A Proxy Re-Encryption Scheme with the Unforgeability of Re-Encryption Keys against Collusion Attacks

Ryotaro Hayashi and Tatsuyuki Matsushita

Toshiba Corporation
1, Komukai Toshiba-Cho, Saiwai-Ku, Kawasaki-Shi, Kanagawa 212-8582, Japan
{ryotaro.hayashi, tatsuyuki.matsushita}@toshiba.co.jp

Abstract

Proxy re-encryption (PRE) schemes are cryptosystems which allow a proxy who has a re-encryption key to convert a ciphertext originally encrypted for one party into a ciphertext which can be decrypted by another party. In [1], Hayashi et al. proposed the new security notion for PRE called "unforgeability of re-encryption keys against collusion attacks," UFReKey-CA for short. They proposed the PRE schemes and claimed that their schemes meet UFReKey-CA. However, Isshiki et al. [2] pointed out that the schemes do not meet UFReKey-CA in IWSEC 2013. It is an open problem of constructing the scheme which meets UFReKey-CA. In this paper, we propose new PRE schemes which meet confidentiality (RCCA security) assuming that the q-wDBDHI problem is hard and meet UFReKey-CA assuming that the 2-DHI problem is hard.

Keywords: Proxy re-encryption, non-transferability, unforgeability of re-encryption keys against collusion attacks, UFReKey-CA.

1 Introduction

Proxy re-encryption(PRE) schemes introduced by Blaze, Bleumer, and Strauss [3], are cryptosystems with the following special property. Alice, the original recipient of some ciphertext, can delegate the decryption rights to Bob by creating a *re-encryption key* then giving it to a semi-trusted entity called *proxy*. Consequently, Alice lets the proxy to convert ciphertexts for Alice into ciphertexts for Bob without revealing any information about the underlying plaintexts to the proxy. In PRE, Alice and Bob are called a *delegator* and a *delegatee*, respectively.

Needless to say, the basic security property of PRE is confidentiality (indistinguishability). After Blaze et al. introduced the concept of PRE, many concrete PRE schemes with high confidentiality (e.g. RCCA security [4, 5, 6, 7], full CCA (or more strong) security [8, 9, 10]) have been proposed. In the definition of the confidentiality, it is assumed that proxies who have the re-encryption keys for converting ciphertexts of the (target) honest delegators are not corrupted.

In addition to the above basic security property, it is also important to consider the security where such proxies are corrupted. As an example of such kind of security notions, Hayashi et al. [1] introduced *the unforgeability of re-encryption keys against collusion attack (UFReKey-CA)*. Roughly speaking, UFReKey-CA means that even colluding proxies and delegatees cannot generate a re-encryption key which convert ciphertexts for delegator into those for some (malicious)

user. They also proposed PRE schemes and claimed that their scheme meets UFReKey-CA under the assumption that the 2-Diffie-Hellman inversion with randomized answers (2DHIwRA) problem, originally proposed in their paper, is hard. The 2DHIwRA problem is the 2-DHI problem with some additional inputs related to the instance of 2-DHI problem. The additional inputs are used to generate (simulate) the re-encryption keys in the UFReKey-CA proof of the scheme by Hayashi et al. However, Isshiki et al. [2] pointed out that the 2DHIwRA problem is not hard since 2-DHI problem can be solved by using the additional inputs in the 2DHIwRA problem. Isshiki et al. also showed that the scheme in [1] does not meet UFReKey-CA. More precisely, a re-encryption key can be forged when a proxy and two or more delegatees collude. They also mentioned that it is an open problem of constructing the scheme which meets UFReKey-CA.

In this paper, we propose a new (unidirectional single-hop) PRE scheme which meets confidentiality (RCCA security) assuming that the 3-wDBDHI problem is hard and meets UFReKey-CA assuming that the 2-DHI problem is hard. Our scheme is based on the scheme in [1]. To prevent re-encryption key forgery, we change the form of the re-encryption key. We mask every secret key of the delegatee in the exponents of the re-encryption keys with a randomness (or a system secret parameter). Due to this change, we also change the verification equation for first level ciphertexts. In our scheme, the secret key is required to verify the first level ciphertext. Interestingly, in the proof of security, we can simulate the evaluation of whether the equation holds or not without knowing the secret key. Further, we prove that our scheme meets UFReKey-CA under the assumption that the 2-DHI problem is hard. Namely, in the proof of our scheme, we can generate (simulate) the re-encryption key directly from the instance of 2-DHI problem (not required some additional inputs). In this paper, we also propose a scheme which is an extension of the first scheme with temporary delegation, which limits the duration of re-encryption key within a certain time interval. This scheme meets confidentiality (RCCA-CK security) assuming the 1-wDBDHI problem is hard and meets UFReKey-CA assuming that 2-DHI problem is hard.

This paper is organized as follows. In Section 2, we review the definitions related to our proposal. We propose a concrete PRE scheme and prove its security in Section 3, and that supporting temporary delegation in Section 4. Concluding remarks are shown in Section 5.

2 Preliminaries

2.1 Bilinear Maps and Complexity Assumptions

2.1.1 Bilinear Maps

Groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p are called bilinear map groups if there exists a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties: (1)bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, (2) $e(\cdot, \cdot)$ is efficiently computable for any input pair, and (3) non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g \neq 1_{\mathbb{G}_1}$ and $h \neq 1_{\mathbb{G}_2}$. We say that the pairing is symmetric if $\mathbb{G}_1 = \mathbb{G}_2$ and it is asymmetric if $\mathbb{G}_1 \neq \mathbb{G}_2$.

2.1.2 Complexity Assumptions

We describe q-weak the decision bilinear Diffie–Hellman Inversion (q-wDBDHI) problem. The confidentiality of the PRE schemes in [4] and [1] are proved by assuming the hardness of this problem (symmetric version). By using the framework in [11], we can prove this problem is hard in the generic group model.

Definition 1 (q-wDBDHI problem)**.** *The q-weak decision bilinear Diffie–Hellman Inversion problem is to distinguish the two distributions*

- $(g, g^a, g^{a^2}, \ldots, g^{a^q}, g^b, h, h^a, h^{a^2}, \ldots, h^{a^q}, h^b, e(g,h)^{b/a})$, *and*
- $(g, g^a, g^{a^2}, \ldots, g^{a^q}, g^b, h, h^a, h^{a^2}, \ldots, h^{a^q}, h^b, e(g,h)^z)$

where $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $a, b, z \xleftarrow{R} \mathbb{Z}_p^*$.

We next describe 2-Diffie–Hellman inversion (2-DHI) problem. The hardness of this problem was used to prove the security of the schemes in [12, 13, 14] [1]. By using the framework in [11], we can prove this problem is hard in the generic group model.

Definition 2 (2-DHI problem)**.** *The 2-Diffie–Hellman inversion problem is, given $g, g^a, g^{a^2}, h, h^a, h^{a^2}$, computing $h^{1/a}$ for $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $a \xleftarrow{R} \mathbb{Z}_p^*$.*

In case of symmetric version of the above problems, the generator h is replaced by g.

2.2 Strong One-Time Signature

We review the strong one-time signature which we employ to construct our scheme.

One-time signature $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ consists of a triple of algorithms. The algorithm $\mathcal{G}$ takes a security parameter λ and returns a pair of signing/verification keys (ssk, svk). Then, for any message M, $\mathcal{V}(\sigma, svk, M)$ returns 1 whenever $\sigma = \mathcal{S}(ssk, M)$ and 0 otherwise.

We say that Sig is a strong one-time signature (or Sig meets strong unforgeability) if no polynomial time adversary can create a new signature for a previously signed message. The definition of the strong one-time signature [4] is as follows.

Definition 3. *We say that $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strong one-time signature (or Sig meets strong unforgeability) if for any polynomial time algorithm $\mathcal{F}$,*

$$\Pr[(ssk, svk) \leftarrow \mathcal{G}(\lambda); (M, St) \leftarrow \mathcal{F}(svk); \sigma \leftarrow \mathcal{S}(ssk, M); (M', \sigma') \leftarrow \mathcal{F}(M, \sigma, svk, St) : \mathcal{V}(\sigma', svk, M') = 1 \wedge (M', \sigma') \neq (M, \sigma)]$$

is negligible.

3 A UFReKey-CA Secure Proxy Re-Encryption Scheme

In this section, we propose a unidirectional PRE scheme which meets the RCCA security and sUFReKey-CA.

3.1 Definitions

In this section, we describe the syntactic definition of unidirectional proxy re-encryption [15, 16] and its security notion [4, 1].

[1] In [12], 2-DHI problem is called as Weak Diffie–Hellman problem.

3.1.1 Model

First, we describe the syntactic definition of unidirectional proxy re-encryption.

Definition 4. *A (single-hop) unidirectional proxy re-encryption (PRE) scheme consists of the following algorithms:*

$\mathsf{Global\text{-}setup}(\lambda)$ *is a probabilistic algorithm which takes a security parameter* λ *and returns public parameters* par *with a plaintext space* $\mathcal{M}$*.*

$\mathsf{Keygen}(\lambda, \mathsf{par})$ *is a probabilistic algorithm which takes parameters* λ *and* par *and returns a public/secret key pair* (pk, sk)*.*

$\mathsf{Enc}_1(m, pk_j, \mathsf{par})$ *is a probabilistic algorithm which takes a plaintext* $m \in \mathcal{M}$*, a user* j*'s public key* pk_j*, and* par*, and returns a first level ciphertext* C_j *for* j*, which cannot be re-encrypted for another user.*

$\mathsf{Enc}_2(m, pk_i, \mathsf{par})$ *is a probabilistic algorithm which takes a plaintext* $m \in \mathcal{M}$*, a user* i*'s public key* pk_i*, and* par*, and returns a second level ciphertext* C_i *for* i*, which can be re-encrypted with re-encryption keys for another user.*

$\mathsf{Rekeygen}(sk_i, pk_j, \mathsf{par})$ *is a probabilistic algorithm which takes a user* i*'s secret key* sk_i*, a user* j*'s public key* pk_j*, and* par*, and returns a re-encryption key* R_{ij} *to re-encrypt second level ciphertexts for* i *into first level ciphertexts for* j*.*

$\mathsf{Reenc}(R_{ij}, C_i, \mathsf{par})$ *is a probabilistic algorithm which takes a re-encryption key* R_{ij}*, a second level ciphertext* C_i *encrypted under* pk_i*, and public parameters* par*, and returns a first level ciphertext* C_j *re-encrypted for* j *or a distinguished message '*`invalid`*.'*

$\mathsf{Dec}_1(C_j, sk_j, \mathsf{par})$ *is a deterministic algorithm which takes a first level ciphertext* C_j *for* j*, a user* j*'s secret key* sk_j*, and* par*, and returns a plaintext* m *or a distinguished message '*`invalid`*'.*

$\mathsf{Dec}_2(C_i, sk_i, \mathsf{par})$ *is a deterministic algorithm which takes a second level ciphertext* C_i *for* i*, a user* i*'s secret key* sk_i*, and* par*, and returns a plaintext* m *or a distinguished message '*`invalid`*'.*

To lighten notations, we will sometimes omit to explicitly write public parameters par, taken as input by all but one of the above algorithms.

3.1.2 Security Definitions – Confidentiality

Next, we describe the definition of the replayable chosen ciphertext security of unidirectional PRE schemes by Libert and Vergnaud [4]. In unidirectional single-hop PRE schemes, there exist two types of ciphertexts, first level and second level ciphertexts. Therefore, it is necessary to prove the confidentiality of each types of ciphertexts. First, we describe the security definition of second level ciphertexts.

Definition 5 (Second level RCCA security)**.** *A unidirectional single-hop proxy re-encryption scheme is second level secure against replayable chosen-ciphertext attack (RCCA) (or second level*

RCCA secure for short) if

$$|\Pr[\mathsf{par} \leftarrow \mathsf{Global\text{-}setup}(\lambda); (pk_*, sk_*) \leftarrow \mathsf{Keygen}(\lambda); \{(pk_h, sk_h) \leftarrow \mathsf{Keygen}(\lambda)\};$$
$$\{(pk_c, sk_c) \leftarrow \mathsf{Keygen}(\lambda)\}; \{R_{*h} \leftarrow \mathsf{Rekeygen}(sk_*, pk_h)\}; \{R_{h*} \leftarrow \mathsf{Rekeygen}(sk_h, pk_*)\};$$
$$\{R_{hc} \leftarrow \mathsf{Rekeygen}(sk_h, pk_c)\}; \{R_{hh'} \leftarrow \mathsf{Rekeygen}(sk_h, pk_{h'})\};$$
$$(m_0, m_1, St) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{reenc}}, \mathcal{O}_{\text{1-dec}}}(pk_*, \{pk_h\}, \{(pk_c, sk_c)\}, \{R_{*h}\}, \{R_{h*}\}, \{R_{hc}\}, \{R_{hh'}\});$$
$$d^* \xleftarrow{R} \{0,1\}; C^* \leftarrow \mathsf{Enc}_2(m_{d^*}, pk_*); d' \xleftarrow{R} \mathcal{A}^{\mathcal{O}_{\text{reenc}}, \mathcal{O}_{\text{1-dec}}}(C^*, St) : d' = d^*] - 1/2|$$

is negligible for any polynomial time algorithm $\mathcal{A}$. Above, St is the state information maintained by $\mathcal{A}$. (pk^*, sk^*) is the target user's key pair. For honest users other than target user, keys are subscripted by h or h'. We subscript corrupt keys by c or c'. Oracles $\mathcal{O}_{\text{reenc}}$ and $\mathcal{O}_{\text{1-dec}}$ proceed as follows:

- *Re-encryption oracle $\mathcal{O}_{\text{reenc}}$: on input (pk_i, pk_j, C) where C is a second level ciphertext and pk_i, pk_j were produced by* Keygen*, this oracle responds with* `invalid` *if C is not properly shaped (ill-formed) with respect to pk_i. It returns a special symbol $\perp$ if j is a corrupted user and $(pk_i, C) = (pk_*, C^*)$. Otherwise, the re-encrypted first level ciphertext $C_j = \mathsf{Reenc}(\mathsf{Rekeygen}(sk_i, pk_j), C)$ is returned to $\mathcal{A}$.*
- *First level decryption oracle $\mathcal{O}_{\text{1-dec}}$: given a pair (pk, C), where C is a first level ciphertext and pk was produced by* Keygen*, this oracle returns* `invalid` *if C is ill-formed with respect to pk. If the query occurs in the guess stage, it outputs $\perp$ if (pk, C) is a* Derivative *of (pk_*, C^*). Otherwise, the plaintext $m = \mathsf{Dec}_1(sk, C)$ is returned to $\mathcal{A}$.* Derivative*s of (pk_*, C^*) are defined as follows: If C is a first level ciphertext and pk is an honest user's key, (pk, C) is a* Derivative *of (pk_*, C^*) if $\mathsf{Dec}_1(sk, C) \in \{m_0, m_1\}$.*

Next, we describe the definition of RCCA security of first level ciphertexts.

Definition 6 (First level RCCA security)**.** *A unidirectional single-hop proxy re-encryption scheme is first level secure against replayable chosen-ciphertext attack (RCCA) (or first level RCCA secure for short) if*

$$|\Pr[\mathsf{par} \leftarrow \mathsf{Global\text{-}setup}(\lambda); (pk_*, sk_*) \leftarrow \mathsf{Keygen}(\lambda); \{(pk_h, sk_h) \leftarrow \mathsf{Keygen}(\lambda)\};$$
$$\{(pk_c, sk_c) \leftarrow \mathsf{Keygen}(\lambda)\}; \{R_{*c} \leftarrow \mathsf{Rekeygen}(sk_*, pk_c)\}; \{R_{*h} \leftarrow \mathsf{Rekeygen}(sk_*, pk_h)\};$$
$$\{R_{h*} \leftarrow \mathsf{Rekeygen}(sk_h, pk_*)\}; \{R_{hc} \leftarrow \mathsf{Rekeygen}(sk_h, pk_c)\}; \{R_{hh'} \leftarrow \mathsf{Rekeygen}(sk_h, pk_{h'})\};$$
$$(m_0, m_1, St) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{1-dec}}}(pk_*, \{pk_h\}, \{(pk_c, sk_c)\}, \{R_{*c}\}, \{R_{*h}\}, \{R_{h*}\}, \{R_{hc}\}, \{R_{hh'}\});$$
$$d^* \xleftarrow{R} \{0,1\}; C^* \leftarrow \mathsf{Enc}_1(m_{d^*}, pk_*); d' \xleftarrow{R} \mathcal{A}^{\mathcal{O}_{\text{1-dec}}}(C^*, St) : d' = d^*] - 1/2|$$

is negligible for any polynomial time algorithm $\mathcal{A}$. Oracle $\mathcal{O}_{\text{1-dec}}$ is the same as that in Definition 5 except that the definition of Derivative*s: If C is a first level ciphertext and $pk = pk_*$, (pk, C) is a* Derivative *of (pk_*, C^*) if $\mathsf{Dec}_1(sk, C) \in \{m_0, m_1\}$.*

Above, all re-encryption keys are available to the adversary. Therefore, the re-encryption oracle becomes useless and is not given to the adversary.

3.1.3 Security Definitions – Unforgeability of Re-Encryption Keys

Finally, we review the definition of (strong) unforgeability of re-encryption keys against collusion attacks proposed in [1].

In addition to the above basic security property, it is also important to consider the security where such proxies are corrupted. Recently, as cloud computing emerges, PRE gains much more attention as one of the key security components to provide secure cloud services. The security against corrupted proxies is especially important in such applications since the proxies may be out of control of honest users and the proxies are more likely to be attacked than those in on-premise systems.

As an example of such kind of security notions, Ateniese et al. proposed the informal definition of "non-transferability [15]". Consider the situation that a (target) user i^* delegated her decryption rights to users $\{c\}$ (i.e. a proxy has re-encryption keys $\{R_{*c}\}$). Intuitively, the scheme is non-transferable when the only way for users $\{c\}$ and the proxy (colluding parties) to transfer the (offline) decryption capabilities of i^* to some user j (malicious user) is to expose the secret key of anyone in $\{c\}$. Hayashi et al. [1] proposed the formal definition of non-transferability. See [1] for details.

Hayashi et al. also proposed the definition of "unforgeability of re-encryption keys against collusion attacks (UFReKey-CA)."

Definition 7 (Unforgeability of Re-Encryption Keys against Collusion Attack, UFReKey-CA [1])**.** *A unidirectional single-hop proxy re-encryption scheme meets* the unforgeability of re-encryption keys against collusion attack *if there exists a polynomial time algorithm $\mathcal{P}$ such that*

$$\begin{aligned}
&\Pr[\mathsf{par} \leftarrow \mathsf{Global\text{-}setup}(\lambda);\\
&(pk_*, sk_*) \leftarrow \mathsf{Keygen}(\lambda);\ (pk_h, sk_h) \leftarrow \mathsf{Keygen}(\lambda);\ \{(pk_{c_i}, sk_{c_i}) \leftarrow \mathsf{Keygen}(\lambda)\};\\
&(pk_j, sk_j) \leftarrow \mathsf{Keygen}(\lambda);\ \{R_{*c_i} \leftarrow \mathsf{Rekeygen}(sk_*, pk_{c_i})\};\ \{R_{hc_i} \leftarrow \mathsf{Rekeygen}(sk_h, pk_{c_i})\};\\
&m \xleftarrow{R} \mathcal{M};\ C^* \leftarrow \mathsf{Enc_2}(m, pk_*);\ \{m_i \xleftarrow{R} \mathcal{M}\};\ \{C_i \leftarrow \mathsf{Enc_2}(m_i, pk_{c_i})\};\\
&\{m'_i \xleftarrow{R} \mathcal{M}\};\ \{C'_i \leftarrow \mathsf{Enc_1}(m'_i, pk_{c_i})\};\ \{m''_i \xleftarrow{R} \mathcal{M}\};\ \{C''_i \leftarrow \mathsf{Reenc}(R_{hc_i}, \mathsf{Enc_2}(m''_i, pk_h))\};\\
&X \leftarrow \mathcal{C}(pk_*, \{(pk_{c_i}, sk_{c_i})\}, \{R_{*c_i}\});\ R^\dagger_{*j} \leftarrow \mathcal{J}(X, (pk_j, sk_j));\\
&m_{\mathcal{P}} \leftarrow \mathcal{P}(X, (pk_j, sk_j), \{C_i\}, \{C'_i\}, \{C''_i\})\\
&: m \neq \mathsf{Dec_1}(\mathsf{Reenc}(R^\dagger_{*j}, C^*), sk_j) \vee m_{\mathcal{P}} \in \{m_i\} \cup \{m'_i\} \cup \{m''_i\}]
\end{aligned}$$

is overwhelming for any polynomial time algorithm $\mathcal{C}$, $\mathcal{J}$.

Intuitively, the scheme meets UFReKey-CA when the only way for the users $\{c_i\}$ and the proxy to expose information which enable the user j to generate a re-encryption key R_{*j} is to expose the secret key of anyone in $\{c_i\}$. Therefore, it is easy to see that UFReKey-CA is a relaxed notion (necessary condition) of the non-transferability. Thus, to construct the scheme which meets non-transferability, it is necessary that the scheme meets UFReKey-CA.

Further, Hayashi et al. proposed the definition of "strong unforgeability of re-encryption keys against collusion attacks (sUFReKey-CA)." We review the definition which is used in this paper.

Definition 8 (Strong Unforgeability of Re-Encryption Keys against Collusion Attack, sUFReKey-CA [1])**.** *A unidirectional single-hop proxy re-encryption scheme meets* the strong unforgeability of re-encryption keys against collusion attack *if the following probability is negligible for any polynomial time algorithm $\mathcal{A}$.*

$$\begin{aligned}
&\Pr[\mathsf{par} \leftarrow \mathsf{Global\text{-}setup}(\lambda); (pk_*, sk_*) \leftarrow \mathsf{Keygen}(\lambda); \{(pk_c, sk_c) \leftarrow \mathsf{Keygen}(\lambda)\};\\
&(pk_j, sk_j) \leftarrow \mathsf{Keygen}(\lambda); \{R_{*c} \leftarrow \mathsf{Rekeygen}(sk_*, pk_c)\}; m \xleftarrow{R} \mathcal{M};\ C^* \leftarrow \mathsf{Enc_2}(m, pk_*);\\
&R^\dagger_{*j} \leftarrow \mathcal{A}(pk_*, \{(pk_c, sk_c)\}, (pk_j, sk_j), \{R_{*c}\}) :\ m = \mathsf{Dec_1}(\mathsf{Reenc}(R^\dagger_{*j}, C^*), sk_j)]
\end{aligned}$$

The scheme meets sUFReKey-CA when the users $\{c\}$, the proxy who has $\{R_{*c}\}$, and the user j cannot generate (forge) a re-encryption key R_{*j}. Since sUFReKey-CA implies UFReKey-CA, and the security definition (model) of sUFReKey-CA is more simple than that of UFReKey-CA, the definition of sUFReKey-CA is useful to prove UFReKey-CA property.

3.2 Our Proposed Scheme

Our scheme is based on the scheme in [1]. To prevent re-encryption key forgery, we change the form of the re-encryption key. We mask every secret key of the delegatee with a randomness or a system secret parameter. Due to this change, we also change the verification equation for first level ciphertexts. In our scheme, the secret key is required to verify the first level ciphertext. Interestingly, in the proof of security, we can simulate the evaluation of whether the equation holds or not without knowing the secret key. Further, we prove that our scheme meets UFReKey-CA under the assumption that the 2-DHI problem is hard. Namely, in the proof of our scheme, we can generate (simulate) the re-encryption key directly from the instance of 2-DHI problem (not required some additional inputs).

3.2.1 Description

$\mathsf{Global\text{-}setup}(\lambda)$: given a security parameter λ, first choose bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order $p > 2^\lambda$, and generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. Next, compute $g_1 = g^\alpha, g_2 = g^\beta, u = g^\delta, v = g^\omega$, $h_1 = h^\alpha, h_2 = h^\beta, \hat{u} = h^\delta, \hat{v} = h^\omega$ where $\alpha, \beta, \delta, \omega \overset{R}{\leftarrow} \mathbb{Z}_p^*$. Finally, choose a one-time signature scheme $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The public parameters are $\mathsf{par} = \{p, (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T), g, g_1, g_2, u, v, h, h_1, h_2, \hat{u}, \hat{v}, \mathsf{Sig}\}$. The message space $\mathcal{M}$ is equal to $\mathbb{G}_T$.

$\mathsf{Keygen}(\lambda, \mathsf{par})$: user i chooses $x_i, y_i, z_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$. The secret key is $sk_i = (x_i, y_i, z_i)$. The public key is $pk_i = (X_i, Y_{1i}, Z_i, Z_{1i}, \hat{X}_i, \hat{Y}_{1i}, \hat{Y}_{2i})$ where $X_i = g^{x_i}, Y_{1i} = {g_1}^{y_i}, Z_i = g^{z_i}, Z_{1i} = {g_1}^{z_i}, \hat{X}_i = h^{x_i}, \hat{Y}_{1i} = {h_1}^{y_i}, \hat{Y}_{2i} = {h_2}^{y_i}$.

$\mathsf{Enc}_1(m, pk_j, \mathsf{par})$: to encrypt a message $m \in \mathbb{G}_T$ under the public key pk_j at the first level, the sender proceeds as follows:

1. Select a one-time signature key pair $(svk, ssk) \leftarrow \mathcal{G}(\lambda)$ and set $C_1 = svk$.
2. Pick $r, s, t, k \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and compute $C'_{2X} = {Y_{1j}}^s, C''_{2X} = {Y_{1j}}^{rs}, C'_{2Y} = {X_j}^t, C''_{2Y} = {X_j}^{rt}, C'_{2Z} = {Y_{1j}}^k, C''_{2Z} = {Y_{1j}}^{rk}, C'_{2Z1} = {X_j}^k, C''_{2Z1} = {X_j}^{rk}, C_3 = m \cdot e(g_1 g_2, h)^r, C_4 = (u^{svk} \cdot v)^r, \hat{C}_4 = (\hat{u}^{svk} \cdot \hat{v})^r, C_{5X} = {h_2}^{\frac{1}{s}} = h^{\frac{\beta}{s}}, C_{5Y} = h^{\frac{1}{t}}, C_{5Z} = (h \cdot h_2)^{\frac{1}{k}} = h^{\frac{1+\beta}{k}}$.
3. Generate a one-time signature $\sigma \leftarrow \mathcal{S}(ssk, (C_3, C_4))$ on (C_3, C_4).

The (first level) ciphertext is $C_j = (C_1, C'_{2X}, C''_{2X}, C'_{2Y}, C''_{2Y}, C'_{2Z}, C''_{2Z}, C'_{2Z1}, C''_{2Z1}, C_3, C_4, \hat{C}_4, C_{5X}, C_{5Y}, C_{5Z}, \sigma)$.

$\mathsf{Enc}_2(m, pk_i, \mathsf{par})$: to encrypt a message $m \in \mathbb{G}_T$ under the public key pk_i at the second level, the sender proceeds as follows:

1. Select a one-time signature key pair $(svk, ssk) \leftarrow \mathcal{G}(\lambda)$ and set $C_1 = svk$.
2. Pick $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and compute $C_{2X} = {X_i}^r, C_{2Y} = {Y_{1i}}^r, C_{2Z} = {Z_i}^r, C_{2Z1} = {Z_{1i}}^r, C_3 = m \cdot e(g_1 g_2, h)^r, C_4 = (u^{svk} \cdot v)^r, \hat{C}_4 = (\hat{u}^{svk} \cdot \hat{v})^r$.

3. Generate a one-time signature $\sigma \leftarrow \mathcal{S}(ssk, (C_3, C_4))$ on (C_3, C_4).

The (second level) ciphertext is $C_i = (C_1, C_{2X}, C_{2Y}, C_{2Z}, C_{2Z1}, C_3, C_4, \hat{C}_4, \sigma)$.

$\mathsf{Rekeygen}(sk_i, pk_j, \mathsf{par})$: given user i's secret key sk_i and user j's public key pk_j, choose $\ell, n \xleftarrow{R} \mathbb{Z}_p^*$ and generate a re-encryption key $R_{ij} = (R_{ij1}, R_{ij2}, R_{ij3})$ as $R_{ij1} = (\hat{X}_j^{\ell} \cdot \hat{Y}_{1j}^{\ell-n-1})^{\frac{1}{x_i}} = h^{\frac{\ell x_j + \alpha(\ell-n-1)y_j}{x_i}}$, $R_{ij2} = (\hat{X}_j^{n} \cdot \hat{Y}_{2j})^{\frac{1}{y_i}} = h^{\frac{n x_j + \beta y_j}{y_i}}$, and $R_{ij3} = (\hat{X}_j^{\ell} \cdot \hat{Y}_{2j})^{\frac{1}{z_i}} = h^{\frac{\ell x_j + \beta y_j}{z_i}}$.

$\mathsf{Reenc}(R_{ij}, C_i, \mathsf{par})$: on input of the re-encryption key R_{ij} and a second level ciphertext C_i, check the validity of the ciphertext by testing:

$$\begin{aligned} e(C_{2X}, \hat{u}^{C_1} \cdot \hat{v}) = e(X_i, \hat{C}_4),\ e(C_{2Y}, \hat{u}^{C_1} \cdot \hat{v}) = e(Y_{1i}, \hat{C}_4),\ e(C_{2Z}, \hat{u}^{C_1} \cdot \hat{v}) = e(Z_i, \hat{C}_4), \\ e(C_{2Z1}, \hat{u}^{C_1} \cdot \hat{v}) = e(Z_{1i}, \hat{C}_4),\ e(g, \hat{C}_4) = e(C_4, h),\ \mathcal{V}(C_1, \sigma, (C_3, C_4)) = 1. \end{aligned} \tag{1}$$

If the relations (1) hold (well-formed), C_i is re-encrypted by choosing $s, t, k \xleftarrow{R} \mathbb{Z}_p^*$ and computing $C'_{2X} = {X_i}^s, C''_{2X} = {C_{2X}}^s = {X_i}^{rs}, C'_{2Y} = {Y_{1i}}^t, C''_{2Y} = {C_{2Y}}^t = {Y_{1i}}^{rt}, C'_{2Z} = {Z_i}^k, C''_{2Z} = {C_{2Z}}^k = {Z_i}^{rk}, C'_{2Z1} = {Z_{1i}}^k, C''_{2Z1} = {C_{2Z1}}^k = {Z_{1i}}^{rk}, C_{5X} = {R_{ij1}}^{\frac{1}{s}}, C_{5Y} = {R_{ij2}}^{\frac{1}{t}}, C_{5Z} = {R_{ij3}}^{\frac{1}{k}}$, and a re-encrypted ciphertext $C_j = (C_1, C'_{2X}, C''_{2X}, C'_{2Y}, C''_{2Y}, C'_{2Z}, C''_{2Z}, C'_{2Z1}, C''_{2Z1}, C_3, C_4, \hat{C}_4, C_{5X}, C_{5Y}, C_{5Z}, \sigma)$ is returned. Otherwise, '`invalid`' is returned.

$\mathsf{Dec}_1(C_j, sk_j)$: the validity of the first level ciphertext C_j is checked by testing:

$$\begin{aligned} e(C''_{2X}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2X}, \hat{C}_4),\ e(C''_{2Y}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2Y}, \hat{C}_4), \\ e(C''_{2Z}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2Z}, \hat{C}_4),\ e(C''_{2Z1}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2Z1}, \hat{C}_4),\ e(g, \hat{C}_4) = e(C_4, h), \\ \mathcal{V}(C_1, \sigma, (C_3, C_4)) = 1,\ \left(\frac{e(C'_{2Z}, C_{5Z})}{e(C'_{2X}, C_{5X})}\right)^{\frac{1}{y_j}} \left(\frac{e(C'_{2Z1}, C_{5Z})}{e(C'_{2Y}, C_{5Y})}\right)^{\frac{1}{x_j}} = e(g_1 g_2, h). \end{aligned} \tag{2}$$

If the relations (2) hold (well-formed), the plaintext is returned as

$$m = C_3 \Big/ \left\{ \left(\frac{e(C''_{2Z}, C_{5Z})}{e(C''_{2X}, C_{5X})}\right)^{\frac{1}{y_j}} \left(\frac{e(C''_{2Z1}, C_{5Z})}{e(C''_{2Y}, C_{5Y})}\right)^{\frac{1}{x_j}} \right\}.$$

Otherwise (ill-formed), the algorithm outputs '`invalid`.'

$\mathsf{Dec}_2(C_i, sk_i)$: if the second level ciphertext C_i satisfies the relations (1), the plaintext $m = C_3 \,/\, e(C_{2X}, h_1 h_2)^{\frac{1}{x_i}}$ is returned. Otherwise, '`invalid`' is returned.

We can check the correctness property of the above scheme as follows. Note that Equations (1) ensure $(r =) \log_{X_i} C_{2X} = \log_{Y_{1i}} C_{2Y} = \log_{Z_i} C_{2Z} = \log_{Z_{1i}} C_{2Z1} = \log_{u^{C_1} v} C_4 = \log_{\hat{u}^{C_1} \hat{v}} \hat{C}_4$, and Equations (2) ensure $(r =) \log_{C'_{2X}} C''_{2X} = \log_{C'_{2Y}} C''_{2Y} = \log_{C'_{2Z}} C''_{2Z} = \log_{C_{2Z1}} C_{2Z1} = \log_{u^{C_1} v} C_4 = \log_{\hat{u}^{C_1} \hat{v}} \hat{C}_4$.

- $\mathsf{Dec}_2(\mathsf{Enc}_2(m, pk_i, \mathsf{par}), sk_i, \mathsf{par}) = m \cdot e(g_1 g_2, h)^r \,/\, e(g^{x_i r}, h_1 h_2)^{\frac{1}{x_i}}$
  $= m \cdot e(g, h)^{(\alpha+\beta)r} / e(g, h)^{(\alpha+\beta)r} = m.$

- $\mathsf{Dec}_1(\mathsf{Enc}_1(m, pk_j, \mathsf{par}), sk_j, \mathsf{par})$
$= C_3 \Big/ \left\{ \left(\frac{e(g^{\alpha y_j r k}, h^{(1+\beta)/k})}{e(g^{\alpha y_j rs}, h^{\beta/s})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{x_j rk}, h^{(1+\beta)/k})}{e(g^{x_j rt}, h^{1/t})} \right)^{\frac{1}{x_j}} \right\} = C_3 \Big/ \left(e(g^{\alpha y_j r}, h)^{\frac{1}{y_j}} \cdot e(g^{x_j r}, h^{\beta})^{\frac{1}{x_j}} \right)$
$= m \cdot e(g_1 g_2, h)^r / (e(g,h)^{\alpha r} \cdot e(g,h)^{\beta r}) = m.$

- $\mathsf{Dec}_1(\mathsf{Reenc}(\mathsf{Rekeygen}(sk_i, pk_j, \mathsf{par}), \mathsf{Enc}_2(m, pk_i, \mathsf{par}), \mathsf{par}), sk_j, \mathsf{par})$
$= C_3 \Big/ \left\{ \left(\frac{e(g^{z_i r k}, h^{(\ell x_j + \beta y_j)/z_i k})}{e(g^{x_i rs}, h^{(\ell x_j + \alpha(\ell - n - 1) y_j)/x_i s})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha z_i r k}, h^{(\ell x_j + \beta y_j)/z_i k})}{e(g^{\alpha y_i rt}, h^{(nx_j + \beta y_j)/y_i t})} \right)^{\frac{1}{x_j}} \right\}$
$= C_3 \Big/ \left\{ \left(\frac{e(g^r, h^{\beta y_j})}{e(g^r, h^{\alpha(\ell - n - 1) y_j})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha r}, h^{\ell x_j})}{e(g^{\alpha r}, h^{n x_j})} \right)^{\frac{1}{x_j}} \right\} = C_3 \Big/ \left\{ \frac{e(g^r, h^{\beta})}{e(g^r, h^{\alpha(\ell - n - 1)})} \cdot \frac{e(g^{\alpha r}, h^{\ell})}{e(g^{\alpha r}, h^{n})} \right\}$
$= m \cdot e(g_1 g_2, h)^r / (e(g,h)^{\beta r} \cdot e(g,h)^{\alpha r}) = \ m.$

3.2.2 Security

Now, we show that our scheme meets RCCA security and sUFReKey-CA.

Theorem 1. *Our proposed scheme with the strong one-time signature satisfies second level RCCA security if the 3-wDBDHI problem is hard.*

Proof. We use the following lemma. The symmetric pairing version of this lemma is proved in [4]. The proof of this lemma is similar to that in [4]

Lemma 1. *The 3-wDBDHI problem is equivalent to the the modified 3-wDBDHI problem which is to distinguish*

- $(g, g^{1/a}, g^a, g^{a^2}, g^b, h, h^{1/a}, h^a, h^{a^2}, h^b, e(g,h)^{b/a^2})$ *and*
- $(g, g^{1/a}, g^a, g^{a^2}, g^b, h, h^{1/a}, h^a, h^{a^2}, \ h^b, e(g,h)^z)$

where $a, b, z \xleftarrow{R} \mathbb{Z}_p^*$.

Proof of Lemma 1. Given an instance of modified 3-wDBDHI problem, $(g, g^{1/a}, g^a, g^{a^2}, g^b, h, h^{1/a}, h^a, h^{a^2}, h^b, T)$, we can construct an instance of 3-wDBDHI problem by setting $(y = g^{1/a}, y^A = g, y^{A^2} = g^a, y^{A^3} = g^{a^2}, y^B = g^b, z = h^{1/a}, z^A = h, z^{A^2} = h^a, z^{A^3} = h^{a^2}, z^B = h^b, T)$ which implicitly defines $A = a$ and $B = ab$. Note that $e(y,z)^{B/A} = e(g^{1/a}, h^{1/a})^{ab/a} = e(g,h)^{b/a^2}$. The converse implication is easily considered. □

Now we prove Theorem 1. We prove that our proposed scheme is second level RCCA secure under the assumption that the above (modified 3-wDBDHI) problem is hard. We build an algorithm $\mathcal{B}$ which is, given $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, h, \hat{A}_{-1} = h^{1/a}, \hat{A}_1 = h^a, \hat{A}_2 = h^{a^2}, \hat{B} = h^b, T)$, solving the modified 3-wDBDHI problem using second level RCCA adversary $\mathcal{A}$.

In the following, we call HU the set of honest users, including the target user i^* that is assigned the target public key pk^*, and CU the set of corrupt users. The algorithm $\mathcal{B}$ simulates $\mathcal{A}$'s input and oracles as follows.

Public parameters: $\mathcal{B}$ chooses $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p^*$ and computes $g_1 = g^\alpha, g_2 = g^\beta, h_1 = h^\alpha, h_2 = h^\beta$. $\mathcal{B}$ chooses the key pair $(svk^*, ssk^*) \xleftarrow{R} \mathcal{G}(1^\lambda)$ of the one-time signature scheme. The generator $u, v, \hat{v}, \hat{v}$ are set as $u = A_1^{\alpha_1}$, $v = A_1^{-\alpha_1 svk^*} A_2^{\alpha_2}$, $\hat{u} = \hat{A}_1^{\alpha_1}$, $\hat{v} = \hat{A}_1^{-\alpha_1 svk^*} \hat{A}_2^{\alpha_2}$, where $\alpha_1, \alpha_2 \xleftarrow{R} \mathbb{Z}_p^*$.

Public and secret keys: For the corrupt user $i \in CU$, $\mathcal{B}$ computes $sk_i = (x_i, y_i, z_i)$ and pk_i by using the key-generation algorithm. For the target user i^* and the honest users $h \in HU \backslash \{i^*\}$, $\mathcal{B}$ chooses $x_*, y_*, z_*, x_h, y_h, z_h \xleftarrow{R} \mathbb{Z}_p^*$ and computes

- $pk_* = (A_2{}^{x_*}, A_2{}^{\alpha y_*}, A_2{}^{z_*}, A_2{}^{\alpha z_*}, \hat{A}_2^{x_*}, \hat{A}_2^{\alpha y_*}, \hat{A}_2^{\beta y_*}) = (g^{a^2 x_*}, g_1{}^{a^2 y_*}, g^{a^2 z_*}, g_1{}^{a^2 z_*}, h^{a^2 x_*}, h_1{}^{a^2 y_*}, h_2{}^{a^2 y_*})$,
- $pk_h = (A_1{}^{x_h}, A_1{}^{\alpha y_h}, A_1{}^{z_h}, A_1{}^{\alpha z_h}, \hat{A}_1^{x_h}, \hat{A}_1^{\alpha y_h}, \hat{A}_1^{\beta y_h}) = (g^{a x_h}, g_1{}^{a y_h}, g^{a z_h}, g_1{}^{a z_h}, h^{a x_h}, h_1{}^{a y_h}, h_2{}^{a y_h})$.

Here, the secret keys for i^* and h are $(a^2 x_*, a^2 y_*, a^2 z_*)$ and $(a x_j, a y_j, a z_j)$, respectively. Note that $\mathcal{B}$ does not have to compute the secret keys of honest users i^* and h to compute the corresponding public keys.

Re-encryption keys: The re-encryption keys $R_{c*}, R_{ch}, R_{cc'}$ can be computed by following Rekeygen. The other re-encryption keys can be computed as follows where $\ell, n \xleftarrow{R} \mathbb{Z}_p^*$.

- $R_{h*} = (\hat{A}_1^{\frac{\ell x_* + \alpha(\ell - n - 1) y_*}{x_h}}, \hat{A}_1^{\frac{n x_* + \beta y_*}{y_h}}, \hat{A}_1^{\frac{\ell x_* + \beta y_*}{z_h}}) = (h^{\frac{a^2 \ell x_* + \alpha(\ell - n - 1) a^2 y_*}{a x_h}}, h^{\frac{n a^2 x_* + \beta a^2 y_*}{a y_h}}, h^{\frac{\ell a^2 x_* + \beta a^2 y_*}{a z_h}})$,
- $R_{*h} = (\hat{A}_{-1}^{\frac{\ell x_h + \alpha(\ell - n - 1) y_h}{x_*}}, \hat{A}_{-1}^{\frac{n x_h + \beta y_h}{y_*}}, \hat{A}_{-1}^{\frac{\ell x_h + \beta y_h}{z_*}}) = (h^{\frac{\ell a x_h + \alpha(\ell - n - 1) a y_h}{a^2 x_*}}, h^{\frac{n a x_h + \beta a y_h}{a^2 y_*}}, h^{\frac{\ell a x_h + \beta a y_h}{a^2 z_*}})$,
- $R_{hc} = (\hat{A}_{-1}^{\frac{\ell x_c + \alpha(\ell - n - 1) y_c}{x_h}}, \hat{A}_{-1}^{\frac{n x_c + \beta y_c}{y_h}}, \hat{A}_{-1}^{\frac{\ell x_c + \beta y_c}{z_h}}) = (h^{\frac{\ell a x_c + \alpha(\ell - n - 1) a y_c}{a x_h}}, h^{\frac{n a x_c + \beta a y_c}{a y_h}}, h^{\frac{\ell a x_c + \beta a y_c}{a z_h}})$,
- $R_{hh'} = (h^{\frac{\ell x_{h'} + \alpha(\ell - n - 1) y_{h'}}{x_h}}, h^{\frac{n x_{h'} + \beta y_{h'}}{y_h}}, h^{\frac{\ell x_{h'} + \beta y_{h'}}{z_h}}) = (h^{\frac{\ell a x_{h'} + \alpha(\ell - n - 1) a y_{h'}}{a x_h}}, h^{\frac{n a x_{h'} + \beta a y_{h'}}{a y_h}}, h^{\frac{\ell a x_{h'} + \beta a y_{h'}}{a z_h}})$

Re-encryption oracle: For the re-encryption query (pk_i, pk_j, C_i), $\mathcal{B}$ checks the validity of C_i by using the equations (1). Note that equations (1) are publicly verified. If C_i is ill-formed, $\mathcal{B}$ outputs `invalid`. Otherwise, if $i \neq i^*$ or $j \notin CU$, $\mathcal{B}$ uses the re-encryption key and responds the query. If $i = i^*$ and $j \in CU$, $C_1 \neq svk^*$ holds with overwhelming probability (because of the strong unforgeability of the one-time signature). Then, the re-encrypted ciphertext C_j can be computed as follows where $s', t', k' \xleftarrow{R} \mathbb{Z}_p^*$.

$$C_j = (C_1, A_1{}^{s'}, (A_1{}^r)^{s'}, A_1{}^{\alpha t'}, (A_1{}^r)^{\alpha t'}, A_1{}^{k'}, (A_1{}^r)^{k'}, A_1{}^{\alpha k'}, (A_1{}^r)^{\alpha k'}, C_3, C_4, \hat{C}_4, \hat{A}_{-1}^{\frac{\ell x_j + \alpha(\ell - n - 1) y_j}{s'}}, \hat{A}_{-1}^{\frac{n x_j + \beta y_j}{t'}}, \hat{A}_{-1}^{\frac{\ell x_j + \beta y_j}{k'}}, \sigma).$$

Note that $A_1{}^r$ can be computed as $A_1{}^r = (C_4 / C_{2X}^{\alpha_2 / x_*})^{\frac{1}{\alpha_1 (C_1 - svk^*)}}$ since $C_4 = (A_1^{\alpha_1 C_1} A_1^{-\alpha_1 svk^*} A_2^{\alpha_2})^r = A_1^{\alpha_1 (C_1 - svk^*) r} A_2^{\alpha_2 r}$ and $C_{2X} = X_*{}^r = A_2^{x_* r}$. Since we have

$$C_j = (C_1, (g^{a^2 x_*})^{s'/a x_*}, ((g^{a^2 x_*})^r)^{s'/a x_*}, (g_1^{a^2 y_*})^{t'/a y_*}, ((g_1^{a^2 y_*})^r)^{t'/a y_*}, (g^{a^2 z_*})^{k'/a z_*}, ((g^{a^2 z_*})^r)^{k'/a z_*}, (g_1^{a^2 z_*})^{k'/a z_*}, ((g_1^{a^2 z_*})^r)^{k'/a z_*}, C_3, C_4, \hat{C}_4, (h^{\frac{\ell x_j + \alpha(\ell - n - 1) y_j}{a^2 x_*}})^{\frac{1}{s'/a x_*}}, (h^{\frac{n x_j + \beta y_j}{a^2 y_*}})^{\frac{1}{t'/a y_*}}, (h^{\frac{\ell x_j + \beta y_j}{a^2 z_*}})^{\frac{1}{k'/a z_*}}, \sigma),$$

this is a valid ciphertext with the randomness $s = s'/a x_*$, $t = t'/a y_*$, $k = k'/a z_*$.

First level decryption oracle: For the first level decryption query (pk_j, C_j), $\mathcal{B}$ checks the validity of C_j by using the equations (2). Here, the secret key is needed to verify the seventh (final) equation in equations (2), but $\mathcal{B}$ does not have the secret key of the honest users. Thus, $\mathcal{B}$ checks the following equation instead of the seventh equation in equations (2).

- If $j \in HU\backslash\{i^*\}$, the secret key of j is equal to (ax_h, ay_h, az_h) where $\mathcal{B}$ knows (x_h, y_h, z_h). Thus $\mathcal{B}$ checks $\left(\frac{e(C'_{2Z},C_{5Z})}{e(C'_{2X},C_{5X})}\right)^{\frac{1}{y_h}} \left(\frac{e(C'_{2Z1},C_{5Z})}{e(C'_{2Y},C_{5Y})}\right)^{\frac{1}{x_h}} = e(g_1g_2, \hat{A}_1)$.
- If $j = i^*$, the secret key of j is equal to (a^2x_*, a^2y_*, a^2z_*) where $\mathcal{B}$ knows (x_*, y_*, z_*). Thus $\mathcal{B}$ checks $\left(\frac{e(C'_{2Z},C_{5Z})}{e(C'_{2X},C_{5X})}\right)^{\frac{1}{y_*}} \left(\frac{e(C'_{2Z1},C_{5Z})}{e(C'_{2Y},C_{5Y})}\right)^{\frac{1}{x_*}} = e(g_1g_2, \hat{A}_2)$.

If C_j is ill-formed, $\mathcal{B}$ outputs `invalid`. When C_j is well-formed (and $j \in HU$), if $C_1 = svk^*$ and $(C_3, C_4, \sigma) = (C_3^*, C_4^*, \sigma^*)$, $\mathcal{B}$ returns $\perp$ since C_j is a Derivative of the challenge ciphertext. In the other case, $C_1 \neq svk^*$ holds with overwhelming probability (because of the strong unforgeability of the one-time signature). Then, we consider the following two cases.

- If $j \in HU\backslash\{i^*\}$, since C_j is a valid ciphertext and user j's secret key is equal to (ax_h, ay_h, az_h), we have
  - $e(C_4, \hat{A}_{-1}) = e((A_1^{\alpha_1 C_1} A_1^{-\alpha_1 svk^*} A_2^{\alpha_2})^r, \hat{A}_{-1}) = e(g,h)^{(C_1 - svk^*)\alpha_1 r + \alpha_2 ar}$
  - $U = \left(\frac{e(C''_{2Z},C_{5Z})}{e(C''_{2X},C_{5X})}\right)^{\frac{1}{y_h}} \left(\frac{e(C''_{2Z1},C_{5Z})}{e(C''_{2Y},C_{5Y})}\right)^{\frac{1}{x_h}} = e(g_1g_2, h)^{ar}$.

  Thus, $\mathcal{B}$ computes $X = \{e(C_4, \hat{A}_{-1})^{\alpha+\beta}/U^{\alpha_2}\}^{\frac{1}{(C_1 - svk^*)\alpha_1}} (= e(g_1g_2, h)^r)$, and $m = C_3/X$.
- If $j = i^*$, since C_j is a valid ciphertext and user j's secret key is equal to (a^2x_*, a^2y_*, a^2z_*), we have
  - $e(C_4, \hat{A}_{-1}) = e((A_1^{\alpha_1 C_1} A_1^{-\alpha_1 svk^*} A_2^{\alpha_2})^r, \hat{A}_{-1}) = e(g,h)^{(C_1 - svk^*)\alpha_1 r + \alpha_2 ar}$,
  - $e(C_4, h) = e(C_4, \hat{A}_{-1})^a = e(g,h)^{(C_1 - svk^*)\alpha_1 ar + \alpha_2 a^2 r}$,
  - $V = \left(\frac{e(C''_{2Z},C_{5Z})}{e(C''_{2X},C_{5X})}\right)^{\frac{1}{y_*}} \left(\frac{e(C''_{2Z1},C_{5Z})}{e(C''_{2Y},C_{5Y})}\right)^{\frac{1}{x_*}} = e(g_1g_2, h)^{a^2 r}$,
  - $Y = \{e(C_4, h)^{\alpha+\beta}/V^{\alpha_2}\}^{\frac{1}{(C_1 - svk^*)\alpha_1}} = e(g_1g_2, h)^{ar}$.

  Thus, $\mathcal{B}$ computes $Z = \{e(C_4, \hat{A}_{-1})^{\alpha+\beta}/Y^{\alpha_2}\}^{\frac{1}{\alpha_1(C_1 - svk^*)}} (= e(g_1g_2, h)^r)$, and $m = C_3/Z$. Note that if $m \in \{m_0, m_1\}$, $\mathcal{B}$ returns $\perp$.

Challenge ciphertext: The challenge ciphertext C^* is computed as follows where $d^* \stackrel{R}{\leftarrow} \{0,1\}$.

$$C^* = (svk^*, B^{x_*}, B^{\alpha y_*}, B^{z_*}, B^{\alpha z_*}, m_{d^*} \cdot T^{(\alpha+\beta)}, B^{\alpha_2}, \hat{B}^{\alpha_2}, \mathcal{S}(ssk^*, (C_3^*, C_4^*)).$$

If $T = e(g,h)^{b/a^2}$, we have

$$C^* = (svk^*, (g^{a^2x_*})^{b/a^2}, (g^{\alpha a^2 y_*})^{b/a^2}, (g^{a^2 z_*})^{b/a^2}, (g^{\alpha a^2 z_*})^{b/a^2}, m_{d^*} \cdot e(g_1g_2, h)^{b/a^2}, (g^{\alpha_2 a^2})^{b/a^2}, (h^{\alpha_2 a^2})^{b/a^2}, \mathcal{S}(ssk^*, (C_3^*, C_4^*)).$$

Thus, C^* is a valid ciphertext with the random exponent $r = b/a^2$. In contrast, if T is random, $\mathcal{A}$ cannot guess d^* with probability better than $1/2$. Therefore, $\mathcal{B}$ decides that $T = e(g,h)^{b/a^2}$ if d^* equals to the adversary's output and that T is random otherwise. □

Theorem 2. *Our proposed scheme with the strong one-time signature satisfies first level RCCA security if the 3-wDBDHI problem is hard.*

Proof. The proof is almost the same as that of Theorem 1. We can build an algorithm $\mathcal{B}$ which is, given $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{a^2}, B = g^b, h, \hat{A}_{-1} = h^{1/a}, \hat{A}_1 = h^a, \hat{A}_2 = h^{a^2}, \hat{B} = h^b, T)$, solving the modified 3-wDBDHI problem using RCCA adversary $\mathcal{A}$ at level 1.

The algorithm $\mathcal{B}$ simulates $\mathcal{A}$'s input and oracles as follows. The public parameters are set in the same way as that in the proof of Theorem 1. The algorithm $\mathcal{B}$ generates the public key for target user i^* as $pk_* = ({A_1}^{x_*}, {A_1}^{\alpha y_*}, {A_1}^{z_*}, {A_1}^{\alpha z_*}, \hat{A}_1^{x_*}, \hat{A}_1^{\alpha y_*}, \hat{A}_1^{\beta z_*}) = (g^{ax_*}, {g_1}^{ay_*}, g^{az_*}, {g_1}^{az_*}, h^{ax_*}, {h_1}^{ay_*}, {h_2}^{az_*})$. Here, the secret keys for the target user is $sk_* = (ax_*, ay_*, az_*)$. Note that $\mathcal{B}$ does not have to compute the secret keys of the target user to compute the corresponding public keys. For other users $j \neq i^*$, $\mathcal{B}$ computes $sk_j = (x_j, y_j, z_j)$ and pk_j by using the key-generation algorithm. The re-encryption keys R_{*j} can be computed $R_{*j} = (\hat{A}_{-1}^{\frac{\ell x_j + \alpha(\ell - n - 1)y_j}{x_*}}, \hat{A}_{-1}^{\frac{nx_j + \beta y_j}{y_*}}, \hat{A}_{-1}^{\frac{\ell x_j + \beta y_j}{z_*}}) = (h^{\frac{\ell x_j + \alpha(\ell - n - 1)y_j}{ax_*}}, h^{\frac{nx_j + \beta y_j}{ay_*}}, h^{\frac{\ell x_j + \beta y_j}{az_*}})$ where $\ell, n \xleftarrow{R} \mathbb{Z}_p^*$. The other re-encryption keys can be computed by following Rekeygen. For the first level decryption query, $\mathcal{B}$ responds in the same way as that for the honest user's case in the proof of Theorem 1.

For the challenge ciphertext C^*, $\mathcal{B}$ chooses $d^* \xleftarrow{R} \{0, 1\}$ and computes as follows where $s', t', k' \xleftarrow{R} \mathbb{Z}_p^*$.

$$C^* = (svk^*, {A_2}^{\alpha y_* s'}, B^{\alpha y_* s'}, {A_2}^{x_* t'}, B^{x_* t'}, {A_2}^{\alpha y_* k'}, B^{\alpha y_* k'}, {A_2}^{x_* k'}, B^{x_* k'}, m_{d^*} \cdot T^{\alpha+\beta},$$
$$B^{\alpha_2}, \hat{B}^{\alpha_2}, \hat{A}_{-1}^{\frac{\beta}{s'}}, \hat{A}_{-1}^{\frac{1}{t'}}, \hat{A}_{-1}^{\frac{1+\beta}{k'}}, \mathcal{S}(ssk^*, (C_3^*, C_4^*)).$$

If $T = e(g, h)^{b/a^2}$, we have

$$C^* = (svk^*, ({g_1}^{ay_*})^{as'}, (({g_1}^{ay_*})^{b/a^2})^{as'}, (g^{ax_*})^{at'}, ((g^{ax_*})^{b/a^2})^{at'}, ({g_1}^{ay_*})^{ak'}, (({g_1}^{ay_*})^{b/a^2})^{ak'},$$
$$(g^{ax_*})^{ak'}, ((g^{ax_*})^{b/a^2})^{ak'}, m_{d^*} \cdot e(g_1 g_2, g)^{b/a^2}, (g^{\alpha_2 a^2})^{b/a^2}, (h^{\alpha_2 a^2})^{b/a^2},$$
$$h^{\frac{\beta}{as'}}, h^{\frac{1}{at'}}, h^{\frac{1+\beta}{ak'}}, \mathcal{S}(ssk^*, (C_3^*, C_4^*))).$$

Thus, C^* is a valid ciphertext with the random exponents $r = b/a^2$, $s = as'$, $t = at'$, $k = ak'$. In contrast, if T is random, $\mathcal{A}$ cannot guess d^* with probability better than $1/2$. Therefore, $\mathcal{B}$ decides that $T = e(g, h)^{b/a^2}$ if d^* equals to the adversary's output and that T is random otherwise. □

Theorem 3. *Our proposed scheme meets sUFReKey-CA if the 2-DHI problem is hard.*

Proof. We specify the polynomial time algorithm $\mathcal{B}$ which solves the 2-DHI problem by using the polynomial time algorithm $\mathcal{A}$ which breaks the strong unforgeability of re-encryption keys against collusion attack of the proposed scheme. Given $(g, A_1 = g^a, A_2 = g^{a^2}, h, \hat{A}_1 = h^a, \hat{A}_2 = h^{a^2})$, $\mathcal{B}$ runs $\mathcal{A}$ with the following inputs:

Public parameters: $\mathcal{B}$ chooses $b, d \xleftarrow{R} \mathbb{Z}_p^*$, and sets $g_1 = A_1 \cdot g^{-d} = g^{a-d}$, $g_2 = {g_1}^b = g^{(a-d)b}$, $h_1 = \hat{A}_1 \cdot g^{-d} = h^{a-d}$ and $h_2 = {h_1}^b = h^{(a-d)b}$ (i.e. $\alpha = a - d$, $\beta = (a - d)b$). $\mathcal{B}$ generates $u, v, \hat{u}, \hat{v}$ by following Global-setup.

Public key pk_* for target user: $\mathcal{B}$ chooses $x, y, z \xleftarrow{R} \mathbb{Z}_p^*$ and computes $pk_* = ((A_2 \cdot {A_1}^{-d})^x, (A_2 \cdot {A_1}^{-d})^y, (A_1 \cdot g^{-d})^z, (A_2 \cdot {A_1}^{-2d} \cdot g^{d^2})^z, (\hat{A}_2 \cdot \hat{A}_1^{-d})^x, (\hat{A}_2 \cdot \hat{A}_1^{-d})^y, (\hat{A}_2 \cdot \hat{A}_1^{-d})^{by}) = (g^{(a-d)ax}, (g^{a-d})^{ay}, g^{(a-d)z}, (g^{a-d})^{(a-d)z}, h^{(a-d)ax}, (h^{a-d})^{ay}, (h^{(a-d)b})^{ay})$. Here, the corresponding secret key of the target honest user is $sk_* = (x_*, y_*, z_*)$ where $x_* = (a - d)ax, y_* = ay, z_* = (a - d)z$ Note that $\mathcal{B}$ does not have to compute sk_*.

Public and secret keys $(pk_c, sk_c), (pk_j, sk_j)$ for malicious users: The secret keys of the corrupt user $sk_c = (x_c, y_c, z_c)$ and the malicious user $sk_j = (x_j, y_j, z_j)$ are set as $x_c, y_c, z_c \xleftarrow{R} \mathbb{Z}_p^*$ and $y_j, z_j \xleftarrow{R} \mathbb{Z}_p^*$, $x_j = dy_j$. The public keys pk_c and pk_j are computed by following Keygen.

Re-encryption key R_{*c}: The re-encryption key $R_{*c} = (R_{*c1}, R_{*c2}, R_{*c3})$ is computed as

- $R_{*c1} = \left[h^{-n'} \cdot \left((\hat{A}_1 \cdot h^{s-d})^{\ell'} \cdot h^{-\frac{1}{s-d}}\right)^{-(s+bd)}\right]^{\frac{y_c}{sx}} = \left[h^{-n'} \cdot h^{-(s+bd)((a+s-d)\ell' - \frac{1}{s-d})}\right]^{\frac{y_c}{sx}}$,
- $R_{*c2} = h^{\frac{(n'+b)y_c}{y}}$, and
- $R_{*c3} = \left[(\hat{A}_1^{\ell'} \cdot h^{-\frac{1}{s-d}})^{-(s+bd)} \cdot h^b\right]^{\frac{y_c}{z}} = \left[h^{-(a\ell' - \frac{1}{s-d})(s+bd)+b}\right]^{\frac{y_c}{z}}$.

Here, $\ell', n' \xleftarrow{R} \mathbb{Z}_p^*$ and $s = \frac{x_c}{y_c}$. This is a correct re-encryption key with the randomness $\ell = -(a-d)(a\ell' - \frac{1}{s-d})\frac{s+bd}{s}$, $n = \frac{an'+bd}{s}$. We can check this as follows.

- $R_{*c1} = h^{\frac{\ell x_c + \alpha(\ell - n - 1)y_c}{x_*}} = h^{\frac{\ell s y_c + (a-d)(\ell - n - 1)y_c}{(a-d)ax}} = \left(h^{-\frac{n+1}{a}} \cdot h^{\frac{\ell(s+a-d)}{(a-d)a}}\right)^{\frac{y_c}{x}}$. Here, we have
  - $h^{-\frac{n+1}{a}} = h^{-\frac{\frac{an'+bd}{s}+1}{a}} = h^{-\frac{n'}{s}} \cdot h^{-\frac{s+bd}{sa}}$, and
  - $h^{\frac{\ell(s+a-d)}{(a-d)a}} = h^{\frac{-(a-d)(a\ell' - \frac{1}{s-d})\frac{s+bd}{s}(s+a-d)}{(a-d)a}} = h^{\frac{-\frac{s+bd}{s}(a(s+a-d)\ell' - \frac{s+a-d}{s-d})}{a}} = h^{\frac{-\frac{s+bd}{s}(a(a+s-d)\ell' - \frac{a+(s-d)}{s-d})}{a}}$
  $= h^{-\frac{s+bd}{s}((a+s-d)\ell' - \frac{1}{s-d} - \frac{1}{a})} = h^{-\frac{s+bd}{s}((a+s-d)\ell' - \frac{1}{s-d})} \cdot h^{\frac{s+bd}{sa}}$.

  Thus, $R_{*c1} = \left(h^{-\frac{n'}{s}} \cdot h^{-\frac{s+bd}{sa}} \cdot h^{-\frac{s+bd}{s}((a+s-d)\ell' - \frac{1}{s-d})} \cdot h^{\frac{s+bd}{sa}}\right)^{\frac{y_c}{x}} = \left[h^{-n'} \cdot h^{-(s+bd)((a+s-d)\ell' - \frac{1}{s-d})}\right]^{\frac{y_c}{sx}}$.
- $R_{*c2} = h^{\frac{nx_c + \beta y_c}{y_*}} = h^{\frac{nsy_c + (a-d)by_c}{ay}} = h^{\frac{(ns+(a-d)b)y_c}{ay}} = h^{\frac{\left(\frac{an'+bd}{s}s + (a-d)b\right)y_c}{ay}}$
$= h^{\frac{(an'+bd+ab-bd)y_c}{ay}} = h^{\frac{(n'+b)y_c}{y}}$.
- $R_{*c3} = h^{\frac{\ell x_c + \beta y_c}{z_*}} = h^{\frac{\ell s y_c + (a-d)by_c}{(a-d)z}} = h^{\frac{(\ell s + (a-d)b)y_c}{(a-d)z}} = h^{\frac{\left(-(a-d)(a\ell' - \frac{1}{s-d})\frac{s+bd}{s}s + (a-d)b\right)y_c}{(a-d)z}}$
$= h^{\frac{\left(-(a\ell' - \frac{1}{s-d})(s+bd)+b\right)y_c}{z}} = \left[h^{-(a\ell' - \frac{1}{s-d})(s+bd)+b}\right]^{\frac{y_c}{z}}$.

Then, $\mathcal{B}$ receives $\mathcal{A}$'s output $R_{*j}^\dagger = (R_1, R_2, R_3)$. Finally, $\mathcal{B}$ outputs $W = \left(\frac{{R_3}^z}{{R_1}^{dx} \cdot {R_2}^y}\right)^{\frac{1}{(1+b)x_j}}$ as the answer of the 2-DHI problem.

We show that the algorithm $\mathcal{B}$ outputs $h^{\frac{1}{a}}$ with non-negligible probability. The distributions of the public parameters and the public/secret/re-encryption keys are identical to those of our proposed scheme except when any one of the following events occurs: "$a - d = 0$", "$s - d = 0$", "$\ell = -(a-d)(a\ell' - \frac{1}{s-d})\frac{s+bd}{s} = 0$", "$n = \frac{an'+bd}{s} = 0$". It is easy to see that the probability that any one of the above events occurs is negligible. Therefore, the algorithm $\mathcal{A}$ outputs a (forged) re-encryption key $R_{*j}^\dagger = (R_1, R_2, R_3)$ which satisfies $m = \mathsf{Dec_1}(\mathsf{Reenc}(R_{*j}^\dagger, \mathsf{Enc_2}(m, pk_*)), sk_j)$ with non-negligible probability. From this equation and the encryption/decryption/re-encryption algorithms of our proposed scheme, we have

$$m = m \cdot e(g_1 g_2, h)^r \Big/ \left\{ \left(\frac{e(g^{z_* rk}, {R_3}^{1/k})}{e(g^{x_* rs}, {R_1}^{1/s})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha z_* rk}, {R_3}^{1/k})}{e(g^{\alpha y_* rt}, {R_2}^{1/t})} \right)^{\frac{1}{x_j}} \right\}$$
$$\Leftrightarrow m \cdot e\left(g, \frac{{R_3}^{z_*(x_j+\alpha y_j)}}{{R_1}^{x_* x_j} \cdot {R_2}^{\alpha y_* y_j}}\right)^r = m \cdot e(g, h^{x_j y_j (\alpha+\beta)})^r.$$

Therefore, $\frac{{R_3}^{z_*(x_j+\alpha y_j)}}{{R_1}^{x_* x_j} \cdot {R_2}^{\alpha y_* y_j}} = h^{x_j y_j (\alpha+\beta)}$. Since we set $x_* = (a-d)ax$, $y_* = ay$, $z_* = (a-d)z$, $x_j = dy_j$, $\alpha = a - d$, $\beta = (a-d)b$, and the probability that $a - d = 0$ or $1 + b = 0$ is negligible, we have

$$\frac{{R_3}^{(a-d)z(dy_j+(a-d)y_j)}}{{R_1}^{(a-d)axdy_j} \cdot {R_2}^{(a-d)ayy_j}} = h^{x_j y_j (a-d+(a-d)b)} \Leftrightarrow (W =) \left(\frac{{R_3}^z}{{R_1}^{dx} \cdot {R_2}^y} \right)^{\frac{1}{(1+b)x_j}} = h^{\frac{1}{a}}.$$

Thus, algorithm $\mathcal{B}$ outputs $h^{\frac{1}{a}}$ with non-negligible probability. □

4 A UFReKey-CA Secure Proxy Re-Encryption Scheme with Temporary Delegation

In this section, we apply similar modification of the re-encryption keys to the PRE scheme with temporary delegation in [4] and propose the PRE scheme supporting temporary delegation which meets sUFReKey-CA.

4.1 Definitions

In this section, we describe the syntactic definition of unidirectional proxy re-encryption with temporary delegation and its security notion [4]. In the PRE scheme with temporary delegation, it only allows the proxy to re-encrypt messages during a limited time period.

4.1.1 Model

First, we describe the model of unidirectional PRE scheme supporting temporary delegation. It is almost the same as that in Definition 4 except that re-encryption key generation, encryption, and re-encryption algorithms take a period $L \in \{1, ..., L_{max}\}$ as input. Intuitively, the re-encryption key generated by Rekeygen with a period L, can be used to re-encrypt the ciphertext generated by Enc_2 with the same period L. Note that the public and secret keys are common to all time periods.

First, we describe the syntactic definition of unidirectional proxy re-encryption.

Definition 9. *A (single-hop) unidirectional proxy re-encryption (PRE) scheme with temporary delegation consists of the following algorithms:*

Global-setup(λ) *is a probabilistic algorithm which takes a security parameter λ and returns public parameters* par *with a plaintext space $\mathcal{M}$.*

Keygen(λ, par) *is a probabilistic algorithm which takes parameters λ and* par *and returns a public/secret key pair (pk, sk).*

$\mathsf{Enc}_1(L, m, pk_j, \mathsf{par})$ *is a probabilistic algorithm which takes a time period L, a plaintext $m \in \mathcal{M}$, a user j's public key pk_j, and* par*, and returns a first level ciphertext C_j for L and j, which cannot be re-encrypted for another user.*

$\mathsf{Enc}_2(L, m, pk_i, \mathsf{par})$ *is a probabilistic algorithm which takes a time period* L*, a plaintext* $m \in \mathcal{M}$*, a user* i*'s public key* pk_i*, and* par*, and returns a second level ciphertext* C_i *for* L *and* i*, which can be re-encrypted with re-encryption keys for another user.*

$\mathsf{Rekeygen}(L, sk_i, pk_j, \mathsf{par})$ *is a probabilistic algorithm which takes a time period* L*, a user* i*'s secret key* sk_i*, a user* j*'s public key* pk_j*, and* par*, and returns a re-encryption key* R_{ijL} *for* L *to re-encrypt second level ciphertexts for* i *into first level ciphertexts for* j*.*

$\mathsf{Reenc}(L, R_{ijL}, C_i, \mathsf{par})$ *is a probabilistic algorithm which takes a time period* L*, a re-encryption key* R_{ijL}*, a second level ciphertext* C_i *for* L *encrypted under* pk_i*, and public parameters* par*, and returns a first level ciphertext* C_j *for* L *which is re-encrypted for* j *or a distinguished message '*`invalid`*.'*

$\mathsf{Dec}_1(C_j, sk_j, \mathsf{par})$ *is a deterministic algorithm which takes a first level ciphertext* C_j *for* j*, a user* j*'s secret key* sk_j*, and* par*, and returns a plaintext* m *or a distinguished message '*`invalid`*'.*

$\mathsf{Dec}_2(C_i, sk_i, \mathsf{par})$ *is a deterministic algorithm which takes a second level ciphertext* C_i *for* i*, a user* i*'s secret key* sk_i*, and* par*, and returns a plaintext* m *or a distinguished message '*`invalid`*'.*

To lighten notations, we will sometimes omit to explicitly write public parameters par, taken as input by all but one of the above algorithms.

4.1.2 Security Definitions – Confidentiality

Next, we describe the security notion of a PRE scheme with temporary delegation. In Definition 5, the challenger generates public keys for all parties and allows the adversary to obtain private keys for some of them (*known key model*).

On the other hand, in [4], a stronger security notion is also proposed, called RCCA security in the *chosen key model* (RCCA-CK security). In this model, the adversary can *arbitrarily* choose public keys without demonstrating knowledge of the private keys. This provides the adversary with much more flexibility and power in attacking other honest parties in the system. The definition of RCCA-CK security can be extended to that for PRE schemes with temporary delegation.

First, we describe the definition of RCCA-CK security of second level ciphertexts for PRE schemes with temporary delegation.

Definition 10 (Second level RCCA-CK security for PRE schemes with temporary delegation)**.** *A unidirectional single-hop proxy re-encryption scheme with temporary delegation is second level secure against replayable chosen-ciphertext attack in the chosen key model (or RCCA-CK secure for short) if*

$$\begin{aligned}|\Pr[\mathsf{par} \leftarrow \mathsf{Global\text{-}setup}(\lambda);\ \{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i \in HU};\ \{R_{ii'} \leftarrow \mathsf{Rekeygen}(sk_i, pk_{i'})\}_{i,i' \in HU};\\ (m_0, m_1, i^*, L^*, St) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{deleg}}, \mathcal{O}_{\text{reenc}}, \mathcal{O}_{\text{1-dec}}}(\{pk_i\}, \{R_{ii'}\});\ d^* \overset{R}{\leftarrow} \{0,1\};\\ C^* \overset{R}{\leftarrow} \mathsf{Enc}_2(L^*, m_{d^*}, pk_{i^*});\ d' \overset{R}{\leftarrow} \mathcal{A}^{\mathcal{O}_{\text{deleg}}, \mathcal{O}_{\text{reenc}}, \mathcal{O}_{\text{1-dec}}}(C^*, St) : d' = d^*] - 1/2|\end{aligned}$$

is negligible for any polynomial time algorithm $\mathcal{A}$*. Above,* St *is the state information maintained by* $\mathcal{A}$*,* HU *is the set of honest users,* $i^* \in HU$ *is the target user, and* L^* *is the target time period.*

Oracles $\mathcal{O}_{\text{deleg}}$*,* $\mathcal{O}_{\text{reenc}}$*, and* $\mathcal{O}_{\text{1-dec}}$ *proceed as follows:*

- *Delegation oracle* $\mathcal{O}_{\text{deleg}}$*: on input* (L, pk_i, pk_j) *where* L *is a time period,* pk_i *is a public key of honest user* $i \in HU$ *(and either* $L \neq L^*$ *or* $i \neq i^*$ *in any stage), and* pk_j *is a public key which* $\mathcal{A}$ *chooses arbitrary, this oracle responds with* $\mathsf{Rekeygen}(L, sk_i, pk_j)$*.*

- *Re-encryption oracle* $\mathcal{O}_{\text{reenc}}$*: on input* (L, pk_i, pk_j, C) *where* L *is a time period,* C *is a second level ciphertext,* pk_i *is a public key of honest user* $i \in HU$*, and* pk_j *is a public key which* $\mathcal{A}$ *chooses arbitrary, this oracle responds with* `invalid` *if* C *is ill-formed with respect to* pk_i*. It returns a special symbol* $\perp$ *if* $j \notin HU$ *and* $(L, pk_i, C) = (L^*, pk_*, C^*)$*. Otherwise, the re-encrypted first level ciphertext* $C_j = \mathsf{Reenc}(\mathsf{Rekeygen}(L, sk_i, pk_j), C)$ *is returned to* $\mathcal{A}$*.*

- *First level decryption oracle* $\mathcal{O}_{\text{1-dec}}$*: given a pair* (pk_i, C)*, where* C *is a first level ciphertext and* $i \in HU$*, this oracle returns* `invalid` *if* C *is ill-formed with respect to* pk_i*. If the query occurs in the guess stage, it outputs a special symbol* $\perp$ *if* (pk_i, C) *is a* Derivative *of the challenge pair* (pk_{i^*}, C^*)*. Otherwise, the plaintext* $m = \mathsf{Dec_1}(sk_i, C)$ *is returned to* $\mathcal{A}$*.* Derivative*s of* (pk_{i^*}, C^*) *are defined as follows: If* C *is a first level ciphertext and* $i \in HU$*,* (pk_i, C) *is a* Derivative *of* (pk_{i^*}, C^*) *if* C *and* C^* *are encrypted for the same time period* L^* *and* $\mathsf{Dec_1}(sk_i, C) \in \{m_0, m_1\}$*.*

Next, we describe the definition of RCCA-CK security of first level ciphertexts for PRE schemes with temporary delegation.

Definition 11 (First level RCCA-CK security for PRE schemes with temporary delegation)**.** *A unidirectional single-hop proxy re-encryption scheme with temporary delegation is first level secure against replayable chosen-ciphertext attack in the chosen key model (RCCA-CK) (or RCCA-CK secure for short) if*

$$\begin{aligned}|\Pr[\mathsf{par} \leftarrow \mathsf{Global\text{-}setup}(\lambda);\ \{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i \in HU};\ \{R_{ii'} \leftarrow \mathsf{Rekeygen}(sk_i, pk_{i'})\}_{i,i' \in HU};\\ (m_0, m_1, i^*, L^*, St) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{deleg}}, \mathcal{O}_{\text{1-dec}}}(\{pk_i\}, \{R_{ii'}\});\ d^* \xleftarrow{R} \{0,1\};\\ C^* \xleftarrow{R} \mathsf{Enc_1}(L^*, m_{d^*}, pk_{i^*});\ d' \xleftarrow{R} \mathcal{A}^{\mathcal{O}_{\text{deleg}}, \mathcal{O}_{\text{1-dec}}}(C^*, St) : d' = d^*] - 1/2|\end{aligned}$$

is negligible for any polynomial time algorithm $\mathcal{A}$*. Above,* St *is the state information maintained by* $\mathcal{A}$*,* HU *is the set of honest users,* $i^* \in HU$ *is the target user, and* L^* *is the target time period. An oracle* $\mathcal{O}_{\text{deleg}}$ *responds with* $\mathsf{Rekeygen}(L, sk_i, pk_j)$ *for* any *query* (L, pk_i, pk_j) *where* $i \in HU$*. That is,* (L^*, pk_{i^*}, pk_j) *can be queried to* $\mathcal{O}_{\text{deleg}}$*. An oracle* $\mathcal{O}_{\text{1-dec}}$ *is the same as that in Definition 10 except that the definition of* Derivative*s: If* C *is a first level ciphertext and* $pk = pk_*$*,* (pk, C) *is a* Derivative *of* (pk_*, C^*) *if* $\mathsf{Dec_1}(sk, C) \in \{m_0, m_1\}$*.*

In the above definition, all re-encryption keys are available to the adversary. Therefore, the re-encryption oracle becomes useless and is not given to the adversary.

4.1.3 Security Definitions – Unforgeability of Re-Encryption Keys

We describe the definition of sUFReKey-CA for PRE schemes with temporary delegation. In this security game, the adversary, given a target time period $L^* \xleftarrow{R} \{1, ..., L_{max}\}$, re-encryption keys R_{*cL} for any corrupted delegatee $c(\neq j)$ at any period $1 \leq L \leq L_{max}$, and re-encryption keys R_{*jL} for the malicious user j at period $L \neq L^*$, tries to forge R_{*jL^*}.

Definition 12 (Strong Unforgeability of Re-Encryption Keys against Collusion Attack for PRE schemes with temporary delegation [1])**.** *A unidirectional single-hop proxy re-encryption scheme with temporary delegation meets* the strong unforgeability of re-encryption keys against collusion attack *if the following probability is negligible for any polynomial time algorithm* $\mathcal{A}$ *and any* L_{max}

which is polynomially bounded.

$$\Pr[\mathsf{par} \leftarrow \mathsf{Global\text{-}setup}(\lambda);\ (pk_*, sk_*) \leftarrow \mathsf{Keygen}(\lambda);\ \{(pk_c, sk_c) \leftarrow \mathsf{Keygen}(\lambda)\};$$
$$(pk_j, sk_j) \leftarrow \mathsf{Keygen}(\lambda);\ L^* \xleftarrow{R} \{1, 2, ..., L_{max}\};$$
$$\{R_{*cL} \leftarrow \mathsf{Rekeygen}(L, sk_*, pk_c)\};\ \{R_{*jL} \leftarrow \mathsf{Rekeygen}(L, sk_*, pk_j)\}_{L \neq L^*};$$
$$m \xleftarrow{R} \mathcal{M};\ C^* \leftarrow \mathsf{Enc}_2(L^*, m, pk_*);$$
$$R^{\dagger}_{*jL^*} \leftarrow \mathcal{A}(L^*, pk_*, \{(pk_c, sk_c)\}, (pk_j, sk_j), \{R_{*cL}\}, \{R_{*jL}\}_{L \neq L^*}) :$$
$$m = \mathsf{Dec}_1(\mathsf{Reenc}(L^*, R^{\dagger}_{*jL^*}, C^*), sk_j)]$$

4.2 Our Proposed Scheme

In this section, we propose a PRE scheme with temporary delegation which meets RCCA-CK and sUFReKey-CA. In [4], Libert and Vergnaud proposed the PRE scheme with temporary delegation by modifying their PRE scheme. By using the similar modification, we can construct the PRE scheme with temporary delegation from our proposed PRE scheme described in Section 3.

4.2.1 Description

$\mathsf{Global\text{-}setup}(\lambda)$: the same as that in Section 3.

$\mathsf{Keygen}(\lambda, \mathsf{par})$: user i chooses $x_i, y_i, z_i \xleftarrow{R} \mathbb{Z}_p^*$. The secret key is $sk_i = (x_i, y_i, z_i)$. The public key is $pk_i = (X_i, Y_{1i}, Z_i, Z_{1i}, W_i, \hat{X}_i, \hat{Y}_{1i}, \hat{Y}_{2i}, \hat{W}_i)$ where $X_i = g^{x_i}, Y_{1i} = {g_1}^{y_i}, Z_i = g^{z_i}, Z_{1i} = {g_1}^{z_i}, W_i = g^{w_i}, \hat{X}_i = h^{x_i}, \hat{Y}_{1i} = {h_1}^{y_i}, \hat{Y}_{2i} = {h_2}^{y_i}, \hat{W}_i = h^{w_i}$.

$\mathsf{Enc}_1(L, m, pk_j, \mathsf{par})$: to encrypt a message $m \in \mathbb{G}_T$ under the public key pk_j at the first level during period L, the sender proceeds as follows:

1. Select a one-time signature key pair $(svk, ssk) \leftarrow \mathcal{G}(\lambda)$ and set $C_1 = svk$.
2. Pick $r, s, t, k, \pi, \delta_x, \delta_y \xleftarrow{R} \mathbb{Z}_p^*$ and compute,
   $C'_{2X} = {Y_{1j}}^s, C''_{2X} = {Y_{1j}}^{rs}, C'_{2Y} = {X_j}^t, C''_{2Y} = {X_j}^{rt}, C'_{2Z} = {Y_{1j}}^k, C''_{2Z} = {Y_{1j}}^{rk}, C'_{2Z1} = {X_j}^k, C''_{2Z1} = {X_j}^{rk}, C'_{2F} = (g^L \cdot W_j)^{\pi}, C''_{2F} = (g^L \cdot W_j)^{r\pi}, C_3 = e(g_1 g_2, h)^r \cdot m, C_4 = (u^{svk} \cdot v)^r, \hat{C}_4 = (\hat{u}^{svk} \cdot \hat{v})^r, C_{5X} = (h_2 \cdot (h^L \cdot \hat{W}_j)^{\delta_y})^{\frac{1}{s}} = h^{\frac{\beta + (L + w_j)\delta_y}{s}}, C_{5Y} = (h \cdot (h^L \cdot \hat{W}_j)^{\delta_x})^{\frac{1}{t}} = h^{\frac{1 + (L + w_j)\delta_x}{t}}, C_{5Z} = (h \cdot h_2)^{\frac{1}{k}} = h^{\frac{1+\beta}{k}}, C_{5FX} = (\hat{Y}_{1j})^{\frac{\delta_y}{\pi}}, C_{5FY} = (\hat{X}_j)^{\frac{\delta_x}{\pi}}$
3. Generate a one-time signature $\sigma \leftarrow \mathcal{S}(ssk, (L, C_3, C_4))$ on (L, C_3, C_4).

The (first level) ciphertext is $C_j = (L, C_1, C'_{2X}, C''_{2X}, C'_{2Y}, C''_{2Y}, C'_{2Z}, C''_{2Z}, C'_{2Z1}, C''_{2Z1}, C'_{2F}, C''_{2F}, C_3, C_4, \hat{C}_4, C_{5X}, C_{5Y}, C_{5Z}, C_{5FX}, C_{5FY}, \sigma)$.

$\mathsf{Enc}_2(L, m, pk_i, \mathsf{par})$: to encrypt a message $m \in \mathbb{G}_T$ under the public key pk_i at the second level during period L, the sender proceeds as follows:

1. Select a one-time signature key pair $(svk, ssk) \leftarrow \mathcal{G}(\lambda)$ and set $C_1 = svk$.
2. Pick $r \xleftarrow{R} \mathbb{Z}_p^*$ and compute, $C_{2X} = {X_i}^r, C_{2Y} = {Y_{1i}}^r, C_{2Z} = {Z_i}^r, C_{2Z1} = {Z_{1i}}^r, C_{2F} = (g^L \cdot W_i)^r, C_3 = e(g_1 g_2, g)^r \cdot m, C_4 = (u^{svk} \cdot v)^r, \hat{C}_4 = (\hat{u}^{svk} \cdot \hat{v})^r$.
3. Generate a one-time signature $\sigma \leftarrow \mathcal{S}(ssk, (L, C_3, C_4))$ on (L, C_3, C_4).

The (second level) ciphertext is $C_i = (L, C_1, C_{2X}, C_{2Y}, C_{2Z}, C_{2Z1}, C_{2F}, C_3, C_4, \hat{C}_4, \sigma)$.

$\mathsf{Rekeygen}(L, sk_i, pk_j, \mathsf{par})$: given a period number L, user i's secret key sk_i and user j's public key pk_j, generate the re-encryption key $R_{ijL} = (R_{ijL1}, R_{ijL2}, R_{ijL3}, R_{ijL4}, R_{ijL5})$ where $\theta, n, \delta_x, \delta_y \xleftarrow{R} \mathbb{Z}_p^*$ and

$$R_{ijL1} = (\hat{X}_j^{\ell} \cdot \hat{Y}_{1j}^{\ell-n-1})^{1/x_i} \cdot (h^L \cdot \hat{W}_i)^{\delta_x} = h^{\frac{\ell x_j + \alpha(\ell-n-1)y_j}{x_i} + (L+w_i)\delta_x},$$
$$R_{ijL2} = (\hat{X}_j^{n} \cdot \hat{Y}_{2j})^{1/y_i} \cdot (h^L \cdot \hat{W}_i)^{\delta_y} = h^{\frac{n x_j + \beta y_j}{y_i} + (L+w_i)\delta_y}, R_{ijL3} = (\hat{X}_j^{\ell} \cdot \hat{Y}_{2j})^{1/z_i} = h^{\frac{\ell x_j + \beta y_j}{z_i}},$$
$$R_{ijL4} = \hat{X}_i^{\delta_x} = h^{x_i \delta_x}, R_{ijL5} = \hat{Y}_{1i}^{\delta_y} = h^{\alpha y_i \delta_y}.$$

$\mathsf{Reenc}(L, R_{ijL}, C_i, \mathsf{par})$: on input of the re-encryption key R_{ijL} for period L and a second level ciphertext C_i, check the validity of the ciphertext by testing:

$$\begin{array}{l} e(C_{2X}, \hat{u}^{C_1} \cdot \hat{v}) = e(X_i, \hat{C}_4),\ e(C_{2Y}, \hat{u}^{C_1} \cdot \hat{v}) = e(Y_{1i}, \hat{C}_4),\ e(C_{2Z}, \hat{u}^{C_1} \cdot \hat{v}) = e(Z_i, \hat{C}_4), \\ e(C_{2Z1}, \hat{u}^{C_1} \cdot \hat{v}) = e(Z_{1i}, \hat{C}_4),\ e(C_{2F}, \hat{u}^{C_1} \cdot \hat{v}) = e(g^L \cdot W_i, \hat{C}_4),\ e(g, \hat{C}_4) = e(C_4, h), \\ \mathcal{V}(C_1, \sigma, (L, C_3, C_4)) = 1. \end{array} \quad (3)$$

If the relations (3) hold (well-formed), C_i is re-encrypted by computing $C'_{2X} = {X_i}^s, C''_{2X} = {C_{2X}}^s, C'_{2Y} = {Y_{1i}}^t, C''_{2Y} = {C_{2Y}}^t, C'_{2Z} = {Z_i}^k, C''_{2Z} = {C_{2Z}}^k, C'_{2Z1} = {Z_{1i}}^k, C''_{2Z1} = {C_{2Z1}}^k, C'_{2F} = (g^L \cdot W_i)^{\pi}, C''_{2F} = {C_{2F}}^{\pi}, C_{5X} = {R_{ijL1}}^{\frac{1}{s}}, C_{5Y} = {R_{ijL2}}^{\frac{1}{t}}, C_{5Z} = {R_{ijL3}}^{\frac{1}{k}}, C_{5FX} = {R_{ijL4}}^{\frac{1}{\pi}}, C_{5FY} = {R_{ijL5}}^{\frac{1}{\pi}}$ where $s, t, k, \pi \xleftarrow{R} \mathbb{Z}_p^*$, and re-encrypted ciphertext $C_j = (L, C_1, C'_{2X}, C''_{2X}, C'_{2Y}, C''_{2Y}, C'_{2Z}, C''_{2Z}, C'_{2Z1}, C''_{2Z1}, C'_{2F}, C''_{2F}, C_3, C_4, \hat{C}_4, C_{5X}, C_{5Y}, C_{5Z}, C_{5FX}, C_{5FY}, \sigma)$ is returned. Otherwise (ill-formed), the algorithm outputs '`invalid`.'

$\mathsf{Dec}_1(C_j, sk_j)$: the validity of the first level ciphertext C_j is checked by testing:

$$\begin{array}{l} e(C''_{2X}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2X}, \hat{C}_4),\ e(C''_{2Y}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2Y}, \hat{C}_4), \\ e(C''_{2Z}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2Z}, \hat{C}_4),\ e(C''_{2Z1}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2Z1}, \hat{C}_4), \\ e(C''_{2F}, \hat{u}^{C_1} \cdot \hat{v}) = e(C'_{2F}, \hat{C}_4),\ e(g, \hat{C}_4) = e(C_4, h),\ \hat{V}(C_1, \sigma, (L, C_3, C_4)) = 1, \\ \left(\frac{e(C'_{2Z}, C_{5Z}) \cdot e(C'_{2F}, C_{5FX})}{e(C'_{2X}, C_{5X})}\right)^{\frac{1}{y_j}} \cdot \left(\frac{e(C'_{2Z1}, C_{5Z}) \cdot e(C'_{2F}, C_{5FY})}{e(C'_{2Y}, C_{5Y})}\right)^{\frac{1}{x_j}} = e(g_1 g_2, h). \end{array} \quad (4)$$

If the relations (4) hold (well-formed), the plaintext

$$m = C_3 \Big/ \left\{ \left(\frac{e(C''_{2Z}, C_{5Z}) \cdot e(C''_{2F}, C_{5FX})}{e(C''_{2X}, C_{5X})}\right)^{\frac{1}{y_j}} \cdot \left(\frac{e(C''_{2Z1}, C_{5Z}) \cdot e(C''_{2F}, C_{5FY})}{e(C''_{2Y}, C_{5Y})}\right)^{\frac{1}{x_j}} \right\}$$

is returned. Otherwise (ill-formed), the algorithm outputs '`invalid`.'

$\mathsf{Dec}_2(C_i, sk_i)$: if the second level ciphertext C_i satisfies the relations (3) (well-formed), the plaintext $m = C_3 \,/\, e(C_{2X}, h_1 h_2)^{\frac{1}{x_i}}$ is returned. Otherwise (ill-formed), the algorithm outputs '`invalid`.'

We can check the correctness property of the above scheme as follows. Note that Equations (3) ensure $(r =) \log_{X_i} C_{2X} = \log_{Y_{1i}} C_{2Y} = \log_{Z_i} C_{2Z} = \log_{Z_{1i}} C_{2Z1} = \log_{g^L \cdot W_i} C_{2F} = \log_{u^{C_1} v} C_4 = \log_{\hat{u}^{C_1} \hat{v}} \hat{C}_4$, and Equations (4) ensure $(r =) \log_{C'_{2X}} C''_{2X} = \log_{C'_{2Y}} C''_{2Y} = \log_{C'_{2Z}} C''_{2Z} = \log_{C_{2Z1}} C_{2Z1} = \log_{C'_{2F}} C''_{2F} = \log_{u^{C_1} v} C_4 = \log_{\hat{u}^{C_1} \hat{v}} \hat{C}_4$.

- $\mathsf{Dec}_2(\mathsf{Enc}_2(L, m, pk_i, \mathsf{par}), sk_i, \mathsf{par}) = C_3 \,/\, e(C_{2X}, h_1 h_2)^{\frac{1}{x_i}} = m \cdot e(g_1 g_2, h)^r \,/\, e(g^{x_i r}, h_1 h_2)^{\frac{1}{x_i}}$
  $= m \cdot e(g, h)^{(\alpha+\beta)r} / e(g, h)^{(\alpha+\beta)r} = m.$

- $\mathsf{Dec}_1(\mathsf{Enc}_1(L, m, pk_j, \mathsf{par}), sk_j, \mathsf{par})$

$$= C_3 \Big/ \left\{ \left(\frac{e(C''_{2Z}, C_{5Z}) \cdot e(C''_{2F}, C_{5FX})}{e(C''_{2X}, C_{5X})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(C''_{2Z1}, C_{5Z}) \cdot e(C''_{2F}, C_{5FY})}{e(C''_{2Y}, C_{5Y})} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ \left(\frac{e(g^{\alpha y_j r k}, h^{\frac{1+\beta}{k}}) \cdot e(g^{(L+w_j) r \pi}, h^{\frac{\alpha y_j \delta_y}{\pi}})}{e(g^{\alpha y_j r s}, h^{\frac{\beta + (L+w_j)\delta_y}{s}})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{x_j r k}, h^{\frac{1+\beta}{k}}) \cdot e(g^{(L+w_j) r \pi}, h^{\frac{x_j \delta_x}{\pi}})}{e(g^{x_j r t}, h^{\frac{1+(L+w_j)\delta_x}{t}})} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ \left(\frac{e(g^{\alpha y_j r}, h^{1+\beta}) \cdot e(g^{(L+w_j) r}, h^{\alpha y_j \delta_y})}{e(g^{\alpha y_j r}, h^{\beta + (L+w_j)\delta_y})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{x_j r}, h^{1+\beta}) \cdot e(g^{(L+w_j) r}, h^{x_j \delta_x})}{e(g^{x_j r}, h^{1+(L+w_j)\delta_x})} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ e(g^{\alpha y_j r}, h)^{\frac{1}{y_j}} \cdot e(g^{x_j r}, h^{\beta})^{\frac{1}{x_j}} \right\}$$

$$= m \cdot e(g_1 g_2, h)^r \big/ (e(g, h)^{\alpha r} \cdot e(g, h)^{\beta r}) = m.$$

- $\mathsf{Dec}_1(\mathsf{Reenc}(L, \mathsf{Rekeygen}(L, sk_i, pk_j, \mathsf{par}), \mathsf{Enc}_2(L, m, pk_i, \mathsf{par}), \mathsf{par}), sk_j, \mathsf{par})$

$$C_3 \Big/ \left\{ \left(\frac{e(C''_{2Z}, C_{5Z}) \cdot e(C''_{2F}, C_{5FX})}{e(C''_{2X}, C_{5X})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(C''_{2Z1}, C_{5Z}) \cdot e(C''_{2F}, C_{5FY})}{e(C''_{2Y}, C_{5Y})} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ \left(\frac{e(g^{z_i r k}, h^{\frac{\ell x_j + \beta y_j}{z_i k}}) \cdot e(g^{(L+w_i) r \pi}, h^{\frac{x_i \delta_x}{\pi}})}{e(g^{x_i r s}, h^{\frac{((\ell x_j + \alpha(\ell - n - 1) y_j)/x_i) + (L+w_i)\delta_x}{s}})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha z_i r k}, h^{\frac{\ell x_j + \beta y_j}{z_i k}},) \cdot e(g^{(L+w_i) r \pi}, h^{\frac{\alpha y_i \delta_y}{\pi}})}{e(g^{\alpha y_i r t}, h^{\frac{((n x_j + \beta y_j)/y_i) + (L+w_i)\delta_y}{t}},)} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ \left(\frac{e(g^r, h^{\ell x_j + \beta y_j}) \cdot e(g^{(L+w_i) r}, h^{x_i \delta_x})}{e(g^r, h^{\ell x_j + \alpha(\ell - n - 1) y_j + (L+w_i) x_i \delta_x})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha r}, h^{\ell x_j + \beta y_j}) \cdot e(g^{(L+w_i) r}, h^{\alpha y_i \delta_y})}{e(g^{\alpha r}, h^{n x_j + \beta y_j + (L+w_i) y_i \delta_y})} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ \left(\frac{e(g^r, h^{\ell x_j + \beta y_j})}{e(g^r, h^{\ell x_j + \alpha(\ell - n - 1) y_j})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha r}, h^{\ell x_j + \beta y_j})}{e(g^{\alpha r}, h^{n x_j + \beta y_j})} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ \left(\frac{e(g^r, h^{\beta y_j})}{e(g^r, h^{\alpha(\ell - n - 1) y_j})} \right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha r}, h^{\ell x_j})}{e(g^{\alpha r}, h^{n x_j})} \right)^{\frac{1}{x_j}} \right\}$$

$$= C_3 \Big/ \left\{ \frac{e(g^r, h^{\beta})}{e(g^r, h^{\alpha(\ell - n - 1)})} \cdot \frac{e(g^{\alpha r}, h^{\ell})}{e(g^{\alpha r}, h^{n})} \right\}$$

$$= m \cdot e(g_1 g_2, h)^r \big/ (e(g, h)^{\beta r} \cdot e(g, h)^{\alpha r}) = m.$$

4.2.2 Security

We prove the following theorems with respect to the confidentiality (RCCA-CK security) of our scheme.

Theorem 4. *Assuming the strong unforgeability of one-time signature, our proposed scheme with temporary delegation satisfies second level RCCA-CK security if the 1-wDBDHI problem is hard.*

Proof. We prove that our proposed scheme is second level RCCA-CK secure under the 1-wDBDHI problem. We build an algorithm $\mathcal{B}$ which is, given $(g, A = g^a, h, \hat{A} = h^a, B = g^b, \hat{B} = h^b, T)$, solving the 1-wDBDHI problem using second level RCCA-CK adversary $\mathcal{A}$.

The algorithm $\mathcal{B}$ simulates $\mathcal{A}$'s input and oracles as follows.

Target user and target period: $\mathcal{B}$ randomly chooses $i^* \stackrel{R}{\leftarrow} HU$ and $L^* \stackrel{R}{\leftarrow} \{1, \ldots, L_{max}\}$. The probability that the adversary $\mathcal{A}$ chooses (i^*, L^*) in the guess stage is $\frac{1}{|HU| L_{max}}$ and it is non-negligible since $|HU|$ and L_{max} are polynomially bounded. In the following, we consider the case that the adversary chooses (i^*, L^*) in the guess stage.

Public parameters: $\mathcal{B}$ chooses $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p^*$ and computes $g_1 = g^\alpha, g_2 = g^\beta, h_1 = h^\alpha, h_2 = h^\beta$. $\mathcal{B}$ chooses the key pair $(svk^*, ssk^*) \xleftarrow{R} \mathcal{G}(1^\lambda)$ of the one-time signature scheme. The generator $u, v, \hat{v}, \hat{v}$ are set as $u = g^{\alpha_1}$, $v = g^{-\alpha_1 svk^*} A^{\alpha_2}$, $\hat{u} = h^{\alpha_1}$, $\hat{v} = h^{-\alpha_1 svk^*} \hat{A}^{\alpha_2}$, where $\alpha_1, \alpha_2 \xleftarrow{R} \mathbb{Z}_p^*$.

Public and secret keys: For the target user i^*, $\mathcal{B}$ chooses $x_*, y_*, z_*, w_* \xleftarrow{R} \mathbb{Z}_p^*$ and computes $pk_* = (A^{x_*}, A^{\alpha y_*}, A^{z_*}, A^{\alpha z_*}, g^{-L^*} \cdot A^{w_*}, \hat{A}^{x_*}, \hat{A}^{\alpha y_*}, \hat{A}^{\beta y_*}, h^{-L^*} \cdot \hat{A}^{w_*},) = (g^{ax_*}, {g_1}^{ay_*}, g^{az_*}, {g_1}^{az_*}, g^{-L^*+aw_*}, h^{ax_*}, {h_1}^{ay_*}, {h_2}^{ay_*}, h^{-L^*+aw_*})$. For the honest user $h \in HU \backslash \{i^*\}$, $\mathcal{B}$ chooses $x_h, y_h, z_h, w_h \xleftarrow{R} \mathbb{Z}_p^*$ and computes $pk_h = (A^{x_h}, A^{\alpha y_h}, A^{z_h}, A^{\alpha z_h}, g^{w_h}, \hat{A}^{x_h}, \hat{A}^{\alpha y_h}, \hat{A}^{\beta y_h}, h^{w_h},) = (g^{ax_h}, {g_1}^{ay_h}, g^{az_h}, {g_1}^{az_h}, g^{w_h}, h^{ax_h}, {h_1}^{ay_h}, {h_2}^{ay_h}, h^{w_h})$. Here, the secret keys for i^* and h are $(ax_*, ay_*, az_*, -L^* + aw_*)$ and (ax_h, ay_h, az_h, w_h), respectively. Note that $\mathcal{B}$ does not have to compute these secret keys.

Delegation oracle: For the query (L, pk_i, pk_j) to the delegation oracle, B responds as follows.

- If $i, j \in HU$, $\mathcal{B}$ chooses $\ell, n, \delta_x, \delta_y \xleftarrow{R} \mathbb{Z}_p^*$ and computes $R_{ijL} = (h^{\frac{\ell x_j + \alpha(\ell - n - 1) y_j}{x_i}} (h^L \hat{W}_i)^{\delta_x}$, $h^{\frac{n x_j + \beta y_j}{y_i}} (h^L \hat{W}_i)^{\delta_y}$, $h^{\frac{\ell x_j + \beta y_j}{z_i}}$, $A^{x_i \delta_x}$, $A^{\alpha y_i \delta_y})$.
- If $i \in HU \backslash \{i^*\}$ and $j \in CU$, $sk_i = (ax_i, ay_i, az_i, w_i)$. Then, $\mathcal{B}$ chooses $\ell', n', \delta'_x, \delta_y \xleftarrow{R} \mathbb{Z}_p^*$ and computes $R_{ijL} = ((\hat{X}_j^{\ell'} \hat{Y}_{1j}^{\ell' - n'})^{\frac{1}{x_i}} h^{(L+w_i)\delta'_x}$, $\hat{X}_j^{\frac{n'}{y_i}} h^{(L+w_i)\delta_y}$, $\hat{X}_j^{\frac{\ell'}{z_i}}$, $(\hat{Y}_{1j} \hat{Y}_{2j})^{\frac{1}{L+w_i}} \hat{A}^{x_i \delta'_x}$, $\hat{A}^{\alpha y_i \delta_y})$. This is a correct re-encryption key with the randomness $\ell = a\ell' - \frac{\beta y_j}{x_j}, n = an' - \frac{\beta y_j}{x_j}, \delta_x = \frac{(\alpha+\beta) y_j}{(L+w_i) a x_i} + \delta'_x$, since
  - $R_{ijL1} = h^{\frac{\ell x_j + \alpha(\ell - n - 1) y_j}{a x_i} + (L+w_i)\delta_x} = h^{\frac{(a\ell' - \frac{\beta y_j}{x_j}) x_j + \alpha((a\ell' - \frac{\beta y_j}{x_j}) - (an' - \frac{\beta y_j}{x_j}) - 1) y_j}{a x_i} + (L+w_i)(\frac{(\alpha+\beta) y_j}{(L+w_i) a x_i} + \delta'_x)}$
    $= h^{\frac{a\ell' x_j - \beta y_j + \alpha(a\ell' - an' - 1) y_j}{a x_i} + \frac{(\alpha+\beta) y_j}{a x_i} + (L+w_i)\delta'_x} = h^{\frac{a(\ell' x_j + \alpha(\ell' - n') y_j) - (\alpha+\beta) y_j}{a x_i} + \frac{(\alpha+\beta) y_j}{a x_i} + (L+w_i)\delta'_x}$
    $= h^{\frac{\ell' x_j + \alpha(\ell' - n') y_j}{x_i} + (L+w_i)\delta'_x} = (\hat{X}_j^{\ell'} \hat{Y}_{1j}^{\ell' - n'})^{\frac{1}{x_i}} h^{(L+w_i)\delta'_x}$,
  - $R_{ijL2} = h^{\frac{n x_j + \beta y_j}{a y_i} + (L+w_i)\delta_y} = h^{\frac{(an' - \frac{\beta y_j}{x_j}) x_j + \beta y_j}{a y_i} + (L+w_i)\delta_y} = h^{\frac{an' x_j - \beta y_j + \beta y_j}{a y_i} + (L+w_i)\delta_y}$
    $= h^{\frac{n' x_j}{y_i} + (L+w_i)\delta_y} = \hat{X}_j^{\frac{n'}{y_i}} h^{(L+w_i)\delta_y}$,
  - $R_{ijL3} = h^{\frac{\ell x_j + \beta y_j}{a z_i}} = h^{\frac{(a\ell' - \frac{\beta y_j}{x_j}) x_j + \beta y_j}{a z_i}} = h^{\frac{a\ell' x_j - \beta y_j + \beta y_j}{a z_i}} = h^{\frac{a\ell' x_j}{a z_i}} = h^{\frac{\ell' x_j}{z_i}} = \hat{X}_j^{\frac{\ell'}{z_i}}$,
  - $R_{ijL4} = h^{(ax_i)\delta_x} = h^{ax_i(\frac{(\alpha+\beta) y_j}{(L+w_i) a x_i} + \delta'_x)} = h^{\frac{(\alpha+\beta) y_j}{L+w_i} + a x_i \delta'_x} = (\hat{Y}_{1j} \hat{Y}_{2j})^{\frac{1}{L+w_i}} \hat{A}^{x_i \delta'_x}$,
  - $R_{ijL5} = h^{\alpha(ay_i)\delta_y} = \hat{A}^{\alpha y_i \delta_y}$.
- If $i = i^*$, $j \in CU$, and $L \neq L^*$, $sk_i = (ax_*, ay_*, az_*, -L^* + aw_*)$. Then, $\mathcal{B}$ chooses $\ell', n', \delta'_x, \delta_y \xleftarrow{R} \mathbb{Z}_p^*$ and computes $R_{*jL} = ((\hat{X}_j^{\ell'} \hat{Y}_{1j}^{\ell' - n'})^{\frac{1}{x_*}} h^{(L-L^*)\delta'_x} (\hat{Y}_{1j} \hat{Y}_{2j})^{\frac{w_*}{(L-L^*) x_*}} \hat{A}^{w_* \delta'_x}$, $\hat{X}_j^{\frac{n'}{y_*}} (h^{L-L^*} \hat{A}^{w_*})^{\delta_y}$, $\hat{X}_j^{\frac{\ell'}{z_*}}$, $(\hat{Y}_{1j} \hat{Y}_{2j})^{\frac{1}{L-L^*}} \hat{A}^{x_* \delta'_x}$, $\hat{A}^{\alpha y_* \delta_y})$. This is a correct re-encryption key with the randomness $\ell = a\ell' - \frac{\beta y_j}{x_j}, n = an' - \frac{\beta y_j}{x_j}, \delta_x = \frac{(\alpha+\beta) y_j}{(L-L^*) a x_*} + \delta'_x$, since
  - $R_{ijL1} = h^{\frac{\ell x_j + \alpha(\ell - n - 1) y_j}{a x_*} + (L - L^* + aw_*)\delta_x}$
    $= h^{\frac{(a\ell' - \frac{\beta y_j}{x_j}) x_j + \alpha((a\ell' - \frac{\beta y_j}{x_j}) - (an' - \frac{\beta y_j}{x_j}) - 1) y_j}{a x_*} + (L - L^* + aw_*)(\frac{(\alpha+\beta) y_j}{(L-L^*) a x_*} + \delta'_x)}$

$$= h^{\frac{a\ell' x_j - \beta y_j + \alpha(a\ell' - an' - 1)y_j}{ax_*} + \frac{(\alpha+\beta)y_j}{ax_*} + (L-L^*)\delta'_x + \frac{w_*(\alpha+\beta)y_j}{(L-L^*)x_*} + aw_*\delta'_x}$$
$$= h^{\frac{a(\ell' x_j + \alpha(\ell' - n')y_j) - (\alpha+\beta)y_j}{ax_*} + \frac{(\alpha+\beta)y_j}{ax_*} + (L-L^*)\delta'_x + \frac{w_*(\alpha+\beta)y_j}{(L-L^*)x_*} + aw_*\delta'_x}$$
$$= h^{\frac{\ell' x_j + \alpha(\ell' - n')y_j}{x_*} + (L-L^*)\delta'_x + \frac{w_*(\alpha+\beta)y_j}{(L-L^*)x_*} + aw_*\delta'_x}$$
$$= (\hat{X}_j^{\ell'} \hat{Y}_{1j}^{\ell'-n'})^{\frac{1}{x_*}} h^{(L-L^*)\delta'_x} (\hat{Y}_{1j}\hat{Y}_{2j})^{\frac{w_*}{(L-L^*)x_*}} \hat{A}^{w_*\delta'_x},$$

– $R_{ijL2} = h^{\frac{nx_j + \beta y_j}{ay_*} + (L-L^*+aw_*)\delta_y} = h^{\frac{(an' - \frac{\beta y_j}{x_j})x_j + \beta y_j}{ay_*} + (L-L^*+aw_*)\delta_y}$
$= h^{\frac{an' x_j - \beta y_j + \beta y_j}{ay_*} + (L-L^*+aw_*)\delta_y} = h^{\frac{n' x_j}{y_*} + (L-L^*+aw_*)\delta_y} = \hat{X}_j^{\frac{n'}{y_*}} (h^{L-L^*}\hat{A}^{w_*})^{\delta_y},$

– $R_{ijL3} = h^{\frac{\ell x_j + \beta y_j}{az_*}} = h^{\frac{(a\ell' - \frac{\beta y_j}{x_j})x_j + \beta y_j}{az_*}} = h^{\frac{a\ell' x_j - \beta y_j + \beta y_j}{az_*}} = h^{\frac{a\ell' x_j}{az_*}} = h^{\frac{\ell' x_j}{z_*}} = \hat{X}_j^{\frac{\ell'}{z_*}},$

– $R_{ijL4} = h^{(ax_*)\delta_x} = h^{ax_*(\frac{(\alpha+\beta)y_j}{(L-L^*)ax_*} + \delta'_x)} = h^{\frac{(\alpha+\beta)y_j}{L-L^*} + ax_*\delta'_x} = (\hat{Y}_{1j}\hat{Y}_{2j})^{\frac{1}{L-L^*}} \hat{A}^{x_*\delta'_x},$

– $R_{ijL5} = h^{\alpha(ay_*)\delta_y} = \hat{A}^{\alpha y_*\delta_y}.$

Re-encryption oracle: For the re-encryption query (L, pk_i, pk_j, C_i), $\mathcal{B}$ checks the validity of C_i by using equations (3). Note that equations (3) are publicly verified. If C_i is ill-formed, $\mathcal{B}$ outputs `invalid`. Otherwise, if either $i \neq i^*$, $j \notin CU$, or $L \neq L^*$, $\mathcal{B}$ uses the re-encryption key and responds the query. If $i = i^*$, $j \in CU$, and $L = L^*$, $C_1 \neq svk^*$ holds with overwhelming probability (because of the strong unforgeability of the one-time signature). Then, the re-encrypted ciphertext C_j can be computed as

$$(C_1, g^{s'}, (g^r)^{s'}, g^{\alpha t'}, (g^r)^{\alpha t'}, g^{k'}, (g^r)^{k'}, g^{\alpha k'}, (g^r)^{\alpha k'}, g^{\pi'}, (g^r)^{\pi'}, C_3, C_4, \hat{C}_4,$$
$$(\hat{X}_j^{\ell} \cdot \hat{Y}_{1j}^{\ell-n-1} \cdot A^{w_* x_* \delta'_x})^{\frac{1}{s'}}, (\hat{X}_j^{n} \cdot \hat{Y}_{2j} \cdot A^{w_* y_* \delta'_y})^{\frac{1}{t'}}, (\hat{X}_j^{\ell} \cdot \hat{Y}_{2j})^{\frac{1}{k'}}, A^{\frac{w_* x_* \delta'_x}{\pi'}}, A^{\frac{\alpha w_* y_* \delta'_y}{\pi'}}, \sigma)$$

where $s', t', k', \pi' \xleftarrow{R} \mathbb{Z}_p^*$. Note that g^r can be computed as $g^r = (C_4 / {C_{2X}}^{\alpha_2/x_*})^{\frac{1}{\alpha_1(C_1 - svk^*)}}$ since $C_4 = (g^{\alpha_1 C_1} g^{-\alpha_1 svk^*} A^{\alpha_2})^r = g^{(C_1 - svk^*)\alpha_1 r} A^{\alpha_2 r}$ and $C_{2X} = {X_*}^r = A^{x_* r}$. This is a valid ciphertext with the randomness $s = s'/ax_*$, $t = t'/ay_*$, $k = k'/az_*$, $\pi = \pi'/aw_*$, $\delta_x = \delta'_x/a$, $\delta_y = \delta'_y/a$.

First level decryption oracle: For the first level decryption query (pk_j, C_j), $\mathcal{B}$ checks the validity of C_j by using the equations (4). Here, the secret key is needed to verify the eighth (final) equation in equations (4). Since the first and second elements of the secret key of the honest user are formed as ax_j and ay_j, respectively, $\mathcal{B}$ checks the following equation instead of the eighth equation in equations (4):

$$\left(\frac{e(C'_{2Z}, C_{5Z}) \cdot e(C'_{2F}, C_{5FX})}{e(C'_{2X}, C_{5X})}\right)^{\frac{1}{y_j}} \cdot \left(\frac{e(C'_{2Z1}, C_{5Z}) \cdot e(C'_{2F}, C_{5FY})}{e(C'_{2Y}, C_{5Y})}\right)^{\frac{1}{x_j}} = e(g_1 g_2, \hat{A}).$$

If C_j is ill-formed, $\mathcal{B}$ outputs `invalid`. When C_j is well-formed (and $j \in HU$), if $C_1 = svk^*$ and $(L, C_3, C_4, \sigma) = (L^*, C_3^*, C_4^*, \sigma^*)$, $\mathcal{B}$ returns $\perp$ since C_j is a Derivative of the challenge ciphertext. In the other case, $C_1 \neq svk^*$ holds with overwhelming probability (because of the strong unforgeability of the one-time signature). Since the first and second elements of the secret key are formed as ax_j and ay_j, respectively, $\mathcal{B}$ computes $X = \left(\frac{e(C''_{2Z}, C_{5Z}) \cdot e(C''_{2F}, C_{5FX})}{e(C''_{2X}, C_{5X})}\right)^{\frac{1}{y_j}} \cdot \left(\frac{e(C''_{2Z1}, C_{5Z}) \cdot e(C''_{2F}, C_{5FY})}{e(C''_{2Y}, C_{5Y})}\right)^{\frac{1}{x_j}} (= e(g_1 g_2, h)^{ar})$, $Y = \{e(C_4, h)^{\alpha+\beta} / X^{\alpha_2}\}^{\frac{1}{(C_1 - svk^*)\alpha_1}} (= e(g_1 g_2, h)^r)$ and $m = C_3 / Y$. Note that if $m \in \{m_0, m_1\}$, $\mathcal{B}$ returns $\perp$.

Challenge ciphertext: The challenge ciphertext is computed as $C^* = (L^*, svk^*, B^{x_*}, B^{\alpha y_*}, B^{z_*}, B^{\alpha z_*}, B^{w_*}, m_{d^*} \cdot T^{\alpha+\beta}, B^{\alpha_2}, \hat{B}^{\alpha_2}, \mathcal{S}(ssk^*, (L^*, C_3^*, C_4^*)))$ where $d^* \stackrel{R}{\leftarrow} \{0,1\}$. If $T = e(g,h)^{b/a}$, C^* is a valid ciphertext with the random exponent $r = b/a$. In contrast, if T is random, $\mathcal{A}$ cannot guess d^* with probability better than 1/2. Therefore, $\mathcal{B}$ decides that $T = e(g,h)^{b/a}$ if d^* equals to the adversary's output and that T is random otherwise. □

Theorem 5. *Assuming the strong unforgeability of one-time signature, our proposed scheme with temporary delegation satisfies first level RCCA-CK security if the 1-wDBDHI problem is hard.*

Proof. The proof is almost the same as that for the second level RCCA-CK security. We can build an algorithm $\mathcal{B}$ which is, given $(g, A = g^a, h, \hat{A} = h^a, B = g^b, \hat{B} = h^b, T)$, solving the 1-wDBDHI problem using first level RCCA-CK adversary $\mathcal{A}$.

The algorithm $\mathcal{B}$ simulates $\mathcal{A}$'s input and oracles as follows. The target user, the target period, and the public parameters are set in the same way as that in the proof of first level RCCA-CK security. The algorithm $\mathcal{B}$ generates the public key for target user i^* as $pk_* = (A^{x_*}, A^{\alpha y_*}, A^{z_*}, A^{\alpha z_*}, g^{w_*}, \hat{A}^{x_*}, \hat{A}^{\alpha y_*}, \hat{A}^{\beta y_*}, h^{w_*},) = (g^{ax_*}, {g_1}^{ay_*}, g^{az_*}, {g_1}^{az_*}, g^{w_*}, h^{ax_*}, {h_1}^{ay_*}, {h_2}^{ay_*}, h^{w_*})$ where $x_*, y_*, z_*, w_* \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Here, the secret keys for the target user is $sk_* = (ax_*, ay_*, az_*, w_*)$. Note that $\mathcal{B}$ does not have to compute the secret keys of the target user to compute the corresponding public keys. The public keys for other honest users $h \in HU\{i^*\}$ are set in the same way as that in the proof of first level RCCA-CK security.

For the delegation query (L, pk_i, pk_j), $\mathcal{B}$ responds in the same way as that for $i \in HU\backslash\{i^*\}$ and $j \in CU$ in the proof of first level RCCA-CK security. $\mathcal{B}$ can respond any first level decryption query in the same way as that in the proof of first level RCCA-CK security.

For the challenge ciphertext C^*, $\mathcal{B}$ chooses $s, t, k, \pi', \delta'_x, \delta'_y \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ $d^* \stackrel{R}{\leftarrow} \{0,1\}$ and computes $(svk^*, A^{\alpha y_* s}, B^{\alpha y_* s}, A^{x_* t}, B^{x_* t}, A^{\alpha y_* k}, B^{\alpha y_* k}, A^{x_* k}, B^{x_* k}, A^{(L^*+w_*)\pi'}, B^{(L^*+w_*)\pi'}, m_{d^*} \cdot T^{\alpha+\beta}, B^{\alpha_2}, \hat{B}^{\alpha_2}, (h^\beta \cdot \hat{A}^{(L^*+w_*)\delta'_y})^{\frac{1}{s}}, (h \cdot \hat{A}^{(L^*+w_*)\delta'_x})^{\frac{1}{t}}, h^{\frac{1+\beta}{k}}, A^{\frac{\alpha y_* \delta'_y}{\pi'}}, A^{\frac{x_* \delta'_x}{\pi'}}, \mathcal{S}(ssk^*, (L^*, C_3^*, C_4^*)))$. If $T = e(g,h)^{b/a}$, C^* is a valid ciphertext with the random exponent $r = b/a$, $\pi = a\pi'$, $\delta_x = a\delta'_x$, $\delta_y = a\delta'_y$. In contrast, if T is random, $\mathcal{A}$ cannot guess d^* with probability better than 1/2. Therefore, $\mathcal{B}$ decides that $T = e(g,h)^{b/a}$ if d^* equals to the adversary's output and that T is random otherwise. □

We show the strong unforgeability of re-encryption keys against collusion attack (sUFReKey-CA) for our proposed scheme with temporary delegation.

Theorem 6. *Our proposed scheme with temporary delegation meets sUFReKey-CA if the 2-DHI problem is hard.*

Proof. We specify the polynomial time algorithm $\mathcal{B}$ which solves the 2-DHI problem by using the polynomial time algorithm $\mathcal{A}$ which breaks the strong unforgeability of re-encryption keys against collusion attack of the proposed scheme described in Section 4. Given $(g, A_1 = g^a, A_2 = g^{a^2}, h, \hat{A}_1 = h^a, \hat{A}_2 = h^{a^2})$, $\mathcal{B}$ runs $\mathcal{A}$ with the following inputs:

Public parameters: The public parameters par are the same as those in the proof of Theorem 3. That is, $\mathcal{B}$ chooses $b, d \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, and sets $g_1 = A_1 \cdot g^{-d} = g^{a-d}$, $g_2 = {g_1}^b = g^{(a-d)b}$, $h_1 = \hat{A}_1 \cdot g^{-d} = h^{a-d}$ and $h_2 = {h_1}^b = h^{(a-d)b}$ (i.e. $\alpha = a - d$, $\beta = (a-d)b$). $\mathcal{B}$ generates $u, v, \hat{u}, \hat{v}$ by following Global-setup.

Public key pk_* for target user: $\mathcal{B}$ chooses $x, y, z \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $pk_* = ((A_2 \cdot {A_1}^{-d})^x, (A_2 \cdot {A_1}^{-d})^y, (A_1 \cdot g^{-d})^z, (A_2 \cdot {A_1}^{-2d} \cdot g^{d^2})^z, g^{-L^*} \cdot (A_2 \cdot A_1^{-d})^w (\hat{A}_2 \cdot \hat{A}_1^{-d})^x, (\hat{A}_2 \cdot \hat{A}_1^{-d})^y, (\hat{A}_2 \cdot \hat{A}_1^{-d})^{by}, h^{-L^*} \cdot (\hat{A}_2 \cdot \hat{A}_1^{-d})^w) = (g^{(a-d)ax}, (g^{a-d})^{ay}, g^{(a-d)z}, (g^{a-d})^{(a-d)z}, g^{-L^*+(a-d)aw}, h^{(a-d)ax}, (h^{a-d})^{ay}, (h^{(a-d)b})^{ay},$

$h^{-L^*+(a-d)aw}$). Here, the corresponding secret key of the target honest user is $sk_* = (x_*, y_*, z_*, w_*)$ where $x_* = (a-d)ax, y_* = ay, z_* = (a-d)z, w_* = -L^* + (a-c)aw$. Note that $\mathcal{B}$ does not have to compute sk_*.

Public and secret keys $(pk_c, sk_c), (pk_j, sk_j)$ for malicious users: The secret keys of the corrupt user $sk_c = (x_c, y_c, z_c, w_c)$ and the malicious user $sk_j = (x_j, y_j, z_j, w_j)$ are set as $x_c, y_c, z_c, w_c \xleftarrow{R} \mathbb{Z}_p^*$ and $y_j, z_j, w_j \xleftarrow{R} \mathbb{Z}_p^*$, $x_j = dy_j$. The public keys pk_c and pk_j are computed by following Keygen.

Re-encryption key R_{*cL}: The re-encryption key $R_{*cL} = (R_{*cL1}, R_{*cL2}, R_{*cL3}, R_{*cL4}, R_{*cL5})$ is computed as

$$R_{*cL1} = R_{*c1} \cdot (h^L \hat{W}_*)^{\delta_x},\ R_{*cL2} = R_{*c2} \cdot (h^L \hat{W}_*)^{\delta_y},\ R_{*cL3} = R_{*c3},\ R_{*cL4} = \hat{X}_*^{\delta_x},\ R_{*cL5} = \hat{Y}_{1*}^{\delta_y}$$

where $R_{*c1}, R_{*c2}, R_{*c3}$ are the same as those in the proof of Theorem 3, respectively and $\delta_x, \delta_y \xleftarrow{R} \mathbb{Z}_p^*$. From the proof of Theorem 3, we have $R_{*c1} = (\hat{X}_c^{\ell} \cdot \hat{Y}_{1c}^{\ell-n-1})^{1/x_*}$, $R_{*c2} = (\hat{X}_c^{n} \cdot \hat{Y}_{2c})^{1/y_*}$, $R_{*c3} = (\hat{X}_c^{\ell} \cdot \hat{Y}_{2c})^{1/z_*}$ for some random ℓ and n, it is easy to see that R_{*cL} is a correct re-encryption key.

Re-encryption key R_{*jL} $(L \neq L^*)$: The re-encryption key $R_{*jL} = (R_{*jL1}, R_{*jL2}, R_{*jL3}, R_{*jL4}, R_{*jL5})$ for $L \neq L^*$ is computed as

$$R_{*jL1} = h^{\frac{(\ell'-n')y_i}{x} + (L-L^*)\delta'_x} D^{\frac{w}{x}} E^{w\delta'_x},\ R_{*jL2} = h^{\frac{(n'd+b)y_i}{y} + (L-L^*)\delta'_y} D^{\frac{bdw}{y}} E^{w\delta'_y},\ R_{*jL3} = h^{\frac{(\ell'd+b)y_j}{z}},$$
$$R_{*jL4} = DE^{x\delta'_x},\ R_{*jL5} = D^{bd} E^{y\delta'_y}$$

where $D = (\hat{A}_1 h^{-d})^{\frac{y_j}{L-L^*}} = h^{\frac{(a-d)y_j}{L-L^*}}$, $E = \hat{A}_2 \hat{A}_1^{-d} = h^{(a-d)a}$, and $\ell', n', \delta'_x, \delta'_y \xleftarrow{R} \mathbb{Z}_p^*$. This is a correct re-encryption key with the randomness $\ell = (a-d)\ell', n = an', \delta_x = \frac{y_j}{a(L-L^*)x} + \delta'_x, \delta_y = \frac{bdy_j}{a(L-L^*)y} + \delta'_y$, since

- $R_{*jL1} = h^{\frac{\ell x_j + \alpha(\ell-n-1)y_j}{x_*} + (L+w_*)\delta_x} = h^{\frac{(a-d)\ell' dy_j + (a-d)((a-d)\ell' - an' - 1)y_j}{(a-d)ax} + (L-L^*+(a-d)aw)(\frac{y_j}{a(L-L^*)x} + \delta'_x)}$
  $= h^{\frac{\ell' dy_j + ((a-d)\ell' - an' - 1)y_j}{ax} + (L-L^*+(a-d)aw)(\frac{y_j}{a(L-L^*)x} + \delta'_x)} = h^{\frac{(a\ell' - an' - 1)y_j}{ax} + (L-L^*+(a-d)aw)(\frac{y_j}{a(L-L^*)x} + \delta'_x)}$
  $= h^{\frac{(\ell'-n')y_j}{x} - \frac{y_j}{ax} + \frac{y_j}{ax} + (L-L^*)\delta'_x + \frac{(a-d)wy_j}{(L-L^*)x} + (a-d)aw\delta'_x} = h^{\frac{(\ell'-n')y_j}{x} + (L-L^*)\delta'_x} D^{\frac{w}{x}} E^{w\delta'_x},$

- $R_{*jL2} = h^{\frac{nx_j + \beta y_j}{y_*} + (L+w_*)\delta_y} = h^{\frac{an' dy_j + (a-d)by_j}{ay} + (L-L^*+(a-d)aw)(\frac{bdy_j}{a(L-L^*)y} + \delta'_y)}$
  $= h^{\frac{an' dy_j + abyj - bdy_j}{ay} + (L-L^*+(a-d)aw)(\frac{bdy_j}{a(L-L^*)y} + \delta'_y)}$
  $= h^{\frac{(n'd+b)y_j}{y} - \frac{bdy_j}{ay} + \frac{bdy_j}{ay} + (L-L^*)\delta'_y + \frac{(a-d)wbdy_j}{(L-L^*)y} + (a-d)aw\delta'_y} = h^{\frac{(n'd+b)y_j}{y} + (L-L^*)\delta'_y} D^{\frac{bdw}{y}} E^{w\delta'_y},$

- $R_{*jL3} = h^{\frac{\ell x_j + \beta y_j}{z_*}} = h^{\frac{(a-d)\ell' dy_j + (a-d)by_j}{(a-d)z}} = h^{\frac{\ell' dy_j + by_j}{z}} = h^{\frac{(\ell'd+b)y_j}{z}},$

- $R_{*jL4} = h^{x_*\delta_x} = h^{(a-d)ax(\frac{y_j}{a(L-L^*)x} + \delta'_x)} = h^{\frac{(a-d)y_j}{L-L^*} + (a-d)ax\delta'_x} = DE^{x\delta'_x},$

- $R_{*jL5} = h^{\alpha y_*\delta_y} = h^{(a-d)ay(\frac{bdy_j}{a(L-L^*)y} + \delta'_y)} = h^{\frac{(a-d)bdy_j}{L-L^*} + (a-d)ay\delta'_y} = D^{bd} E^{y\delta'_y}.$

Then, $\mathcal{B}$ receives $\mathcal{A}$'s output $R^{\dagger}_{*jL^*} = (R_1, R_2, R_3, R_4, R_5)$. Finally, $\mathcal{B}$ outputs

$$W = \left(\frac{{R_3}^z {R_4}^{dw} {R_5}^w}{{R_1}^{dx} \cdot {R_2}^y} \right)^{\frac{1}{(1+b)x_j}}$$

as the answer of the 2-DHI problem.

We show that the algorithm $\mathcal{B}$ outputs $h^{\frac{1}{a}}$ with non-negligible probability. The distributions of the public parameters and the public/secret/re-encryption keys are identical to those of our proposed scheme except when any one of the following events occurs: "$a-d=0$", "$s-d=0$", "$-(a-d)(a\ell'-\frac{1}{s-d})\frac{s+bd}{s}=0$", "$\frac{an'+bd}{s}=0$". Note that $s-d, -(a-d)(a\ell'-\frac{1}{s-d})\frac{s+bd}{s}, \frac{an'+bd}{s}$ are used to generate (simulate) re-encryption key R_{*cL}. It is easy to see that the probability that any one of the above events occurs is negligible. Therefore, the algorithm $\mathcal{A}$ outputs a (forged) re-encryption key $R^{\dagger}_{*jL^*}=(R_1,R_2,R_3,R_4,R_5)$ which satisfies $m=\mathsf{Dec_1}(\mathsf{Reenc}(L^*,R^{\dagger}_{*jL^*},\mathsf{Enc_2}(L,m,pk_*)),sk_j)$ with non-negligible probability. From this equation and the encryption/decryption/re-encryption algorithms of our proposed scheme, we have

$$m = m\cdot e(g_1g_2,h)^r \Big/ \left\{ \left(\frac{e(g^{z_*rk},{R_3}^{1/k})\cdot e(g^{(L^*+w_*)r\pi},{R_4}^{1/\pi})}{e(g^{x_*rs},{R_1}^{1/s})}\right)^{\frac{1}{y_j}} \cdot \left(\frac{e(g^{\alpha z_*rk},{R_3}^{1/k})\cdot e(g^{(L^*+w_*)r\pi},{R_5}^{1/\pi})}{e(g^{\alpha y_*rt},{R_2}^{1/t})}\right)^{\frac{1}{x_j}} \right\}$$
$$\Leftrightarrow m\cdot e\left(g,\frac{{R_3}^{z_*(x_j+\alpha y_j)}{R_4}^{(L^*+w_*)x_j}{R_5}^{(L^*+w_*)y_j}}{{R_1}^{x_*x_j}\cdot{R_2}^{\alpha y_*y_j}}\right)^r = m\cdot e(g,h^{x_jy_j(\alpha+\beta)})^r.$$

Therefore, $\frac{{R_3}^{z_*(x_j+\alpha y_j)}{R_4}^{(L^*+w_*)x_j}{R_5}^{(L^*+w_*)y_j}}{{R_1}^{x_*x_j}\cdot{R_2}^{\alpha y_*y_j}} = h^{x_jy_j(\alpha+\beta)}$. Since we set $x_*=(a-d)ax$, $y_*=ay$, $z_*=(a-d)z$, $w_*=-L^*+(a-d)aw$, $x_j=dy_j$, $\alpha=a-d$, $\beta=(a-d)b$, and the probability that $a-d=0$ or $1+b=0$ is negligible, we have

$$\frac{{R_3}^{(a-d)z(dy_j+(a-d)y_j)}{R_4}^{L^*-L^*+(a-d)awdy_j}{R_5}^{L^*-L^*+(a-d)awy_j}}{{R_1}^{(a-d)axdy_j}\cdot{R_2}^{(a-d)ayy_j}} = h^{x_jy_j(a-d+(a-d)b)}$$

$$\Leftrightarrow \quad \frac{{R_3}^{az}{R_4}^{awd}{R_5}^{aw}}{{R_1}^{axd}\cdot{R_2}^{ay}} = h^{x_j(1+b)} \quad \Leftrightarrow \quad (W=)\ \left(\frac{{R_3}^{z}{R_4}^{dw}{R_5}^{w}}{{R_1}^{dx}\cdot{R_2}^{y}}\right)^{\frac{1}{(1+b)x_j}} = h^{\frac{1}{a}}.$$

Thus, the algorithm $\mathcal{B}$ outputs $h^{\frac{1}{a}}$ with non-negligible probability. □

5 Concluding Remarks

In this paper, we have proposed two PRE schemes. They satisfy the RCCA security and the strong unforgeability of re-encryption keys if the q-wDBDHI problem and the 2-DHI problem are hard.

Note that our schemes can be realized on a symmetric bilinear group $(\mathbb{G},\mathbb{G}_T)$ by setting $g=h\in\mathbb{G}$. In this case, $\hat{u},\hat{v},\hat{X}_i,\hat{Y}_{1i},\hat{W}_i,\hat{C}_4$ are equal to u,v,X_i,Y_{1i},W_i,C_4, respectively. Thus, the sizes of the public key and the ciphertext can be reduced by replacing $\hat{u},\hat{v}$ in the public parameters, $\hat{X}_i,\hat{Y}_{1i},\hat{W}_i$ in the public key, and $\hat{C}_4$ in the ciphertext with u,v,X_i,Y_{1i},W_i,C_4, respectively, and removing $\hat{u},\hat{v},\hat{X}_i,\hat{Y}_{1i},\hat{W}_i,\hat{C}_4$. We can also remove the equation "$e(g,\hat{C}_4)=e(C_4,h)$" from (1)–(4). We can prove that these schemes meet confidentiality and sUFReKey-CA under the assumption that q-wDBDHI problem and 2-DHI problem are hard in $(\mathbb{G},\mathbb{G}_T)$. The proofs are almost the same as those for the asymmetric schemes.

Unfortunately, our schemes do not meet the non-transferability since the attack shown [1] (Section 6) can be applied to our scheme. It might be interesting to construct a non-transferable PRE scheme. Constructing the scheme which meets full CCA security and UFReKey-CA might be one of future works.

References

[1] Hayashi, R., Matsushita, T., Yoshida, T., Fujii, Y., Okada, K.: Unforgeability of Re-Encryption Keys against Collusion Attack in Proxy Re-Encryption. In: IWSEC 2011. Volume 7038 of LNCS. (2011) 210–229

[2] Isshiki, T., Manh Ha, N., Tanaka, K.: Attacks to the Proxy Re-Encryption Schemes from IWSEC2011. In: IWSEC 2013. Volume 8231 of LNCS. (2013) 290–302

[3] Blaze, M., Bleumer, G., Strauss, M.: Divertible Protocols and Atomic Proxy Cryptography. In: EUROCRYPT '98. Volume 1403 of LNCS. (1998) 127–144

[4] Libert, B., Vergnaud, D.: Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption. In: PKC 2008. Volume 4939 of LNCS. (2008) 360–379

[5] Shao, J., Cao, Z.: CCA-Secure Proxy Re-Encryption without Pairings. In: PKC 2009. Volume 5443 of LNCS. (2009) 357–376

[6] Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient Unidirectional Proxy Re-Encryption. In: AFRICACRYPT 2010. Volume 6055 of LNCS. (2010) 316–332

[7] Sur, C., Jung, C.D., Park, Y., Rhee, K.H.: Chosen-Ciphertext Secure Certificateless Proxy Re-Encryption. In: CMS 2010. Volume 6109 of LNCS. (2010) 214–232

[8] Canard, S., Devigne, J., Laguillaumie, F.: Improving the Security of an Efficient Unidirectional Proxy Re-Encryption Scheme. Journal of Internet Services and Information Security **1** (2011) 140–160

[9] Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic Construction of Chosen Ciphertext Secure Proxy Re-Encryption. In: CT-RSA 2012. Volume 7178 of LNCS. (2012) 349–364

[10] Isshiki, T., Nguyen, M.H., Tanaka, K.: Proxy Re-Encryption in a Stronger Security Model Extended from CT-RSA2012. In: CT-RSA 2013. Volume 7779 of LNCS. (2013) 277–292

[11] Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: EUROCRYPT 2005. Volume 3493 of LNCS. (2005) 440–456 Full version is available at https://eprint.iacr.org/2005/015.

[12] Mitsunari, S., Sakai, R., Kasahara, M.: A New Traitor Tracing. IEICE Transactions **E85-A** (2002) 481–484

[13] Boneh, D., Boyen, X.: Efficient Selective-ID Identity Based Encryption without Random Oracles. In: EUROCRYPT 2004. Volume 3027 of LNCS. (2005) 223–238

[14] Dodis, Y., Yampolskiy, A.: A Verifiable Random Function With Short Proofs and Keys. In: PKC 2005. Volume 3386 of LNCS. (2005) 416–431

[15] Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. In: NDSS 2005. (2005)

[16] Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. ACM Transactions on Information and System Security (TISSEC) **9** (2006) 1–30