An extended abstract of this paper appears in PKC 2009, S. Jarecki, G. Tsudik (Eds.), volume 5443 of LNCS, pp. 357-376, Sringer-Verlag, 2009.

CCA-Secure Proxy Re-Encryption without Pairings*

Jun Shao[1,2] chn.junshao@gmail.com
Zhenfu Cao[1†] zfcao@cs.sjtu.edu.cn

[1]Department of Computer Science and Engineering
Shanghai Jiao Tong University
[2]College of Information Sciences and Technology
Pennsylvania State University

Abstract

In a proxy re-encryption scheme, a semi-trusted proxy can transform a ciphertext under Alice's public key into another ciphertext that Bob can decrypt. However, the proxy cannot access the plaintext. Due to its transformation property, proxy re-encryption can be used in many applications, such as encrypted email forwarding. In this paper, by using signature of knowledge and Fijisaki-Okamoto conversion, we propose a proxy re-encryption scheme *without* pairings, in which the proxy can only transform the ciphertext in one direction. The proposal is secure against chosen ciphertext attack (CCA) and collusion attack in the *random oracle model* based on Decisional Diffie-Hellman (DDH) assumption over $\mathbb{Z}_{N^2}^*$ and integer factorization assumption, respectively. To the best of our knowledge, it is the *first* unidirectional PRE scheme with CCA security and collusion-resistance.

Keywords: Unidirectional PRE, DDH, random oracle, CCA security, collusion-resistance

1 Introduction

In 1998, Blaze, Bleumer, and Strauss [6] proposed the concept of *proxy re-encryption* (PRE), where a semi-trusted proxy can transform a ciphertext for Alice into another ciphertext that Bob can decrypt.[1] However, the proxy cannot get the plaintext. According to the direction of transformation, PRE schemes can be classified into two types, one is *bidirectional*, i.e., the proxy can transform from Alice to Bob and vice versa; the other is *unidirectional*, i.e., the proxy can only convert in one direction. Blaze *et al.* [6] also gave another method to classify PRE schemes: *multi-use*, i.e., the ciphertext can be transformed from Alice to Bob to Charlie and so on; and *single-use*, i.e., the ciphertext can be transformed only once.

Due to its transformation property, PRE can be used in many applications, including simplification of key distribution [6], key escrow [21], distributed file systems [2, 3], security in publish/subscribe systems [23], multicast [10], secure certified email mailing lists [24, 22], the DRM of Apple's iTunes [36], interoperable architecture of DRM [34], access control [35], and privacy for public transportation [19]. Recently,

*Supported by Research Fund for the Doctoral Program of Higher Education No. 20060248008, National Natural Science Foundation of China No. 60673079, Special Foundation of Huawei No. YZCB2006001, and National 973 Program No. 2007CB311201.

†Corresponding Author.

[1]In almost all related papers, the concept of PRE is introduced as "PRE allows a semi-trusted proxy to convert a ciphertext under Alice's public key to another ciphertext under Bob's public key". However, all existing unidirectional PRE schemes (including ours) do not exactly follow the definition. In particular, in these unidirectional PRE schemes, there are two kinds of ciphertexts, one is the original ciphertext, and the other is the transformed ciphertext. The transformed ciphertext is not exactly as the ciphertext under Bob's public key, but Bob can decrypt the transformed ciphertext only by his secret key. To the best of our knowledge, only the bidirectional schemes in [6, 9] satisfy the definition.

Hohenberger *et al.* got a result of securely obfuscating re-encryption [20], which is the first positive result for obfuscating an encryption functionality and against a series of impossibility results [18, 16, 4].

Since the introduction of PRE by Blaze, Bleumer, and Strauss [6], there have been many papers [6, 21, 2, 3, 17, 9, 11, 25] that have proposed different PRE schemes with different security properties. Some of them are related to chosen ciphertext attack (CCA) security. Ivan and Dodis [21] proposed a CCA security model for PRE and a generic construction of single-use PRE in the security model. Nevertheless, their security model allows the delegatee (Bob) to make use of the proxy as an oracle. As a result, the schemes only secure in their security model are not enough for some applications. For example, in encrypted email forwarding, an adversary (Bob) might hope to gain access to the original encrypted email by re-forming it, sending it to the proxy, and then hoping that the proxy responds with, "Can you forward the following to me again? [Encrypted attachment.]"

To fix the problem, Green and Ateniese [17], Canetti and Hohenberger [9] proposed new CCA security models for ID-based PRE and PRE, respectively. In these two new security models, it requires that the proxy checks the validity of the ciphertext before transformation, which is called public verifiability. Following this intuition, the first CCA secure, single-use, unidirectional ID-based PRE scheme in the *random oracle model* and the first CCA secure, multi-use, bidirectional PRE scheme in the *standard model* are proposed in [17, 9], respectively. However, the scheme in [17] suffers from the attack in Remark 2. Furthermore, the generic construction of PRE in [21] cannot be proved secure in the CCA security model in [9]. (See Appendix A for details. Hereafter, we refer CCA security to the definition in [9] or Section 2 of this paper.) Chu and Tzeng [11] proposed a multi-use, unidirectional ID-based PRE scheme, and claimed that it was CCA secure in the standard model. However, we showed that it was not true [31], since its transformed ciphertext $(C_{v1}, R, d'_1, d_2, d'_2)$ can be modified to another well-formed transformed ciphertext $(C_{v1}, R, d'_1 F_2(vk)^r, d_2, d'_2 g^r)$ by anyone, where r is a random number from $\mathbb{Z}_p^*$. Recently, Libert and Vergnaud [25] proposed a new unidirectional PRE scheme, which is replayable chosen ciphertext attack (RCCA) secure but *not* CCA-secure. It is fair to say that there is no CCA-secure unidirectional PRE scheme.[2] Furthermore, according to the results in [5, 29], the timing of a pairing computation is more than twice of that of a modular exponentiation computation. Hence, the CCA-secure unidirectional PRE schemes without pairings are desired.

Another important security notion on unidirectional PRE is collusion-resistance, which disallows Bob and the proxy to collude to reveal Alice's (long term) secret key, but allows the recovery of Alice's "weak" secret key only. In this case, Alice can delegate decryption rights, while keeping signing rights for the same public key. Till now, there are only a few PRE schemes [2, 3, 25] holding this security.[3]

Though many PRE schemes have been proposed, we find that no unidirectional PRE scheme without pairings but satisfying CCA security and collusion-resistance simultaneously, even in the random oracle model. In this paper, we attempt to propose such a unidirectional PRE scheme.

1.1 Our Contribution

We present a proxy re-encryption scheme *without* pairings, named scheme $\mathfrak{U}$, which is unidirectional and single-use, and proven CCA-secure and collusion resistant in the *random oracle model* based on Decisional Diffie-Hellman (DDH) assumption over $\mathbb{Z}_{N^2}^*$ and integer factorization assumption, respectively. Here, N is a safe-prime modulus.

The difficulty in constructing a CCA secure PRE scheme is to add the *public verifiability* to original ciphertexts. This public verifiability can prevent malicious Bob from gaining some advantage by using the proxy as an oracle. In pairing setting, such as [9], we can use the gap Diffie-Hellman problem (decisional Diffie-Hellman problem is easy, but computational Diffie-Hellman problem is hard) to achieve this. In

[2] When we prepared the camera-ready version, we found another paper [13] dealing the similar problems, and getting the similar results with us. In [13], the authors use Schnorr signature [28] to make the original ciphertext be publicly verifiable, while we use signature of knowledge [8, 1]. In our submission version, we have a CCA-secure bidirectional PRE scheme, however, the bidirectional one in [13] beats ours in every aspect. Hence, in the current version, we removed our bidirectional one, which can be found in [30]. Furthermore, the unidirectional scheme in [13] suffers from the attack in Remark 2.

[3] The unidirectional PRE scheme in [13] suffers from the collusion attack.

particular, the gap Diffie-Hellman problem allows us to check whether $\log_g A = \log_h B$. In this paper, we use *signature of knowledge* [8, 1] to provide $\log_g A = \log_h B$, hence obtaining public verifiability for original ciphertexts. In fact, using the signature of knowledge to provide public verifiability is due to Shoup and Gennaro [33]. Furthermore, we use Fujisaki-Okamoto conversion [14, 15] to provide the validity check of both original ciphertexts and re-encrypted ciphertexts for the decryptor (Alice or Bob).

Following the construction of the public key encryption scheme with double trapdoors in [7], scheme $\mathfrak{U}$ holds collusion-resistance. In particular, the factors of N are the long term secret key, and an exponent is the "weak" secret key, and revealing the exponent does not hurt the secrecy of the factors of N. To the best of our knowledge, scheme $\mathfrak{U}$ is the *first* unidirectional PRE scheme holding CCA security and collusion-resistance simultaneously.

Finally, we extend scheme $\mathfrak{U}$ to scheme $\mathfrak{U}_T$, where the delegator can revoke the proxy's transformation ability. In particular, the proxy can only transform the ciphertext during a restricted time interval.

1.2 Organization

The remaining paper is organized as follows. In Section 2, we review the definitions related to our proposals. In what follows, we present scheme $\mathfrak{U}$ and its security analysis, and scheme $\mathfrak{U}_T$ and its security analysis, in Section 3 and Section 4, respectively. In Section 5 we compare scheme $\mathfrak{U}$ with previous unidirectional PRE schemes. Finally, we conclude the paper in Section 6.

2 Preliminaries

In this section, we briefly review the definitions related to our proposals, some similar content can be found in [8, 1, 17, 9].

2.1 Public Key Encryption

Definition 1 (Public Key Encryption (PKE)) *A public key encryption scheme* PKE *is a triple of PPT algorithms* (KeyGen, Enc, Dec)*:*

- *KeyGen*$(1^k) \to (pk, sk)$. *On input the security parameter* 1^k*, the key generation algorithm KeyGen outputs a public key* pk *and a secret key* sk*.*
- *Enc*$(pk, m) \to C$. *On input a public key* pk *and a message* m *in the message space, the encryption algorithm Enc outputs a ciphertext* C*.*
- *Dec*$(sk, C) \to m$. *On input a secret key* sk *and a ciphertext* C*, the decryption algorithm Dec outputs a message* m *in the message space or* $\perp$*.*

2.1.1 Correctness.

The correctness property is that for any message m in the message space and any key pair $(pk, sk) \leftarrow \mathtt{KeyGen}(1^k)$. Then the following condition must hold: $\mathtt{Dec}(sk, \mathtt{Enc}(pk, m)) = m$.

2.2 Unidirectional Proxy Re-Encryption

Definition 2 (Unidirectional PRE) *A unidirectional proxy re-encryption scheme* UniPRE *is a tuple of PPT algorithms* (KeyGen, ReKeyGen, Enc, ReEnc, Dec)*:*

- *KeyGen, Enc, Dec: Identical to those in public key encryption.*
- *ReKeyGen*$(sk_1, pk_2) \to rk_{1\to 2}$. *On input a secret key* sk_1 *and a public key* pk_2*, the re-encryption key generation algorithm ReKeyGen outputs a unidirectional re-encryption key* $rk_{1\to 2}$*.*

- $\texttt{ReEnc}(rk_{1\to 2}, C_1) \to C_2$. *On input a re-encryption key* $rk_{1\to 2}$ *and a ciphertext* C_1*, the re-encryption algorithm* ReEnc *outputs a re-encrypted ciphertext* C_2 *or* $\perp$.

2.2.1 Correctness.

A correct proxy re-encryption scheme should satisfy two requirements:

$$\texttt{Dec}(sk, \texttt{Enc}(pk, m)) = m,$$

and

$$\texttt{Dec}(sk', \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk'), C)) = m,$$

where $(pk, sk), (pk', sk') \leftarrow \texttt{KeyGen}(1^k)$, and C is the ciphertext of message m for pk from algorithm Enc or algorithm ReEnc.

2.2.2 Chosen Ciphertext Security for Unidirectional Proxy Re-Encryption.

This security note is a modification of replayable chosen ciphertext security in [25], where the corrupted public keys are *not* decided before start of the Uni-PRE-CCA game, and the adversary is *allowed* adaptive corruption of users[4], and proxies between corrupted and uncorrupted users. But unlike [25], we require that one well-formed ciphertext *cannot* be modified (but can be transformed) to be another well-formed ciphertext. In [25], anyone can modify the transformed ciphertext, such that $(C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma) \to (C_1, {C_2'}^t, {C_2''}^{t^{-1}}, {C_2'''}^t, C_3, C_4, \sigma)$, where t is a random number from $\mathbb{Z}_p$.

Note that this security model is only for single-use scheme.

Phase 1: The adversary $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

- *Public key generation oracle* $\mathcal{O}_{pk}$: On input an index i,[5] the Challenger takes a security parameter k, and responds by running algorithm $\texttt{KeyGen}(1^k)$ to generate a key pair (pk_i, sk_i), gives pk_i to $\mathcal{A}$ and records (pk_i, sk_i) in table T_K.
- *Secret key generation oracle* $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, where pk is from $\mathcal{O}_{pk}$, the Challenger searches pk in table T_K and returns sk.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}$: On input (pk, pk') by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, the Challenger returns the re-encryption key $rk_{pk\to pk'} = \texttt{ReKeyGen}(sk, pk')$, where sk is the secret key corresponding to pk.
- *Re-encryption oracle* $\mathcal{O}_{re}$: On input (pk, pk', C) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, the re-encrypted ciphertext $C' = \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk'), C)$ is returned by the Challenger, where sk is the secret key corresponding to pk.
- *Decryption oracle* $\mathcal{O}_{dec}$: On input (pk, C), where pk is from $\mathcal{O}_{pk}$, the Challenger returns $\texttt{Dec}(sk, C)$, where sk is the secret key corresponding to pk.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, and a public key pk^* on which it wishes to be challenged. There are three constraints on the public key pk^*, (i) it is from $\mathcal{O}_{pk}$; (ii) it did not appear in any query to $\mathcal{O}_{sk}$ in Phase 1; (iii) if $(pk^*, \bigstar)$ did appear in any query to $\mathcal{O}_{rk}$, then $\bigstar$ did not appear in any query to $\mathcal{O}_{sk}$. The Challenger picks a random bit $b \in \{0, 1\}$ and sets $C^* = \texttt{Enc}(pk^*, m_b)$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ issues more queries $q_{n_1+1}, \cdots, q_n$ where query q_i is one of:

[4]The security model in [13] does not allow such adaptive corruption.

[5]This index is just used to distinguish different public keys.

- $\mathcal{O}_{pk}$: The Challenger responds as in Phase 1.
- $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk is from $\mathcal{O}_{pk}$;
  - $pk \neq pk^*$;
  - (pk^*, pk) is not a query to $\mathcal{O}_{rk}$ before;
  - (pk', pk, C') is not a query to $\mathcal{O}_{re}$ before, where (pk', C') is a `derivative`[6] of (pk^*, C^*).
- $\mathcal{O}_{rk}$: On input (pk, pk') by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if $pk = pk^*$, then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{re}$: On input (pk, pk', C) by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if (pk, C) is a derivative of (pk^*, C^*), then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{dec}$: On input (pk, C), if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk is from $\mathcal{O}_{pk}$;
  - (pk, C) is not a derivative of (pk^*, C^*).

These queries may be also asked adaptively.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as a Uni-PRE-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking `UniPRE` as the following function of the security parameter k: $\mathrm{Adv}_{\mathtt{UniPRE},\mathcal{A}}(k) = |\Pr[b = b'] - 1/2|$. Using the Uni-PRE-CCA game we can define chosen ciphertext security for unidirectional proxy re-encryption schemes.

Definition 3 (Uni-PRE-CCA security) *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time Uni-PRE-CCA adversary $\mathcal{A}$ the function $Adv_{\mathtt{UniPRE},\mathcal{A}}(k)$ is negligible. As shorthand, we say that* `UniPRE` *is Uni-PRE-CCA secure.*

Remark 1 *In [25], the authors considered this model as a static corruption model, since it does not capture some scenarios, such as the adversary generate public keys on behalf of corrupted parties. However, we think this model is an* adaptive *corruption model. Since* Adaptive Security *usually refers to the ability of the adversary to choose which parties to corrupt depending on the information gathered so far,* but *the Challenger still generates all parties' key pairs. Allowing adversaries to generate malicious parties' public keys on their own is usually called "chosen-key model" [26].*[7]

[6] Derivatives of (pk^*, C^*) are defined as follows [9]:

1. (pk^*, C^*) is a derivative of itself.
2. If (pk, C) is a derivative of (pk^*, C^*) and (pk', C') is a derivative of (pk, C), then (pk', C') is a derivative of (pk^*, C^*).
3. If $\mathcal{A}$ has queried $\mathcal{O}_{re}$ on input (pk, pk', C) and obtained (pk', C'), then (pk', C') is a derivative of (pk, C).
4. If $\mathcal{A}$ has queried $\mathcal{O}_{rk}$ on input (pk, pk'), and $C' = \mathtt{ReEnc}(\mathcal{O}_{re}(pk, pk'), C)$, then (pk', C') is a derivative of (pk, C).

[7] We thank an anonymous reviewer of Indocrypt 2008 to point out this.

Besides CCA security, there is another security notion, collusion resistance, for unidirectional PRE schemes.

Definition 4 (Uni-PRE-CR security) [8] *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *is collusion resistant if for any polynomial bounded adversary $\mathcal{A}$, the following probability is negligible:*

$$\begin{aligned} Pr[(sk_1, pk_1) \leftarrow \texttt{KeyGen}(1^k), \{(sk_i, pk_i) \leftarrow \texttt{KeyGen}(1^k)\}, \\ \{rk_{i\rightarrow 1} \leftarrow \texttt{ReKeyGen}(sk_i, pk_1)\}, \\ \{rk_{1\rightarrow i} \leftarrow \texttt{ReKeyGen}(sk_1, pk_i)\}, \\ i = 2, \cdots, \\ \alpha \leftarrow \mathcal{A}(pk_1, \{pk_i, sk_i\}, \{rk_{1\rightarrow i}\}, \{rk_{i\rightarrow 1}\}) : \\ \alpha = sk_1]. \end{aligned}$$

Due to its similarity with that of unidirectional PRE schemes, we put the definitions of unidirectional PRE schemes with temporary delegation in the Appendix.

2.3 Signature of Knowledge

In our proposal, we apply the following non-interactive zero-knowledge proof of knowledge, named signature of knowledge of equality of two discrete logarithms [8, 1, 32].

Definition 5 *Let $y_1, y_2, g, h \in \mathbb{G}$, $\mathbb{G}$ be a cyclic group of quadratic residues modulo N^2 (N is a safe-prime modulus), and $H(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^k$ (k is the security parameter). A pair (c, s), verifying $c = H(y_1||y_2||g||h||g^s y_1^c||h^s y_2^c||m)$ is a signature of knowledge of the discrete logarithm of both $y_1 = g^x$ w.r.t. base g and $y_2 = h^x$ w.r.t. base h, on a message $m \in \{0,1\}^*$.*

The party in possession of the secret x is able to compute the signature, provided that $x = \log_g y_1 = \log_h y_2$, by choosing a random $t \in \{0, \cdots, 2^{|N^2|+k} - 1\}$ ($|n|$ is the bit-length of n). And then computing c and s as:

$$c = H(y_1||y_2||g||h||g^t||h^t||m) \text{ and } s = t - cx.$$

We denote $\texttt{SoK.Gen}(y_1, y_2, g, h, m)$ as the generation of the proof.

2.4 Complexity Assumption

The security of our proposal is based on the Decisional Diffie-Hellman assumption (DDH) over $\mathbb{Z}_{N^2}^*$.
DDH Problem. The DDH problem is as follows: Given $\langle g, g^a, g^b \rangle$ for some $a, b \in ord(\mathbb{G})$ and $T \in \mathbb{G}$, decide whether $T = g^{ab}$, where $\mathbb{G}$ is a cyclic group of quadratic residues modulo N^2 (N is a safe-prime modulus), g is a random number of $\mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ε in solving DDH problem if $|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, T) = 0]| \geq \varepsilon$, where the probability is over the random choices of a, b in $ord(\mathbb{G})$, the random choices of g, T in $\mathbb{G}$, and the random bits of $\mathcal{A}$.

Definition 6 (DDH Assumption) *We say that the ε-DDH assumption holds if no PPT algorithm has advantage at least ε in solving the DDH problem.*

Note that the DDH problem over $\mathbb{Z}_{N^2}^*$ is easy if the factors of N is known [7].

[8] This security notion is from [2, 3], called Master Secret Security.

2.5 The Public Key Encryption with Double Trapdoors

The basic public key encryption of our proposal is the public key encryption with double trapdoors in [7], named BCP03.

The following description is from [7]. Let $N = pq$ be a safe prime modulus, such that $p = 2p' + 1$, $q = 2q' + 1$, and p, p', q, q' are primes. Assume $\mathbb{G}$ is the cyclic group of quadratic residues modulo N^2, then we have the order of $\mathbb{G}$ is $Np'q'$.

- $\texttt{KeyGen}(1^k) \rightarrow (pk, sk)$. Choose a random element $\alpha \in \mathbb{Z}^*_{N^2}$, a random value $a \in [1, Np'q']$, and set $g = \alpha^2 \mod N^2$ and $h = g^a \mod N^2$. The public key is (N, g, h), and the secret key is a.
- $\texttt{Enc}(pk, m) \rightarrow C$. On input a public key pk and a message $m \in \mathbb{Z}_N$, the ciphertext (A, B) is computed as
$$A = g^r \bmod N^2, \quad B = h^r(1 + mN) \bmod N^2,$$
where r is a random number from $\mathbb{Z}_{N^2}$.
- $\texttt{Dec}(sk, C) \rightarrow m$. There are two methods to decrypt.
  - Knowing a, one can compute m by
$$m = \frac{B/(A^a) - 1 \bmod N^2}{N}.$$
  - Knowing p', q', one can compute m by
$$m = \frac{D - 1 \bmod N^2}{N} \cdot \pi \bmod N,$$
where $D = \left(\frac{B}{g^{w_1}}\right)^{2p'q'}$, $w_1 = ar \bmod N$, $ar \bmod pqp'q' = w_1 + w_2 N$, π is the inverse of $2p'q' \bmod N$.
Note the values of $a \bmod N$ and $r \bmod N$ can be computed when given $h = g^a \bmod N^2$, $A = g^r \bmod N^2$, and p', q', by the method in [27] (Theorem 1 in [27]).

3 New Unidirectional Proxy Re-Encryption Scheme without Pairings

The proposed unidirectional scheme $\mathfrak{U}$ is based on the CPA secure and collusion resistant unidirectional PRE scheme in [2, 3] (the first attempt scheme in [2, 3]), and with the signature of knowledge [8, 1] and Fujisaki-Okamoto conversion [14, 15]. The basic public key encryption is scheme BCP03.

The intuition in scheme $\mathfrak{U}$ is as follows. Firstly, since there are two trapdoors (a and the factorization of the modulus) in scheme BCP03, we can use the key sharing technique in [17] to share a. In particular, let $a = r_1 + r_2$, and sent the proxy r_1 and the ciphertext of r_2 under the delegatee's public key. Knowing a cannot hurt the secrecy of the factorization of the modulus, hence, collusion-resistance obtained. Secondly, scheme BCP03 is CPA-secure, hence, we use Fijisaki-Okamoto conversion to make scheme BCP03 be CCA-secure. Thirdly, we use the signature of knowledge to make the original ciphertext be publicly verifiable.

3.1 Scheme $\mathfrak{U}$ with Single-Use

Scheme $\mathfrak{U}$ contains three cryptographic hash functions for all users: $H_1(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$, $H_2(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^n$, and $H_3(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^{k_2}$, where k_1 and k_2 are the security parameter, n is the bit-length of messages to be encrypted. The details are as follows.

`KeyGen`: Choose a safe-prime modulus $N = pq$, three random numbers $\alpha \in \mathbb{Z}^*_{N^2}$, $a, b \in [1, pp'qq']$, a hash function $H(\cdot)$, where $p = 2p' + 1$, $q = 2q' + 1$, p, p', q, q' are primes, and $H(\cdot) : \{0,1\}^* \to \mathbb{Z}_{N^2}$. Furthermore, set $g_0 = \alpha^2 \bmod N^2$, $g_1 = {g_0}^a \bmod N^2$, and $g_2 = {g_0}^b \bmod N^2$. The public key is $pk = (H(\cdot), N, g_0, g_1, g_2)$, the "weak" secret key is $wsk = (a, b)$, and the long term secret key is $sk = (p, q, p', q')$.

`ReKeyGen`: On input a public key $pk_Y = (H_Y(\cdot), N_Y, g_{Y0}, g_{Y1}, g_{Y2})$, a "weak" secret key $wsk_X = a_X$, and a long term secret key $sk_X = (p_X, q_X, p'_X, q'_X)$, it outputs the unidirectional re-encryption key $rk_{X\to Y} = (rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$, where $rk^{(1)}_{X\to Y} = (\dot{A}, \dot{B}, \dot{C})$, and computed as follows:

- Choose two random numbers $\dot{\sigma} \in \mathbb{Z}_N$, $\dot{\beta} \in \{0,1\}^{k_1}$.
- Compute $rk^{(2)}_{X\to Y} = a_X - \dot{\beta} \bmod (p_X q_X p'_X q'_X)$.
- Compute $r_{X\to Y} = H_Y(\dot{\sigma}||\dot{\beta})$, $\dot{A} = (g_{Y0})^{r_{X\to Y}} \bmod (N_Y)^2$, $\dot{C} = H_1(\dot{\sigma}) \oplus \dot{\beta}$,

$$\dot{B} = (g_{Y2})^{r_{X\to Y}} \cdot (1 + \dot{\sigma} N_Y) \bmod (N_Y)^2. \tag{1}$$

`Enc`: On input a public key $pk = (H(\cdot), N, g_0, g_1, g_2)$ and a message $m \in \{0,1\}^n$, the encryptor does the following performances:

- Choose a random number $\sigma \in \mathbb{Z}_N$.
- Compute $r = H(\sigma||m)$, $A = (g_0)^r \bmod N^2$, $C = H_2(\sigma) \oplus m$, $D = (g_2)^r \bmod N^2$,

$$B = (g_1)^r \cdot (1 + \sigma N) \bmod N^2. \tag{2}$$

- Run $(c, s) \leftarrow \texttt{SoK.Gen}(A, D, g_0, g_2, (B, C))$, where the underlying hash function is H_3.
- Output the ciphertext $K = (A, B, C, D, c, s)$.

`ReEnc`: On input a re-encryption key $rk_{X\to Y} = (rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ and a ciphertext $K = (A, B, C, D, c, s)$ under key $pk_X = (H_X(\cdot), N_X, g_{X0}, g_{X1}, g_{X2})$, check whether $c = H_3(A||D||g_{X0}||g_{X2}||(g_{X0})^s A^c||(g_{X2})^s D^c||(B||C))$. If not hold, output $\perp$ and terminate; otherwise, re-encrypt the ciphertext to be under key pk_Y as:

- Compute $A' = A^{rk^{(2)}_{X\to Y}} = (g_{X0})^{r(a_X - \dot{\beta})} \bmod (N_X)^2$.
- Output the new ciphertext $(A, A', B, C, rk^{(1)}_{X\to Y}) = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$.

`Dec`: On input a secret key and any ciphertext K, parse $K = (A, B, C, D, c, s)$, or $K = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$.

Case $K = (A, B, C, D, c, s)$**:** Check whether $c = H_3(A||D||g_0||g_2||(g_0)^s A^c||(g_2)^s D^c||(B||C))$, if not, output $\perp$ and terminate; otherwise,

- if the input secret key is the "weak" secret key a, compute $\sigma = \frac{B/(A^a) - 1 \bmod N^2}{N}$.
- if the secret key is the long term secret key (p, q, p', q'), compute $\sigma = \frac{(B/g_0^{w_1})^{2p'q'} - 1 \bmod N^2}{N} \cdot \pi (\bmod N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $m = C \oplus H_2(\sigma)$, if $B = (g_1)^{H(\sigma||m)} \cdot (1 + \sigma N) \bmod N^2$ holds, output m; otherwise, output $\perp$ and terminate.

Case $K = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$**:** In this case, the decryptor should know the delegator's (Alice's) public key $(H'(\cdot), N', g'_0, g'_1, g'_2)$.

- If the input secret key is the "weak" secret key b, compute $\dot{\sigma} = \frac{\dot{B}/(\dot{A}^b) - 1 \bmod N^2}{N}$.

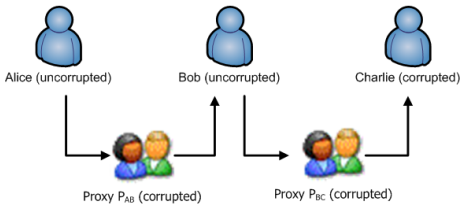

Figure 1: An example of delegation relationship.

- If the input secret key is the long term secret key (p, q, p', q'), computes $\dot{\sigma} = \frac{(\dot{B}/g_0^{w_1})^{2p'q'} - 1 \bmod N^2}{N} \cdot \pi (\bmod N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $\dot{\beta} = \dot{C} \oplus H_1(\dot{\sigma})$, if $\dot{B} = (g_2)^{H(\dot{\sigma}||\dot{\beta})} \cdot (1 + \dot{\sigma}N) \bmod N^2$ holds, then compute $\sigma = \frac{B/(A' \cdot A^{\dot{\beta}}) - 1 \bmod N'^2}{N'}$, $m = C \oplus H_2(\sigma)$; otherwise, output $\perp$ and terminate. If $A = (g_0')^{H'(\sigma||m)} \bmod N'^2$ and $B = (g_1')^{H'(\sigma||m)} \cdot (1 + \sigma N') \bmod N'^2$ both hold, then output m; otherwise, output $\perp$ and terminate.

Note that $(H(\cdot), N, g_0, g_1, g_2)$ is the public key of the decryptor.

Remark 2 *The values of $\dot{B}$ and B are computed differently, in particular, in equation (1), the base is g_1, while in equation (2), the base is g_2. This difference aims to resist the following attack: Assume that there is the delegation relationship as in Fig. 1. Alice delegates her decryption rights to Bob via the proxy P_{AB}, and Bob delegates his decryption rights to Charlie via the proxy P_{BC}. Alice and Bob are uncorrupted, the rest parties are corrupted, and the target (challenged) user is Alice. This corruption situation is allowed in the security model in Section 2 (Note that the attacked scheme should be single-use). If the bases in equations (1) and (2) are both g_1, then the adversary can decrypt any ciphertext for Alice as follows. The proxy P_{BC} and Charlie colludes to get Bob's weak secret key a_B, and then they collude with the proxy P_{AB} to get Alice's weak secret key a_A. As a result, the adversary can use a_A to decrypt any ciphertext for Alice. However, in scheme $\mathfrak{U}$, the proxy P_{BC} and Charlie cannot get Bob's weak secret key b_B (which is for decrypting partial re-encryption key), hence, they cannot collude with the proxy P_{AB} to get Alice's weak secret key a_A (which is for decrypting ciphertexts).*

Note that the above attack is also allowed in the security model in [17, 13], since they only disallow the adversary to corrupt the proxy between the target user and the uncorrupted user. The unidirectional schemes in [17, 13] suffer from the above attack. To resist the above attack, we can use the same method in scheme $\mathfrak{U}$, in particular, every user has two public/secret key pairs, one is for decrypting ciphertexts of messages, and the other is for decrypting the partial re-encryption key.

Correctness. The correctness property is easily obtained by the correctness of scheme BCP03 [7] and Fujisaki-Okamoto conversion [14, 15].

Theorem 1 (Uni-PRE-CCA security) *In the random oracle model, scheme $\mathfrak{U}$ is CCA-secure under the assumptions that DDH problem over $\mathbb{Z}^*_{\mathbf{N}^2}$ is hard, and that the signature of knowledge is secure.*

Proof. We show that if there exists an algorithm $\mathcal{A}$ that can break $\mathfrak{U}$ with probability ϵ in time t, then there is another algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve DDH problem over $\mathbb{Z}^*_{N^2}$, i.e., on DDH input $(\mathbf{N}, \mathbf{g}, \mathbf{g^u}, \mathbf{g^v}, \mathbf{T})$, $\mathcal{B}$ decides if $\mathbf{T} = \mathbf{g^{uv}}$ or not.

$\mathcal{B}$ interacts with $\mathcal{A}$ in a Uni-PRE-CCA game as follows ($\mathcal{B}$ simulates the Challenger for $\mathcal{A}$). In the following, we use starred letters $(A^*, B^*, C^*, D^*, c^*, s^*)$ to refer to the challenge ciphertext corresponding to an uncorrupted pk^*.

Hash Oracles:

$\mathcal{O}_H$: One oracle $\mathcal{O}_H$ is corresponding to a hash function $H(\cdot)$ which is a part of user's public key. As a result, there are many such oracles, and they are all constructed in the following method. On input (σ_i, m_i), $\mathcal{B}$ first checks whether triple $(\sigma_i, m_i, \alpha_i)$ exists in table T_H. If yes, $\mathcal{B}$ responds $\mathcal{A}$ with α_i; otherwise $\mathcal{B}$ chooses a random number $\alpha_i \in \mathbb{Z}_{N^2}$, responds $\mathcal{A}$ with α_i, and records $(\sigma_i, m_i, \alpha_i)$ in table T_H, where N is the corresponding safe-prime modulus and a part of user's public key.

$\mathcal{O}_{H_1}$: On input σ_i, $\mathcal{B}$ first checks whether pair (σ_i, β_i) exists in table T_{H_1}. If yes, $\mathcal{B}$ responds $\mathcal{A}$ with β_i; otherwise, $\mathcal{B}$ chooses a random number $\beta_i \in \{0,1\}^{k_1}$, responds $\mathcal{A}$ with β_i, and records (σ_i, β_i) in table T_{H_1}.

$\mathcal{O}_{H_2}$: On input σ_i, $\mathcal{B}$ first checks whether pair (σ_i, γ_i) exists in table T_{H_2}. If yes, $\mathcal{B}$ responds $\mathcal{A}$ with γ_i; otherwise, $\mathcal{B}$ chooses a random number $\gamma_i \in \{0,1\}^{n}$, responds $\mathcal{A}$ with γ_i, and records (σ_i, γ_i) in table T_{H_2}.

$\mathcal{O}_{H_3}$: On input $(A_i, D_i, g_{i0}, g_{i2}, E_i, F_i, B_i, C_i)$, $\mathcal{B}$ first checks whether tuple $(A_i, D_i, g_{i0}, g_{i2}, E_i, F_i, B_i, C_i, \delta_i)$ exists in table T_{H_2}. If yes, $\mathcal{B}$ responds $\mathcal{A}$ with δ_i; otherwise, $\mathcal{B}$ chooses a random number $\delta_i \in \{0,1\}^{k_2}$, responds $\mathcal{A}$ with δ_i, and records $(A_i, D_i, g_{i0}, g_{i1}, E_i, F_i, B_i, C_i, \delta_i)$ in table T_{H_3}.

Phase 1:

$\mathcal{O}_{pk}$: On input an index i, $\mathcal{B}$ decides whether pk_i is the attacked public key pk^*.

- If yes, $\mathcal{B}$ sets $N = \mathbf{N}$, $H(\cdot) : \{0,1\}^* \to \mathbb{Z}_{\mathbf{N}^2}$, $g_0 = \mathbf{g}$, $g_1 = \mathbf{g}^{\mathbf{u}}$, and $g_2 = \mathbf{g}^w$, where $w \in \mathbb{Z}_{\mathbf{N}^2}$. And then, $\mathcal{B}$ records $(H(\cdot), N, g_0, g_1, g_2, \perp, \perp, \perp, \perp, coin_i)$ in table T_K.
- Otherwise, $\mathcal{B}$ runs `KeyGen` to get the public key $(H(\cdot), N, g_0, g_1, g_2)$, the long-term secret key (p', q'), and weak secret key (a, b), and records $(H(\cdot), N, g_0, g_1, g_2, a, b, p', q')$ in table T_K.

At last, $\mathcal{B}$ returns $(H(\cdot), N, g_0, g_1, g_2)$ to $\mathcal{A}$ as pk_i.

$\mathcal{O}_{sk}$: On input $pk_X = (H_X(\cdot), N_X, g_{X0}, g_{X1}, g_{X2})$, $\mathcal{B}$ checks whether pk_X exists in T_K. If not, $\mathcal{B}$ terminates. Otherwise, if pk_X is the guessed attacked public key, $\mathcal{B}$ reports *failure* and aborts; otherwise, $\mathcal{B}$ responds $\mathcal{A}$ with corresponding (p'_X, q'_X), and records pk_X into table T_{sk}.

$\mathcal{O}_{rk}$: On input (pk_X, pk_Y), $\mathcal{B}$ checks whether pk_X and pk_Y both exist in T_K. If not, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ checks whether $(pk_X, pk_Y, rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ is in table T_{rk}, or $(pk_X, pk_Y, \beta_{X\to Y}, rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ is in table T_{urk}, if it exists, $\mathcal{B}$ returns $(rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ to $\mathcal{A}$; otherwise,

- If pk_X is in table T_{sk} or pk_X is not the guessed attacked public key, $\mathcal{B}$ responds $\mathcal{A}$ with $(rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y}) \leftarrow$ `ReKeyGen`(sk_X, pk_Y), and records $(pk_X, pk_Y, rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ in table T_{rk}.
- If pk_X is the guessed attacked public key, and pk_Y is not in table T_{sk}, $\mathcal{B}$ chooses three random numbers $\dot{\beta} \in \{0,1\}^{k_1}$, $rk^{(2)}_{X\to Y} \in \mathbb{Z}_{N^2}$, $\dot{\sigma} \in \mathbb{Z}_N$, and does
  - Compute
$$r_{X\to Y} = H_Y(\dot{\sigma}||\dot{\beta}),$$
$$\dot{A} = (g_{Y0})^{r_{X\to Y}} \bmod (N_Y)^2,$$
$$\dot{B} = (g_{Y1})^{r_{X\to Y}} \cdot (1 + \dot{\sigma} N_Y) \bmod (N_Y)^2,$$
$$\dot{C} = H_1(\dot{\sigma}) \oplus \dot{\beta},$$
    where $(H_Y(\cdot), N_Y, g_{Y0}, g_{Y1}, g_{Y2}) = pk_Y$.
  - Set $rk^{(1)}_{X\to Y} = (\dot{A}, \dot{B}, \dot{C})$.
  - Return $(rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ to $\mathcal{A}$.
  - Record $(pk_X, pk_Y, \dot{\beta}, rk^{(1)}_{X\to Y}, rk^{(2)}_{X\to Y})$ in table T_{urk}.

Because of the security property of scheme BCP03 with Fujisaki-Okamoto conversion, we have that only if we can successfully respond the decryption oracle queried with the re-encrypted ciphertexts re-encrypted by the above re-encryption keys, this step is the undistinguishable from the real execution from the viewpoint of $\mathcal{A}$.

- If pk_X is the guessed attacked public key, and pk_Y is in table T_{sk}, $\mathcal{B}$ reports *failure* and aborts.

$\mathcal{O}_{re}$: On input (pk_X, pk_Y, K), $\mathcal{B}$ checks whether pk_X and pk_Y both exist in table T_K. If not, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ parses $K = (A, B, C, D, c, s)$, and checks whether $c = H_3(A||D||g_{X0}||g_{X2}||(g_{X0})^s A^c||(g_{X2})^s D^c||(B||C))$, where $pk_X = (H_X(\cdot), N_X, g_{X0}, g_{X1}, g_{X2})$, if not, $\mathcal{B}$ outputs $\perp$ and terminates; otherwise, do the following performances.

- If pk_X is the guessed attacked public key, and pk_Y is in table T_{sk}, $\mathcal{B}$ does:
  1. Set two empty lists, S_1 and S_2.
  2. Find all elements $(\sigma_i, m_i, \alpha_i)$ in table T_{H_X} such that $A = (g_{X0})^{\alpha_i} \bmod (N_X)^2$, and put them into list S_1. If $S_1 = \varnothing$, then output $\perp$ and terminate.
     This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_X(\sigma_i||m_i)$ without querying $\mathcal{O}_{H_X}$. The probability of this event is $q_{H_X}/|\mathbb{Z}_{(N_X)^2}|$, where q_{H_X} is the number of queries to $\mathcal{O}_{H_X}$.
  3. For every $(\sigma_i, m_i, \alpha_i)$ in list S_1, find all elements in table T_{H_2} such that $\sigma_j = \sigma_i$ and put them (i.e., $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \gamma_j)$'s) into list S_2. If $S_2 = \varnothing$, then output $\perp$ and terminate.
     This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_2(\sigma_i)$ without querying $\mathcal{O}_{H_2}$. The probability of this event is $q_{H_2}/2^n$, where q_{H_2} is the number of queries to $\mathcal{O}_{H_2}$.
  4. Check in S_2 if there exists a $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \gamma_j)$ such that $(g_{X1})^{\alpha_i} \cdot (1+\sigma_i N_X) \bmod (N_X)^2 = B$ and $\gamma_j \oplus m_i = C$. If it does not exist or more than one exist, then output $\perp$ and terminate.
  5. Search $(pk_X, pk_Y, \dot{\beta})$ in table T_{frk}, if not, choose a random number from $\{0,1\}^{k_1}$ for $\dot{\beta}$, and record $(pk_X, pk_Y, \dot{\beta})$ in table T_{frk}.
  6. Choose a random number $\dot{\sigma} \in \mathbb{Z}_N$.
  7. Compute
  $$r_{X\to Y} = H_Y(\dot{\sigma}||\dot{\beta}),$$
  $$\dot{A} = (g_{Y0})^{r_{X\to Y}},$$
  $$\dot{B} = (g_{Y1})^{r_{X\to Y}} \cdot (1 + \dot{\sigma} N_Y) \bmod (N_Y)^2,$$
  $$\dot{C} = H_1(\dot{\sigma}) \oplus \beta_{X\to Y},$$
  where $pk_Y = (H_Y(\cdot), N_Y, g_{Y0}, g_{Y1}, g_{Y2})$.
  8. Set $rk^{(1)}_{X\to Y} = (\dot{A}, \dot{B}, \dot{C})$.
  9. Return $(A, (g_{X1})^{\alpha_i} \cdot (g_{X0})^{-\dot{\beta}}, B, C, rk^{(1)}_{X\to Y})$ to $\mathcal{A}$.
- Otherwise, $\mathcal{B}$ calls the oracle $\mathcal{O}_{rk}$ to get the re-encryption key $rk_{X\to Y}$, and returns $\texttt{ReEnc}(rk_{X\to Y}, K)$.

$\mathcal{O}_{dec}$: On input (pk_X, K), $\mathcal{B}$ checks whether pk_X exists in table T_K, if not, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ does the following performances.

- If pk_X is not the guessed attacked public key, then sk_X is known to $\mathcal{B}$, who responds $\mathcal{A}$ with $\texttt{Dec}(sk_X, K)$.
- If pk_X is the guessed attacked public key and $K = (A, B, C, D, c, s)$, $\mathcal{B}$ checks whether $c = H_3(A||D||g_{X0}||g_{X2}||(g_{X0})^s A^c||(g_{X2})^s D^c||(B||C))$, if not, $\mathcal{B}$ outputs $\perp$ and terminates; otherwise, $\mathcal{B}$ does:
  1. Set two empty lists, S_1 and S_2.

2. Find all elements $(\sigma_i, m_i, \alpha_i)$ in table T_{H_X} such that $A = (g_{X0})^{\alpha_i} \bmod (N_X)^2$, and put them into list S_1. If $S_1 = \varnothing$, then output $\perp$ and terminate.
   This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_X(\sigma_i||m_i)$ without querying $\mathcal{O}_{H_X}$. The probability of this event is $q_{H_X}/|\mathbb{Z}_{(N_X)^2}|$.
3. For every $(\sigma_i, m_i, \alpha_i)$ in list S_1, find all elements in table T_{H_2} such that $\sigma_j = \sigma_i$ and put them (i.e., $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \gamma_j)$'s) into list S_2. If $S_2 = \varnothing$, then output $\perp$ and terminate.
   This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_2(\sigma_i)$ without querying $\mathcal{O}_{H_2}$. The probability of this event is $q_{H_2}/2^n$.
4. Check in S_2 if there exists a $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \gamma_j)$ such that $(g_{X1})^{\alpha_i \cdot (1+\sigma_i N_X)} \bmod (N_X)^2 = B$ and $\gamma_j \oplus m_i = C$. If none exists or more than one exist, then output $\perp$ and terminate; otherwise, output m_i.

- If pk_X is the guessed attacked public key and $K = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$, $\mathcal{B}$ searches $(pk_Y, pk_X, \bigstar_1, \bigstar_2, \bigstar_3)$ in table T_{urk}, such that $\bigstar_2 = (\dot{A}, \dot{B}, \dot{C})$.
  If it does not exist, $\mathcal{B}$ does:
  1. Set two empty lists, S_1 and S_2.
  2. Find all elements $(\sigma_i, m_i, \alpha_i)$ in table T_{H_X} such that $\dot{A} = (g_{X0})^{\alpha_i} \bmod (N_X)^2$, and put them into list S_1. If $S_1 = \varnothing$, then output $\perp$ and terminate.
     This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_X(\sigma_i||m_i)$ without querying $\mathcal{O}_{H_X}$. The probability of this event is $q_{H_X}/|\mathbb{Z}_{(N_X)^2}|$.
  3. For every $(\sigma_i, m_i, \alpha_i)$ in list S_1, find all elements in table T_{H_1} such that $\sigma_j = \sigma_i$ and put them (i.e., $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \beta_j)$'s) into list S_2. If $S_2 = \varnothing$, then output $\perp$ and terminate.
     This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_1(\sigma_i)$ without querying $\mathcal{O}_{H_1}$. The probability of this event is $q_{H_1}/2^{k_1}$, where q_{H_1} is the number of queries to $\mathcal{O}_{H_1}$.
  4. Check in S_2 if there exists a $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \beta_j)$ such that $\dot{B} = (g_{X1})^{\alpha_i \cdot (1+\sigma_i N_X)} \bmod (N_X)^2$ and $\beta_j \oplus m_i = \dot{C}$. If none exists or more than one exist, then output $\perp$ and terminate.
  5. Compute $\sigma = \frac{(B/(A' \cdot A^{m_i}) - 1 \bmod (N_Y)^2)}{N_Y}$, $m = C \oplus H_2(\sigma)$. If $A = (g_{Y0})^{H_Y(\sigma||m)} \bmod (N_Y)^2$, output m, where $pk_Y = (H_Y(\cdot), N_Y, g_{Y0}, g_{Y1}, g_{Y2})$ is the corresponding delegator's public key; otherwise, output $\perp$ and terminate.

  If it does exist, $\mathcal{B}$ checks $A' \stackrel{?}{=} A^{\bigstar_3}$. If not, output $\perp$ and terminate; otherwise,
  1. Set two empty lists, S_1 and S_2.
  2. Find all elements $(\sigma_i, m_i, \alpha_i)$ in table T_{H_Y} such that $A = (g_{Y0})^{\alpha_i} \bmod (N_Y)^2$, and put them into list S_1. If $S_1 = \varnothing$, then output $\perp$ and terminate.
     This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_Y(\sigma_i||m_i)$ without querying $\mathcal{O}_{H_Y}$. The probability of this event is $q_{H_Y}/|\mathbb{Z}_{(N_Y)^2}|$, where q_{H_Y} is the number of queries to $\mathcal{O}_{H_Y}$.
  3. For every $(\sigma_i, m_i, \alpha_i)$ in list S_1, find all elements in table T_{H_2} such that $\sigma_j = \sigma_i$ and put them (i.e., $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \gamma_j)$'s) into list S_2. If $S_2 = \varnothing$, then output $\perp$ and terminate.
     This step makes this oracle be distinguishable from the real execution when the adversary can guess the correct value of $H_2(\sigma_i)$ without querying $\mathcal{O}_{H_2}$. The probability of this event is $q_{H_2}/2^n$.
  4. Check in S_2 if there exists a $(\sigma_i, m_i, \alpha_i)||(\sigma_j, \gamma_j)$ such that $(g_{Y1})^{\alpha_i \cdot (1+\sigma_i N_Y)} \bmod (N_Y)^2 = B$ and $\gamma_j \oplus m_i = C$. If none exists or more than one exist, then output $\perp$ and terminate; otherwise, output m_i.

Challenge: At some point, $\mathcal{A}$ outputs a challenge tuple (pk^*, m_0, m_1). If pk^* is not the public key $\mathcal{B}$ guessed in oracle $\mathcal{O}_{pk}$, $\mathcal{B}$ reports *failure* and aborts. Otherwise, $\mathcal{B}$ responds choosing random $d \in \{0,1\}, \sigma \in \mathbb{Z}_N$ and setting:

$$A^* = \mathbf{g}^{\mathbf{v}} \bmod \mathbf{N}^2,\ B^* = \mathbf{T}(1 + m_d\mathbf{N}) \bmod \mathbf{N}^2,$$
$$C^* = H_2(\sigma) \oplus m_d,\ D^* = (\mathbf{g}^{\mathbf{v}})^w \bmod \mathbf{N}^2.$$

And then $\mathcal{B}$ chooses two random numbers $c^* \in \{0,1\}^{k_2}$, $s^* \in \{0, \cdots, 2^{L(\mathbf{N}^2)+k_2} - 1\}$, computes $E^* = (\mathbf{g})^{s^*} A^{*c^*} \bmod \mathbf{N}^2$ and $F^* = (\mathbf{g}^{\mathbf{u}})^{s^*} D^{*c^*} \bmod \mathbf{N}^2$, and checks wether $(A^*, D^*, \mathbf{g}^{\mathbf{u}}, \mathbf{g}^w, E^*, F^*, B^*, C^*, \bigstar)$ exists in table T_{H_3}. If yes, $\mathcal{B}$ reports *failure* and aborts; otherwise, $\mathcal{B}$ outputs $(A^*, B^*, C^*, D^*, c^*, s^*)$, and records $(A^*, D^*, \mathbf{g}^{\mathbf{u}}, \mathbf{g}^w, E^*, F^*, B^*, C^*, c^*)$ in table T_{H_3}.

Phase 2:

$\mathcal{O}_{pk}$: $\mathcal{B}$ responds as in Phase 1.

$\mathcal{O}_{sk}$: On input pk_i, if $pk_i = pk^*$, or (pk^*, pk_i) is in table T_{rk}, then $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ responds as in Phase 1.

$\mathcal{O}_{rk}$: On input (pk_i, pk_j), if $pk_i = pk^*$, and pk_j is in table T_{sk}, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ responds as in Phase 1.

$\mathcal{O}_{re}$: On input (pk_i, pk_j, K), if $(pk_i, K) = (pk^*, K^*)$ and pk_j is in table T_{sk}, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ responds as in Phase 1, except when $pk_i = pk^*$ and $(A, B, C, D, c, s) = (A^*, B^*, C^*, D^*, c^*, s^*)$, $\mathcal{B}$ should record the result $(pk_j, A', C, D, \dot{A}, \dot{C}, \dot{D})$ in table T_{der}, where the derivatives of the challenge ciphertext are recorded.

$\mathcal{O}_{dec}$: On input (pk_i, K), if $(pk_i, K) = (pk^*, K^*)$, or (pk_i, K) is in T_{der}, or $K = \texttt{ReEnc}(\mathcal{O}_{rk}(pk^*, pk_i), K^*)$, then $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ responds as in Phase 1.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $d' \in \{0,1\}$. If $d = d'$, then $\mathcal{B}$ outputs 1 (i.e., DDH instance), otherwise, $\mathcal{B}$ outputs 0 (i.e., not a DDH instance).

Firstly, we analyze the probability of $\mathcal{B}$ do not abort due to the *failure* events, which are as follows.

1. $\mathcal{B}$ did not guess the right attacked public key.
2. The record $(A^*, D^*, \mathbf{g}^{\mathbf{u}}, \mathbf{g}^w, E^*, F^*, B^*, C^*, \bigstar)$ is in table T_{H_3} before Challenge phase.

Suppose $\mathcal{A}$ makes a total of q_{pk} queries to public key generation oracle, q_{rk} queries to re-encryption key generation orale, q_{de} queries to decryption orale, q_H queries to H hash function oracle, q_{H_1} queries to H_1 hash function oracle, q_{H_2} queries to H_2 hash function oracle, and q_{H_3} queries to H_3 hash function oracle.

The probabilities that $\mathcal{B}$ does not abort due to the first *failure* event and the second *failure* event are $1/q_{pk}$ and $1-(q_{H_3}+1)/2^{k_2}$, respectively. Therefore, the probability that $\mathcal{B}$ does not abort due to the *failure* events during the simulation is $(1-(q_{H_3}+1)/2^{k_2})/q_{pk}$.

Secondly, oracles $\mathcal{O}_{re}$ and $\mathcal{O}_{dec}$ are indistinguishable from the corresponding real executions with probabilities at least

$$\left(\frac{1+q_{max}+(q_{max})^2}{(1+q_{max})^2} + \frac{q_{max}}{(1+q_{max})^2}\left(1-\frac{q_{H_X}}{|\mathbb{Z}_{N_{mX}}|}\right)\left(1-\frac{q_{H_1}}{2^{k_1}}\right)\right)^{q_{re}} > \left(\frac{1+q_{max}+(q_{max})^2}{(1+q_{max})^2}\right)^{q_{re}}$$

and

$$\left(\frac{q_{max}}{1+q_{max}} + \frac{1}{1+q_{max}}\left(1-\frac{q_{H_X}}{|\mathbb{Z}_{N_{mX}}|}\right)(1-q_2)\right)^{q_{de}} > \left(\frac{q_{max}}{1+q_{max}}\right)^{q_{de}},$$

respectively, where q_{H_X} is the amount of queries to the same kind of oracle $\mathcal{O}_H$, N_{mX} is the largest number among users' public key N's, $|\mathbb{Z}_{N_{mX}}|$ is the size of $\mathbb{Z}_{N_{mX}}$, and $q_2 = \max\{\frac{q_{H_1}}{2^{k_1}}, \frac{q_{H_2}}{2^n}\}$.

Finally, in the re-encryption oracle, we assume that the signature of knowledge is secure, hence, we should minus the probability of breaking the signature of knowledge ξ.

As a result, $\mathcal{B}$'s advantage is at least

$$\epsilon \cdot \frac{\left(1-(q_{H_3}+1)/2^{k_2}\right)\cdot\left(1+q_{max}+(q_{max})^2\right)^{q_{re}}\cdot(q_{max})^{q_{de}}}{q_{pk}\cdot(1+q_{max})^{2q_{re}+q_{de}}}-\xi$$

and its running time is at most

$$t+\mathcal{O}(3q_{pk}+(7+q_H)q_{rk}+(5+q_H)q_{de})t_e,$$

where t_e is the time of computing one exponentiation in a cyclic group of quadratic residues modulo, and we only consider the exponentiation computation. □

Note that if we modifies B as computed by $H'((g_2)^r)\cdot(1+\sigma\cdot N)$, where H' is another user's own hash function, $H'(\cdot):\{0,1\}^*\rightarrow\mathbb{Z}_{N^2}$, we can get a new PRE scheme proven secure against chosen ciphertext based on CDH assumption over $\mathbb{Z}^*_{N^2}$ and secure signature of knowledge.[9] The proof is almost the same as that of Theorem 1, but the probability of solving CDH assumption will be $1/q_{H'}$ of that of solving DDH assumption, where $q_{H'}$ is the number of queries to oracle H'.

Theorem 2 (Uni-PRE-CR security) *In the random oracle, if N is hard to factor, then scheme $\mathfrak{U}$ is collusion resistant.*

Proof. One can easily show that an algorithm for against scheme $\mathfrak{U}$'s collusion resistance, i.e., from an algorithm that it is given $(\mathbf{N},\mathbf{g_0},\mathbf{g_1},\mathbf{g_2},\mathbf{a},N(=pq),p(=2p'+1),q(=2q'+1),p',q',g_0,g_1,g_2)$, it can compute $\mathbf{p},\mathbf{q},\mathbf{p'},\mathbf{q'}$, such that $\mathbf{N}=\mathbf{pq}$, $\mathbf{p}=\mathbf{2p'+1}$, and $\mathbf{q}=\mathbf{2q'+1}$, we can easily get another algorithm for factoring $\mathbf{N}$. □

4 Scheme $\mathfrak{U}_T$ with Temporary Delegation

This section describes scheme $\mathfrak{U}_T$, a variant of scheme $\mathfrak{U}$, supporting temporary delegation. Like the temporary unidirectional PRE schemes in [2, 3, 25], the proxy is only allowed to transform ciphertexts from the delegator to the delegatee during a limited time period. The point of modifying scheme $\mathfrak{U}$ to scheme $\mathfrak{U}_T$ is to make different g_1's for every time period.

Scheme $\mathfrak{U}_T$ also contains three cryptographic hash functions for all users: $H_1(\cdot):\{0,1\}^*\rightarrow\{0,1\}^{k_1}$, $H_2(\cdot):\{0,1\}^*\rightarrow\{0,1\}^n$, and $H_3(\cdot):\{0,1\}^*\rightarrow\{0,1\}^{k_2}$, where k_1 and k_2 are the security parameter, n is the bit-length of messages to be encrypted. The details are as follows.

`KeyGen`: Choose a safe-prime modulus $N=pq$, $T+2$ random numbers $\alpha\in\mathbb{Z}^*_{N^2}$, $a_1,\cdots,a_T,b\in[1,pp'qq']$, a hash function $H(\cdot)$, where $p=2p'+1$, $q=2q'+1$, p, p', q, q' are primes, T is the number of time intervals, and $H(\cdot):\{0,1\}^*\rightarrow\mathbb{Z}_{N^2}$. Furthermore, set $g_0=\alpha^2 \bmod N^2$, $g_1^{(i)}={g_0}^{a_i} \bmod N^2$ $(i=1,\cdots,T)$, and $g_2={g_0}^b \bmod N^2$. The public key is $pk=(H(\cdot),N,g_0,g_1^{(i)}\ (i=1,\cdots,T),g_2)$, the "weak" secret key is $(a_i\ (i=1,\cdots,T),b)$, and the long-term secret key is $sk=(p,q,p',q')$.

`ReKeyGen`: On input a public key $pk_Y=(H_Y(\cdot),N_Y,g_{Y0},g_{Y1}^{(1)},\cdots,g_{Y1}^{(T_Y)},g_{Y2})$, a "weak" secret key $a_{X,j}$ for time period $j\in\{1,\cdots,T_X\}$, and a secret key $sk_X=(p_X,q_X,p'_X,q'_X)$, it outputs the unidirectional re-encryption key $rk_{X\rightarrow Y,j}=(rk^{(1)}_{X\rightarrow Y,j},rk^{(2)}_{X\rightarrow Y,j})$ for the j-th time period, which is generated as follows.

- Choose two random numbers $\dot{\sigma}_j\in\mathbb{Z}_N$, $\dot{\beta}_j\in\{0,1\}^{k_1}$.
- Compute $rk^{(2)}_{X\rightarrow Y,j}=a_{X,j}-\dot{\beta}_j \bmod (p_Xq_Xp'_Xq'_X)$.
- Compute

$$r_{X\rightarrow Y,j}=H_Y(\dot{\sigma}_j||\dot{\beta}_j),\quad \dot{A}_j=(g_{Y0})^{r_{X\rightarrow Y,j}} \bmod (N_Y)^2,$$
$$\dot{B}_j=(g_{Y2})^{r_{X\rightarrow Y,j}}\cdot(1+\dot{\sigma}_jN_Y) \bmod (N_Y)^2,\quad \dot{C}_j=H_1(\dot{\sigma}_j)\oplus\dot{\beta}_j$$

[9] We thank Eike Kiltz to point out this to us.

- Set $rk^{(1)}_{X\to Y,j} = (\dot{A}_j, \dot{B}_j, \dot{C}_j)$.

`Enc`: On input a public key $pk = (H(\cdot), N, g_0, g_1^{(1)}, \cdots, g_1^{(T)}, g_2)$, a time period $j \in \{1, \cdots, T\}$ and a message $m \in \{0,1\}^n$, the encryptor does the following performances:

- Choose a random number $\sigma_j \in \mathbb{Z}_N$.
- Compute

$$r_j = H(\sigma_j||m),$$

$$A_j = (g_0)^r \bmod N^2, \quad B_j = (g_1^{(j)})^r \cdot (1+\sigma_j N) \bmod N^2,$$
$$C_j = H_2(\sigma_j) \oplus m, \quad D_j = (g_2)^r \bmod N^2.$$

- Run $(c_j, s_j) \leftarrow \texttt{SoK.Gen}(A_j, D_j, g_0, g_2, (B_j, C_j))$, where the underlying hash function is H_3.
- Output the ciphertext $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$ for the j-th time period .

`ReEnc`: On input a re-encryption key $rk_{X\to Y,j} = (rk^{(1)}_{X\to Y,j}, rk^{(2)}_{X\to Y,j})$ and a ciphertext $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$ under key $pk_X = (H_X(\cdot), N_X, g_{X0}, g_{X1}^{(1)}, \cdots, g_{X1}^{(T_X)}, g_{X2})$, where $j \in \{1, \cdots, T_X\}$, the proxy checks whether $c_j = H_3(A_j||D_j||g_{X0}||g_{X2}||(g_{X0})^{s_j}(A_j)^{c_j}||(g_{X2})^{s_j}(D_j)^{c_j}||(B_j||C_j))$. If not hold, output $\perp$ and terminate; otherwise, re-encrypt the ciphertext to be under key pk_Y as:

- Compute $A'_j = (A_j)^{rk^{(2)}_{X\to Y,j}} = (g_{X0})^{r(a_{X,j}-\dot{\beta}_j)} \bmod (N_X)^2$.
- Output the new ciphertext

$$(A_j, A'_j, B_j, C_j, rk^{(1)}_{X\to Y,j}) = (A_j, A'_j, B_j, C_j, \dot{A}_j, \dot{B}_j, \dot{C}_j).$$

`Dec`: On input a secret key and any ciphertext K_j for the j-th time period, where $j \in \{1, \cdots, T\}$, the decryptor parses $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$, or $K_j = (A_j, A'_j, B_j, C_j, \dot{A}_j, \dot{B}_j, \dot{C}_j)$.

Case $K_j = (A_j, B_j, C_j, D_j, c_j, s_j)$**:** Check whether $c_j = H_3(A_j||D_j||g_0||g_2||(g_0)^{s_j}(A_j)^{c_j}||(g_2)^{s_j}D_j{}^{c_j}||(B_j||C_j))$, if not, output $\perp$ and terminate; otherwise,

- if the input secret key is the "weak" secret key a_j, compute $\sigma_j = \frac{B_j/((A_j)^{a_j})-1 \bmod N^2}{N}$.
- if the secret key is the long term secret key (p, q, p', q'), compute $\sigma_j = \frac{(B_j/(g_0)^{w_1})^{2p'q'}-1 \bmod N^2}{N} \cdot \pi(\bmod N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $m = C_j \oplus H_2(\sigma_j)$, if $B_j = (g_1^{(j)})^{H(\sigma_j||m)} \cdot (1+\sigma_j \cdot N) \bmod N^2$ holds, output m; otherwise, output $\perp$ and terminate.

Case $K_j = (A_j, A'_j, B_j, C_j, \dot{A}_j, \dot{B}_j, \dot{C}_j)$**:** In this case, the decryptor should know the delegator's (Alice's) public key $(H'(\cdot), N', g'_0, {g_1^{(i)}}', \cdots, {g_1^{(T')}}', g'_2)$.

- If the input secret key is the "weak" secret key b, compute $\dot{\sigma}_j = \frac{\dot{B}_j/((\dot{A}_j)^b)-1 \bmod N^2}{N}$.
- If the input secret key is the long term secret key (p, q, p', q'), computes $\dot{\sigma}_j = \frac{(\dot{B}_j/g_0^{w_1})^{2p'q'}-1 \bmod N^2}{N} \cdot \pi(\bmod N)$, where w_1 is computed as that in scheme BCP03, and π is the inverse of $2p'q' \bmod N$.

Compute $\dot{\beta}_j = \dot{C}_j \oplus H_1(\dot{\sigma}_j)$, if $\dot{B}_j = (g_2)^{H(\dot{\sigma}_j||\dot{\beta}_j)} \cdot (1+\dot{\sigma}_j N) \bmod N^2$ holds, then compute $\sigma_j = \frac{B_j/(A'_j \cdot (A_j)^{\beta_j})-1 \bmod N'^2}{N'}$, $m = C_j \oplus H_2(\sigma_j)$; otherwise, output $\perp$ and terminate. If $B_j = ({g_1^{(j)}}')^{H'(\sigma_j||m)} \cdot (1+\sigma_j \cdot N') \bmod N'^2$ holds, then output m; otherwise, output $\perp$ and terminate.

Note that $(H(\cdot), N, g_0, g_1^{(i)}, \cdots, g_1^{(T)}), g_2)$ is the public key of the decryptor.

Table 1: Comparison between scheme $\mathfrak{U}$ and scheme LV08.

Schemes		LV08	$\mathfrak{U}$
Comput. Cost	ReKeyGen	$2t_{eb}$	$2t_{eN}$
	Enc	$3.5t_{eb} + 1t_s$	$5t_{eN}$
	ReEnc	$2t_p + 4t_{eb} + 1t_v$	$4t_{eN}$
	Dec Original	$3t_p + 2t_{eb} + 1t_v$	$5t_{eN}$
	Dec Transformed	$5t_p + 2t_{eb} + 1t_v$	$4t_{eN}$
Ciphertext Size	Original	$1\|svk\| + 2\|\mathbb{G}_e\| + 1\|\mathbb{G}_T\| + 1\|\sigma\|$	$2k + 3\|{N_X}^2\| + \|m\|$
	Transformed	$1\|svk\| + 4\|\mathbb{G}_e\| + 1\|\mathbb{G}_T\| + 1\|\sigma\|$	$k_1 + 3\|{N_X}^2\| + 2\|{N_Y}^2\| + \|m\|$
Security	Security Level	collusion resistant, RCCA	collusion resistant, CCA
	Standard model	Yes	No
	Underlying Assumptions	3-QDBDH	DDH

Correctness. The correctness property is easily obtained by the same method for scheme $\mathfrak{U}$.

Theorem 3 (Uni-PRETD-CCA Security) *In the random oracle model, scheme $\mathfrak{U}_T$ is CCA-secure under the assumptions that DDH problem over $\mathbb{Z}^*_{N^2}$ is hard, and that the signature of knowledge is secure.*

Proof. In this proof, $\mathcal{B}$ does not only guess which public key is the attacked public key, but also guess which time period is the attacked time period. We set $\mathbf{N}$ as the safe-prime modulus of the target public key, $g_0 = \mathbf{g}$, and set $\mathbf{g^u}$ as the public parameters of the attacked time period. The rest of the simulation can be proceeded by the same method in the proof of Theorem 1.

The probability of this proof is $1/q_T$ of that in the proof of Theorem 1, where q_T is the amount of time periods of the attacked time period. □

Theorem 4 (Uni-PRETD-CR security) *In the random oracle, if $\mathbf{N}$ is hard to factor, then scheme $\mathfrak{U}_T$ is collusion resistant.*

Proof. It is easy to get this proof, since we can know "weak" secret key which can be used to respond all kinds of queries. Once the adversary outputs the long-term secret key (p, q, p', q'), we get the factors of $\mathbf{N}$.□

5 Comparison

In this section, we compare scheme $\mathfrak{U}$ with the previous CCA-secure unidirectional PRE schemes. Since as mentioned above, the unidirectional PRE schemes in [21, 17, 11, 13] are not CCA-secure, we only compare scheme $\mathfrak{U}$ with the scheme in [25] (named LV08).

In Table 1, we denote t_p, t_{eb}, t_{eN}, t_s, and t_v as the computational cost of a bilinear pairings, an exponentiation over a bilinear group, an exponentiation over $\mathbb{Z}^*_{N^2}$ (N is a safe-prime modulus), a one-time signature and verification, respectively. $\mathbb{G}_e$ and $\mathbb{G}_T$ are the bilinear groups used in scheme LV08. N_X and N_Y are the safe-prime modulus corresponding to the delegator and the delegatee, respectively. svk and σ are the one-time signature's public key and signature. Note that we only consider the case of using weak secret key to decrypt in Dec algorithm of scheme $\mathfrak{U}$.

From Table 1, we can see that scheme LV08 is a little bit more efficient than scheme $\mathfrak{U}$. In order to guarantee that N is hard to factor, N should be 1024-bit at least, which makes scheme $\mathfrak{U}$ need more time for an exponentiation and more storage for a ciphertext. However, we emphasize that scheme $\mathfrak{U}$ is CCA-secure and based on the well-studied DDH assumption, while scheme LV08 is RCCA-secure and based on the less-studied 3-quotient decision Bilinear Diffie-Hellman (3-QDBDH) assumption.

6 Conclusions

In this paper, by using signature of knowledge and Fijisaki-Okamoto conversion, we proposed the *first* CCA-secure and collusion resistant unidirectional PRE scheme without pairings, which solves a problem proposed in [9, 25].

There are still many open problems to be solved, such as designing more efficient CCA-secure, collusion resistant unidirectional PRE schemes without pairings, and CCA-secure multi-use unidirectional PRE schemes [9, 25].

Acknowledgements

We thank PKC 2009 chair Stanislaw Jarecki and the anonymous reviewers of PKC 2009 for insightful comments and helpful suggestions.

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270, 2000.

[2] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Internet Society (ISOC): NDSS 2005*, pages 29–43, 2005.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18, 2001.

[5] H.Y. Lynn B. Scott M. Barreto, P.S.L.M. Kim. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO*, volume 2442 of *LNCS*, pages 354–369, 2002.

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

[7] E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 37–54, 2003.

[8] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO 1997*, volume 1296 of *LNCS*, pages 410–424, 1997.

[9] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *ACM CCS 2007*, 2007. Full version: Cryptology ePrint Archieve: Report 2007/171.

[10] Y-P. Chiu, C-L. Lei, and C-Y. Huang. Secure multicast using proxy encryption. In *ICICS 2005*, volume 3783 of *LNCS*, pages 280–290, 2005.

[11] C. Chu and W. Tzeng. Identity-based proxy re-encryption without random oracles. In *ISC 2007*, volume 4779 of *LNCS*, pages 189–202, 2007.

[12] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO 1998*, volume 1462 of *LNCS*, pages 13–25, 1998.

[13] R.H. Deng, J. Weng, S. Liu, and K. Chen. Chosen-ciphertext secure proxy re-encryption schemes without pairings. In *CANS 2008*, volume 5339 of *LNCS*, pages 1–17, 2008. Full version: http://eprint.iacr.org/2008/509.

[14] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *PKC 1999*, volume 1560 of *LNCS*, pages 53–68, 1999.

[15] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 537–554, 1999.

[16] S. Goldwasser and Y.T. Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS 2005*, pages 553–562, 2005.

[17] M. Green and G. Ateniese. Identity-based proxy re-encryption. In *ACNS 2007*, volume 4521 of *LNCS*, pages 288–306, 2007. Full version: Cryptology ePrint Archieve: Report 2006/473.

[18] S. Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 443–457, 2000.

[19] T.S. Heydt-Benjamin, H. Chae, B. Defend, and K. Fu. Privacy for public transportation. In *PET 2006*, volume 4258 of *LNCS*, pages 1–19, 2005.

[20] S. Hohenberger, G.N. Rothblum, A. Shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. In *TCC 2007*, volume 4392 of *LNCS*, pages 233–252, 2007.

[21] A. Ivan and Y. Dodis. Proxy cryptography revisited. In *Internet Society (ISOC): NDSS 2003*, 2003.

[22] H. Khurana and H-S. Hahm. Certified mailing lists. In *ASIACCS 2006*, pages 46–58, 2006.

[23] H. Khurana and R. Koleva. Scalable security and accounting services for content-based publish subscribe systems. *International Journal of E-Business Research*, 2(3), 2006.

[24] H. Khurana, A. Slagell, and R. Bonilla. Sels: A secure e-mail list service. In *ACM SAC 2005*, pages 306–313, 2005.

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

[26] A. Lysyanskaya, Micali S, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 74–90, 2004.

[27] P. Paillier. Public-key cryptosystems based on discrete logarithms residues. In *EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238, 1999.

[28] C.P. Schnorr. Efficient identifications and signatures for smart cards. In *CRYPTO 1998*, volume 435 of *LNCS*, pages 239–251, 1998.

[29] M. Scott. Computing the tate pairing. In *CT-RSA*, volume 3376 of *LNCS*, pages 293–304, 2005.

[30] J. Shao. Proxy re-cryptography revisited. Dec. 2007. PhD Thesis, Shanghai Jiao Tong University.

[31] J. Shao, D. Xing, and Z. Cao. Analysis of cca secure unidirctional id-based pre scheme. 2008. Technical Report of TDT, Shanghai Jiao Tong University.

[32] V. Shoup. Practical threshold signatures. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220, 2000.

[33] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *EUROCRYPT 1998*, volume 1430 of *LNCS*, pages 1–16, 1998.

[34] G. Taban, A.A. Cárdenas, and V.D. Gligor. Towards a secure and interoperable drm architecture. In *ACM DRM 2006*, pages 69–78, 2006.

[35] A. Talmy and O. Dobzinski. Abuse freedom in access control schemes. In *AINA 2006*, pages 77–86, 2006.

[36] Smith. Tony. Dvd jon: buy drm-less tracks from apple itunes. 2005. http://www.theregister.co.uk/2005/03/18/itunes_pymusique.

A Analysis on Ivan-Dodis Construction

A.1 Ivan-Dodis Construction

The Ivan-Dodis construction is based on any CCA-secure PKE. The details are as follows.

UniPRE.KGen: On input the security parameter 1^k, it outputs two key pairs (pk_1, sk_1) and (pk_2, sk_2).

UniPRE.RKGen: On input the delegator's key pairs (pk_1, sk_1) and (pk_2, sk_2), the delegator sends sk_1 as the re-encryption key to the proxy via a secure channel, and sends sk_2 to the delegatee as the partial key via another secure channel.

UniPRE.Enc: On input public keys (pk_1, pk_2) and a message m, it outputs $\mathtt{PKE.Enc}(pk_1, \mathtt{PKE.Enc}(pk_2, m))$.

UniPRE.ReEnc: On input a re-encryption key sk_1 and a ciphertext C, it outputs a re-encrypted ciphertext $C' = \mathtt{PKE.Dec}(sk_1, C)$.

UniPRE.Dec: On input secret keys (sk_1, sk_2), a partial key sk_2' from its delegator and a ciphertext C, UniPRE.Dec does:

- If C is an original ciphertext, then it outputs $\mathtt{PKE.Dec}(sk_2, \mathtt{PKE.Dec}(sk_1, C))$.
- If C is a re-encrypted ciphertext, then it outputs $\mathtt{PKE.Dec}(sk_2', C)$.

Note that the partial key sk_2 can be encrypted by the delegatee's public key, and forwarded to Bob by the proxy. In this case, the delegatee does not require to store extra secrets for every delegation [2, 3].

A.2 Chosen Ciphertext Attacks on the Ivan-Dodis Construction

In this subsection, we will show that the adversary always wins the Uni-PRE-CCA game with the Ivan-Dodis construction's Challenger.

Phase 1: The adversary does not need to make any query in this phase.

Challenge: The adversary outputs two equal length plaintexts m_0, m_1 from the message space, and an uncorrupted public key $pk^* = (pk_1^*, pk_2^*)$.

The Challenger will follow the Uni-PRE-CCA game's specification, i.e., pick a random bit $b \in \{0, 1\}$ and sets $C^* = \mathtt{UniPRE.Enc}(pk^*, m_b)$. It sends C^* as the challenge ciphertext to $\mathcal{A}$.

Phase 2: The adversary performs as follows.

1. The adversary queries $\mathcal{O}_{re}$ with (pk^*, pk, C^*), such that pk is uncorrupted. Then as the Uni-PRE-CCA game's specification, the adversary can get the re-encrypted ciphertext C' such that $C' = \mathtt{PKE.Dec}(sk_1^*, C^*)$, sk_1^* is the key corresponding to pk_1^*.

2. The adversary computes $\hat{C} = \texttt{PKE.Enc}(pk_1^*, C')$. Note that $\hat{C} \neq C^*$ since PKE is CCA-secure, such as the underlying PKE scheme is the Cramer-Shoup scheme [12].
3. The adversary queries $\mathcal{O}_{de}$ with $(pk^*, \hat{C})$ and gets a message m. Note that $(pk^*, \hat{C})$ is not a derivative of (pk^*, C^*), hence this query is valid.

Guess: If $m = m_0$, the adversary $\mathcal{A}$ outputs $b' = 0$; otherwise, output $b' = 1$.

Since $\hat{C}$ and C^* are corresponding to the same message, we always have $b = b'$. As a result, the Ivan-Dodis construction is not CCA-secure for the security model in Section 2.

B Definitions of Unidirectional PRE Schemes with Temporary Delegation

Definition 7 (Unidirectional PRE with Temporary Delegation) *A unidirectional proxy re-encryption scheme* UniPRE *with temporary delegation is a tuple of PPT algorithms* (KeyGen, ReKeyGen, Enc, ReEnc, Dec)*:*

- $\texttt{KeyGen}(1^k) \rightarrow (pk, sk, T)$. On input the security parameter 1^k, the key generation algorithm KeyGen outputs a public/secret key pair (pk, sk), and the number of time intervals T.
- $\texttt{Enc}(pk, m, j) \rightarrow C_j$. On input a public key pk, a message m in the message space, and the time period $j \in \{1, \cdots, T\}$, the encryption algorithm Enc outputs a ciphertext C_j for the j-th time period.
- $\texttt{ReKeyGen}(sk_1, pk_2, j) \rightarrow rk_{1\rightarrow 2,j}$. On input a secret key sk_1, a public key pk_2, and the time period $j \in \{1, \cdots, T_1\}$, where T_1 is the number of time intervals corresponding to the delegator. The re-encryption key generation algorithm ReKeyGen outputs a unidirectional re-encryption key $rk_{1\rightarrow 2,j}$ for the j-th time period.
- $\texttt{ReEnc}(rk_{1\rightarrow 2,j}, C_1^{(j)}) \rightarrow C_2^{(j)}$. On input a re-encryption key $rk_{1\rightarrow 2,}$ and a ciphertext $C_1^{(j)}$ for the j-th time period, where $j \in \{1, \cdots, T_1\}$, T_1 is the number of time intervals corresponding to the delegator. The re-encryption algorithm ReEnc outputs a re-encrypted ciphertext $C_2^{(j)}$ for the j-th time period or $\perp$.
- $\texttt{Dec}(sk, C_j) \rightarrow m$. On input a secret key sk and a ciphertext C_j for the j-th time period, where $j \in \{1, \cdots, T\}$, T is the number of time intervals corresponding to the decryptor. The decryption algorithm Dec outputs a message m in the message space or $\perp$.

B.0.1 Correctness.

A correct proxy re-encryption scheme should satisfy two requirements: $\texttt{Dec}(sk, \texttt{Enc}(pk, m, j)) = m$, and $\texttt{Dec}(sk', \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk', j), C_j)) = m$, where $(pk, sk, T), (pk', sk', T') \leftarrow \texttt{KeyGen}(1^k)$, C_j is the ciphertext of message m for pk and the j-th time period from algorithm Enc or algorithm ReEnc, and $j \in \{1, \cdots, T\}$.

B.0.2 Chosen Ciphertext Security for Unidirectional Proxy Re-Encryption with Temporary Delegation.

Following the method in [25], we extend Uni-PRE-CCA game to Uni-PRETD-CCA game, which is described as follows.

Phase 1: The adversary $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

- $\mathcal{O}_{pk}$, $\mathcal{O}_{sk}$: Identical to those Uni-PRE-CCA game.

- *Re-encryption key generation oracle* $\mathcal{O}_{rk}$: On input (pk, pk', j) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, and the time period $j \in \{1, \cdots, T\}$, the Challenger returns the re-encryption key $rk_{pk \to pk', j} = \texttt{ReKeyGen}(sk, pk', j)$, where sk is the secret key corresponding to pk.
- *Re-encryption oracle* $\mathcal{O}_{re}$: On input (pk, pk', C, j) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, and the time period $j \in \{1, \cdots, T\}$, the re-encrypted ciphertext $C' = \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk', j), C)$ is returned by the Challenger, where sk is the secret key corresponding to pk.
- *Decryption oracle* $\mathcal{O}_{dec}$: On input (pk, C, j), where pk is from $\mathcal{O}_{pk}$, and the time period $j \in \{1, \cdots, T\}$, the Challenger returns $\texttt{Dec}(sk, C)$, where sk is the secret key corresponding to pk.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, a public key pk^*, and the time period j^* on which it wishes to be challenged. There are some constraints on the public key pk^* and j^*: (i) pk^* is from $\mathcal{O}_{pk}$; (ii) pk^* did not appear in any query to $\mathcal{O}_{sk}$ in Phase 1; (iii) if $(pk^*, \bigstar, j^*)$ did appear in any query to $\mathcal{O}_{rk}$, then $\bigstar$ did not appear in any query to $\mathcal{O}_{sk}$. The Challenger picks a random bit $b \in \{0, 1\}$ and sets $C^* = \texttt{Enc}(pk^*, m_b, j)$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ issues more queries $q_{n_1+1}, \cdots, q_n$ where query q_i is one of:

- $\mathcal{O}_{pk}$: The Challenger responds as in Phase 1.
- $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk is from $\mathcal{O}_{pk}$;
  - $pk \neq pk^*$;
  - (pk^*, pk, j^*) is not a query to $\mathcal{O}_{rk}$ before;
  - (pk', pk, C', j^*) is not a query to $\mathcal{O}_{re}$ before, where (pk', C', j^*) is a $\texttt{derivative}$[10] of (pk^*, C^*, j^*).
- $\mathcal{O}_{rk}$: On input (pk, pk', j) by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if $pk = pk^*$ and $j = j^*$, then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{re}$: On input (pk, pk', C, j) by $\mathcal{A}$, if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk, pk' are from $\mathcal{O}_{pk}$;
  - if (pk, C, j) is a derivative of (pk^*, C^*, j^*), then pk' is not a query to $\mathcal{O}_{sk}$.
- $\mathcal{O}_{dec}$: On input (pk, C, j), if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger terminates the game.
  - pk is from $\mathcal{O}_{pk}$;
  - (pk, C, j) is not a derivative of (pk^*, C^*, j^*).

These queries may be also asked adaptively.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as a Uni-PRETD-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking $\texttt{UniPRE}$ as the following function of the security parameter k: $\mathrm{Adv}_{\texttt{UniPRE},\mathcal{A}}(k) = |\Pr[b = b'] - 1/2|$. Using the Uni-PRE-CCA game we can define chosen ciphertext security for unidirectional proxy re-encryption schemes.

[10] Derivatives of (pk^*, C^*, j^*) are defined similarly with that in Section 2.2, and just add j^* into every input/output.

Definition 8 (Uni-PRETD-CCA security) *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *with temporary delegation is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time Uni-PRETD-CCA adversary* $\mathcal{A}$ *the function* $Adv_{\mathtt{UniPRE},\mathcal{A}}(k)$ *is negligible. As shorthand, we say that* `UniPRE` *is Uni-PRETD-CCA secure.*

Definition 9 (Uni-PRETD-CR security) *We say that a unidirectional proxy re-encryption scheme* `UniPRE` *with temporary delegation is collusion resistant if for any polynomial bounded adversary* $\mathcal{A}$*, the following probability is negligible:*

$$\begin{aligned} Pr[(sk_1, pk_1, T_1) \leftarrow \mathtt{KeyGen}(1^k), \{(sk_i, pk_i, T_i) \leftarrow \mathtt{KeyGen}(1^k)\}, \\ \{rk_{i\rightarrow 1,j} \leftarrow \mathtt{ReKeyGen}(sk_i, pk_1, j)\}\ (j = 1, \cdots, T_i), \\ \{rk_{1\rightarrow i,j} \leftarrow \mathtt{ReKeyGen}(sk_1, pk_i, j)\}\ (j = 1, \cdots, T_1), \\ i = 2, \cdots, \\ \alpha \leftarrow \mathcal{A}(pk_1, \{pk_i, sk_i\}, \{rk_{1\rightarrow i,j}\}, \{rk_{i\rightarrow 1,j}\}) : \\ \alpha = sk_1]. \end{aligned}$$