begin{aligned}&\Delta_2(C_2, C_1, RK_3) \\ =&m \cdot \hat{e}(g^s, H_1(id_1))^r \cdot \hat{e}(g^r, SK_{id_1}^{-1} \cdot H_2(X)) \\ =&m \cdot \hat{e}(g, H_1(id_1))^{rs} \cdot \hat{e}(g^r, SK_{id_1}^{-1}) \cdot \hat{e}(g^r, H_2(X)) \\ =&m \cdot \hat{e}(g, H_2(X))^r\end{aligned}$$

For the other three parts, they can be easily obtained - actually there is no need to do further computation. The final re-encrypted ciphertext $c_{id_2}^{\text{RE}}$ is,

$$\begin{aligned}c_{id_2}^{\text{RE}} =&(C_1, C_2, C_3, C_4) \\ =&\big(g^r, \text{CT}_{\text{HE}}(m \cdot \hat{e}(g, H_2(X))^r), \text{CT}_{\text{HE}}(RK_1), \\ &\text{CT}_{\text{HE}}(RK_2)\big)\end{aligned}$$

- $\mathbf{Dec}(params, SK_{id}, c_{id}) = m$: Decryption is categorized into two types according to the ciphertext type.
  - Condition 1: If the input ciphertext is a normal IBE ciphertetx with the form of $c_{id} = (C_1, C_2) = (g^r, m \cdot \hat{e}(g^s, H_1(id))^r)$, the decryption algorithm will just do the Boneh-Franklin IBE decryption calculation, namely,

$$m = \mathbf{Dec}_{\text{IBE}}(C_1, C_2) = C_2 / \hat{e}(C_1, SK_{id})$$

  - Condition 2: If the input ciphertext is a re-encrypted ciphertext with the form of $c_{id}^{\text{RE}} = (C_1, C_2, C_3, C_4) = \big(g^r, \text{CT}_{\text{HE}}(m \cdot \hat{e}(g, H_2(X))^r), \text{CT}_{\text{HE}}(RK_1), \text{CT}_{\text{HE}}(RK_2)\big)$, the decryption algorithm will in addition invoke the FHE decryption algorithm to obtain the plaintext encrypted under PK_{HE}. More specifically, it first decrypts $\text{CT}_{\text{HE}}(RK_1)$, $\text{CT}_{\text{HE}}(RK_2)$ to obtain X. And then it uses X like a "private key" to decrypt the former two parts in the type-II ciphertext.

$$\begin{aligned}X &= \mathbf{Dec}_{\text{IBE}}(\mathbf{Dec}_{\text{HE}}(C_3), \mathbf{Dec}_{\text{HE}}(C_4)) \\ m &= \mathbf{Dec}_{\text{HE}}(C_2) / \hat{e}(C_1, H_2(X))\end{aligned}$$

3.2 Two Variants to Resist Possible FHE Key Leakage

Note that in our settings, we make a moderate assumption that a legal user will not intentionally or unintentionally leak the private homomorphic decryption key SK_{HE} to any parties, especially to the proxy. This assumption is indeed realistic since in real life the role of a proxy server is frequently played by the cloud computing units of those Internet giants like Amazon's AWS or Microsoft's Azure whose commercial reputations are of vital importance. Nevertheless, we still provide two countermeasures to cope with this issue, which give rise to two variants of our main construction. As the main aim of our scheme is to largely reduce the workload in the user side, the two variants will lose some advantages as tradeoffs compared with the main construction in a bid to achieve a higher security goal. Bellow we briefly illustrate the two countermeasures as well as their pros and cons in comparison to the scheme in Section 3.1.

- **Variant I. (Deprive of the full decryption power from a single user)**

  The most direct avenue to deal with possible private homomorphic decryption key leakage is to make a shift in the beginning that the trust authority does not share SK_{HE} to every user. Instead, it has two alternative strategies. The first one is to conduct the outermost homomorphic decryption procedures completely by itself. The main merit is that it has actually solved the FHE private key leakage problem. Nevertheless, potential decryption bottleneck in the authority side might be introduced. The second strategy is to share SK_{HE} to a threshold number of users, following the similar vein of the first strategy, namely, incapacitating a single user for her decryption ability. The shortcomings for this method is that the collaboration of more than one user is a must for conducting a decryption operation.

- **Variant II. (Introduce distinct FHE key pairs to the system for different users)**

  An alternative way to deal with possible private homomorphic decryption key leakage is decentralization. More specifically, the trust authority does not control over the homomorphic encryption system any more. On the contrary, a different homomorphic key pair per user will be used in the **RKGen** procedure to replace the global homomorphic encryption-decryption key pair. Accordingly, other related procedures like **ReEnc** should also be modified. In this way, once an FHE key pair for user U_i is compromised, the only victim is just user U_i, which rules out deliberate FHE key leakage. However, we point out that this change will also lead to inefficiency (due to the involvements of more FHE key pairs) and inconvenience.

4 Security and Efficiency

We first provide mathematical deductions for the correctness of our scheme.

Correctness. The correctness is guaranteed by the following formulas.
Condition 1:

$$C_2/\hat{e}(C_1, SK_{id}) = \frac{m \cdot \hat{e}(g^s, H_1(id))^r}{\hat{e}(g^r, H_1(id)^s)} = m$$

Condition 2:

$$\begin{aligned} &\mathbf{Dec}_{\mathrm{IBE}}(\mathbf{Dec}_{\mathrm{HE}}(C_3), \mathbf{Dec}_{\mathrm{HE}}(C_4)) \\ =&\mathbf{Dec}_{\mathrm{IBE}}(RK_1, RK_2) \\ =&RK_2/\hat{e}(RK_1, SK_{id}) \\ =&X \end{aligned}$$

$$\mathbf{Dec}_{\mathrm{HE}}(C_2)/\hat{e}(C_1, H_2(X)) = \frac{m \cdot \hat{e}(g, H_2(X))^r}{\hat{e}(g^r, H_2(X))} = m$$

Security. Then it goes to the security proof. Actually, our proof of security is mainly divided into two parts. The first part is to prove that the bare scheme Π'_{HURE} without the fully homomorphic encryption scheme Π_{HE} is secure and then prove that the full scheme is secure. In fact, due to the extensive research on FHE, we just need to prove the security of the bare scheme.

Proof. The proof is constructed by contradiction. Suppose there is an adversary $\mathcal{A}$ who has a non-negligible advantage in attacking the scheme Π'_{HURE}, then we can construct another adversary $\mathcal{A}'$ to succeed in attacking the DBDH problem with a non-negligible advantage.

Suppose $\mathcal{A}'$ receives a tuple $\langle g, g^a, g^b, g^c, T\rangle$ from the challenger $\mathcal{C}$, which may be a DBDH tuple with $T = \hat{e}(g,g)^{abc}$ or a random tuple with $T \xleftarrow{\$} \mathbb{G}_T$. To take advantage of $\mathcal{A}$, adversary $\mathcal{A}'$ must prepare parameters and respond to the queries by $\mathcal{A}$. We then illustrate how $\mathcal{A}'$ finishes these steps.

Setup: Adversary $\mathcal{A}'$ first establishes the system parameters as $params = (\hat{e}, \mathbb{G} = \langle g\rangle, \mathbb{G}_T = \langle \hat{e}(g,g)\rangle, g, g^a, H_1, H_2)$. There are two points worth mentioning. First, there is no need to include the homomorphic encryption scheme as well as the public key PK_{HE} of this scheme into $params$, since here we just consider the security of the bare scheme as stated above. Second, although the adversary does not know a, she can still use g^a to replace g^s of the original scheme, since g^a is already in the tuple. Beside, the adversary $\mathcal{A}'$ also maintains a table $\mathcal{T}$ to record the responses.

Simulate hash queries: To simulate $H_1 : \{0,1\}^* \to \mathbb{G}$, the adversary $\mathcal{A}'$ responds as follow. On an input identity id, if $id = id^*$, which is the challenge identity, the response is $h \leftarrow (g^c)^z$, where $z \xleftarrow{\$} \mathbb{Z}_q^*$; otherwise, the response is just $h \leftarrow g^z$, where $z \xleftarrow{\$} \mathbb{Z}_q^*$. To simulate $H_2 : \{0,1\}^* \to \mathbb{G}$, the adversary $\mathcal{A}'$ just returns a random element in $\mathbb{G}$. After each query, the corresponding results *e.g.*, $\{id_i, h_i, z_i\}$ must be included into the table $\mathcal{T}$. Besides, we underline that whenever an id query comes, the adversary $\mathcal{A}'$ must first look up the table $\mathcal{T}$ to find out if id has already been queried. If so, $\mathcal{A}'$ should locate the existing tuple at the table and return the corresponding content. This is also the case for key queries.

Simulate key queries: On a user secret key query for id_i, the adversary $\mathcal{A}'$ first generates the corresponding tuple $\{id_i, h_i, z_i\}$ as stated above. Subsequently, it returns $SK_{id_i} = (g^a)^{z_i}$ as the response, which will then be added up to the tuple in table $\mathcal{T}$. For a re-encryption key query $id_1 \to id_2$, if this query will lead to a trivial decryption of the challenge ciphertext, the adversary $\mathcal{A}'$ returns

$$RK_{id_1 \to id_2} = \left((g^b)^r, T^{rz_2} \cdot X, g^x\right)$$

where $r, x \xleftarrow{\$} \mathbb{Z}_q^*$ and X is computed as stated in the scheme - actually, it can be selected uniformly at random. Note that this RK is not correctly formed, but the adversary $\mathcal{A}$ can not detect this. If a RK query will not lead to a trivial decryption of the challenge ciphertext, the adversary $\mathcal{A}'$ returns

$$RK_{id_1 \to id_2} = \left(g^r, X \cdot \hat{e}(g^a, H_1(id_2))^r, (g^a)^{-z_1} \cdot H_2(X)\right)$$

another form of which is $\left(\mathbf{Enc}(id_2, X), SK_{id_1}^{-1} \cdot H_2(X)\right)$.

Formalize challenge ciphertext: In this step, the adversary $\mathcal{A}'$ receives two equal-length message m_0, m_1 as inputs from $\mathcal{A}$. To formalize the challenge ciphertext c^*, the adversary $\mathcal{A}'$ first evaluates $\{id^*, h^*, z^*\}$ if the hash query of id^* has not been issued previously. Then $\mathcal{A}'$ flips a random coin $f \in \{0, 1\}$ and returns $c^* = (g^b, T^{z^*} \cdot m_f)$.

Note that after the generation of challenge ciphertext c^*, additional queries are also permitted conditioned on two requirements: First, the total number of queries is bounded by a fixed value which we do not explicitly state here. And second, any query which will lead to a trivial decryption for c^* is prohibited.

We argue that although some components are not correctly formed, the adversary $\mathcal{A}$ can not detect this, meaning that the adversary $\mathcal{A}$ can not distinguish the simulation by $\mathcal{A}'$ without the secret s (but with g^a) from a real execution by a real challenge $\mathcal{C}$ who possesses the secret s. Therefore, from her point of view, these two procedures are computationally identical. For ease of description, just take the two kinds of RK's as an example. Since r is a random element, it is easy to see that $(g^b)^r \overset{c}{\equiv} g^r$ and it is the same for either $T^{rz_2} \cdot X$ and $X \cdot \hat{e}(g^a, H_1(id_2))^r$, as well as g^x and $(g^a)^{-z_1} \cdot H_2(X)$.

At the end of all the interactions between adversary $\mathcal{A}'$ and $\mathcal{A}$, $\mathcal{A}$ must return her guess to $\mathcal{A}'$. The latter will make a choice on whether $T = \hat{e}(g, g)^{abc}$ or $T \xleftarrow{\$} \mathbb{G}_T$ according to the guess bit f' returned by $\mathcal{A}$. If $f' = f$, namely, adversary $\mathcal{A}$ succeeds in attacking the bare scheme, then $\mathcal{A}'$ will return 1, indicating that $T = \hat{e}(g, g)^{abc}$ is belonging to a DBDH tuple; otherwise, $\mathcal{A}'$ will return 0, indicating that $T \xleftarrow{\$} \mathbb{G}_T$ is belonging to a random tuple.

To see how it works, we will discuss according to the value of T. When $T = \hat{e}(g, g)^{abc}$, the challenge ciphertext has the following form:

$$\begin{aligned} c^* &= (g^b, T^{z^*} \cdot m_f) = (g^b, \hat{e}(g, g)^{abcz^*} \cdot m_f) \\ &= (g^b, \hat{e}(g^a, g^{cz^*})^b \cdot m_f) \\ &= (g^b, m_f \cdot \hat{e}(g^a, H_1(id^*))^b) \end{aligned}$$

In other word, it is a correctly formed ciphertext. So in this game, the adversary $\mathcal{A}$ has a non-negligible advantage ϵ in distinguishing whether c^* is the encryption of m_0 or the encryption of m_1.

When $T \xleftarrow{\$} \mathbb{G}_T$, the challenge ciphertext has the following form (r is a random element selected from $\mathbb{Z}_q^*$, which makes rz^* a random element in $\mathbb{Z}_q^*$. And R is thus a random element selected from $\mathbb{G}_T$):

$$\begin{aligned} c^* &= (g^b, T^{z^*} \cdot m_f) = (g^b, \hat{e}(g, g)^{rz^*} \cdot m_f) \\ &= (g^b, m_f \cdot R) \end{aligned}$$

Since R is a random element in $\mathbb{G}_T$, the probability for adversary $\mathcal{A}'$ to successfully guess the bit f is just $\frac{1}{2}$, *i.e.*, the advantage for her is 0. So in terms of whether adversary $\mathcal{A}$ succeeds in the Π'_{HURE} game, adversary $\mathcal{A}'$ can also have a non-negligible advantage in distinguishing a DBDH tuple from a random tuple. □

Efficiency. The main advantage of our scheme is the universality property mentioned before, which has never been achieved in any other PRE schemes. To this end, we sacrifice some efficiency as the tradeoffs. Nevertheless, our scheme has gained advantages in communication complexity aspects since the user-specific private information is no longer needed to send.

Below we will provide the readers with a simplified analysis of the efficiency, namely we roughly divide various operations in IBPRE schemes into three categories according to an approximate and empirical evaluation criterion.

1) T_1: Hash operations, including hash operations $H_1(\cdot)$ and $H_2(\cdot)$.

2) T_2: Group operations, including operations in both $\mathbb{G}$ and $\mathbb{G}_T$ as well as FHE operations.

3) T_3: Bilinear pairing operations.

Then the detailed comparisons of computation complexity between our scheme and the most relevant scheme [14] are described in Table 1.

The main reason why we do not take into consideration the efficiency of **Setup** and **KeyGen** is that these two algorithms are run by the trust authority while in reality we care more about the efficiency of the users as well as the proxy. Besides, they are not the core factors for a cryptosystem since the number of running time of them are far less than that of the left four algorithms. For the fourth row of Table 1, $\{0\ (3T_2)\}$ means that in our scheme, the

Table 1: Comparison of computation complexity

	Our scheme	Scheme of [14]
Enc	$T_1 + 3T_2 + T_3$	$T_1 + 3T_2 + T_3$
RKGen	$T_1 + 8T_2 + T_3$	$2T_1 + 5T_2 + T_3$
ReEnc	$0\ (3T_2)$	$T_2 + T_3$
Dec$^{(1)}$	$T_2 + T_3$	$T_2 + T_3$
Dec$^{(2)}$	$5T_2 + 2T_3$	$2T_2 + 2T_3$

user does not need to do any computation and therefore $3T_2$ is merely for the proxy while $\{T_2+T_3\}$ is for the user in their scheme. For the fifth and sixth row, **Dec**$^{(1)}$ means the normal IBE decryption and **Dec**$^{(2)}$ means the decryption of a re-encrypted ciphertext. From Table 1 we can see that our scheme only introduces bearable additional computation workload to some of the algorithms like **Dec**$^{(2)}$ while ours performs better in **ReEnc**. Besides, the special property like universality can only be achieved by our scheme. So in general, ours is of significant usefulness in both theoretical research on re-encryption and some real-life applications.

5 Conclusion and Future Work

In this work, we have introduced a new type of identity-based proxy re-encryption scheme, which is different from as well as superior over the former schemes in some aspects. Specifically, Our scheme enjoys a good feature that there is only very little resource needed in the user side. Besides, potential bottlenecks like re-encryption management are also avoided. However, our scheme also has some limitations, *e.g.*, the introduction of homomorphic encryption decreases the efficiency of the scheme and the scheme is not suitable for multi-hop re-encryptions. We will continue working on enhancing our scheme and constructing more powerful ones.

Acknowledgement

The work described in this paper is supported by the Opening Project of Sichuan Province University Key Laboratory of Bridge Non-destruction Detecting and Engineering Computing (No. 2015QYJ04); Science Founding of Artificial Intelligence Key Laboratory of Sichuan Province (No. 2014RYJ06); Sichuan Provincial Key Laboratory of University Enterprise Information and Network Measurement and Control Technology (No. 2014WYJ03); The Scientific Research Fund Project of Sichuan University of Science & Engineering (No. 2013KY02).

References

[1] M. Blaze, G. Bleumer and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology (Eurocrypt'98)*, pp. 127–144, Springer, 1998.

[2] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[3] D. Boneh, A. Sahai and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography*, pp. 253–273, Springer, 2011.

[4] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[5] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 185–194, 2007.

[6] N. Chandran, M. Chase, F. H. Liu, R. Nishimaki and K. Xagawa, "Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices," in *Public-Key Cryptography (Pkr'14)*, pp. 95–112, Springer, 2014.

[7] N. Chandran, M. Chase and V. Vaikuntanathan, "Functional re-encryption and collusion-resistant obfuscation," in *Theory of Cryptography*, pp. 404–421, Springer, 2012.

[8] K. M. Chung, Y. Kalai and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Advances in Cryptology (Crypto'10)*, pp. 483–501, Springer, 2010.

[9] P. S. Chung, C. W. Liu and M. S. Hwang, "A study of attribute-based proxy re-encryption scheme in cloud environments," *International Journal of Network Security*, vol. 16, no. 1, pp. 1–13, 2014.

[10] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 40–49, 2013.

[11] S. Garg, C. Gentry, S. Halevi, A. Sahai and B. Waters, "Attribute-based encryption for circuits from multilinear maps," in *Advances in Cryptology (Crypto'13)*, pp. 479–499, Springer, 2013.

[12] R. Gennaro, C. Gentry and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Advances in Cryptology (Crypto'10)*, pp. 465–482, Springer, 2010.

[13] C. Gentry et al., "Fully homomorphic encryption using ideal lattices," in *Annual Symposium on the Theory of Computing*, vol. 9, pp. 169–178, 2009.

[14] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Applied Cryptography and Network Security*, pp. 288–306, Springer, 2007.

[15] S. Hada, "Secure obfuscation for encrypted signatures," in *Advances in Cryptology (Eurocrypt'10)*, pp. 92–112, Springer, 2010.

[16] G. Hanaoka, Y. Kawai, N. Kunihiro, T. Matsuda, J. Weng, R. Zhang and Y. Zhao. "Generic

construction of chosen ciphertext secure proxy re-encryption," in *Topics in Cryptology (Ct-rsa'12)*, pp. 349–364, Springer, 2012.

[17] S. Hohenberger, G. N. Rothblum, V. Vaikuntanathan, et al., "Securely obfuscating re-encryption," in *Theory of Cryptography*, pp. 233–252. Springer, 2007.

[18] M. S. Hwang, C. C. Lee and S. F. Tzeng, "A new proxy signature scheme for a specified group of verifiers," *Information Sciences*, vol. 227, pp. 102–115, 2013.

[19] A. A. Ivan and Y. Dodis, "Proxy cryptography revisited," in *Network and Distributed System Security Symposium*, 2003.

[20] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Public Key Cryptography (Pkc'08)*, pp. 360–379, Springer, 2008.

[21] M. Mambo and E. Okamoto, "Proxy cryptosystems: Delegation of the power to decrypt ciphertexts," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 80, no. 1, pp. 54–63, 1997.

[22] T. Matsuo, "Proxy re-encryption systems for identity-based encryption," in *Pairing-Based Cryptography (Pairing'07)*, pp. 247–267, Springer, 2007.

[23] M. Naehrig, K. Lauter and V. Vaikuntanathan, "Can homomorphic encryption be practical?," in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, pp. 113–124, 2011.

[24] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology*, pp. 47–53, Springer, 1985.

[25] F. Wang, C. C. Chang, C. Lin and S. C. Chang, "Secure and efficient identity-based proxy multi-signature using cubic residues," *International Journal of Network Security*, vol. 18, no. 1, pp. 90–98, 2016.

[26] J. Ye, H. Zhang and C. Fu, "Verifiable delegation of polynomials," *International Journal of Network Security*, vol. 18, no. 2, pp. 283–290, 2016.

Liang Liu received his B.S. degree in Computer Science and M.S. degree in Cryptography, both at School of Computer Science, Shaanxi Normal University. He is now a Ph.D. candidate in Cryptography at Xidian University. His research interests include theoretical cryptography, provable security and functional encryption.

Jun Ye received his B.S. degree in Applied Mathematics at Chongqing University and M.S. degree in Cryptography at Guilin University of Electronic Technology. He is now a Ph.D. candidate in Cryptography and Cloud Computing Security at Xidian University. His research interests include cryptography and information security.