PRE$^+$: Dual of Proxy Re-encryption and Its Application

Xu An Wang, Yunlong Ge, Xiaoyuan Yang

Key Laboratory of Information and Network Security
Engineering University of Chinese Armed Police Force, Xi'an 710086, P.R. China
wangxazjd@163.com

Abstract. In Eurocrypt'98, Blaze et al. introduced the concept of proxy re-encryption (PRE). It allows a semi-trusted proxy to convert a ciphertext originally intended for Alice into one which can be decrypted by Bob, without the proxy knowing the corresponding plaintext. PRE has found many applications, such as in encrypted e-mail forwarding [8], distributed secure file systems [1,2], multicast [10]cloud computation etc. However, all the PRE schemes until now require the delegator (or the delegator and the delegatee cooperatively) to generate the re-encryption keys. We observe that this is not the only way to generate the re-encryption keys, *the encrypter* also has the ability to generate re-encryption keys. Based on this observation, we introduce a new primitive: PRE$^+$, which is almost the same as the traditional PRE except the re-encryption keys generated by *the encrypter*. Interestingly, this PRE$^+$ can be viewed as the dual of the traditional PRE. Compared with PRE, PRE$^+$ can easily achieve *the non-transferable property and message-level based fine-grained delegation*, while these two properties are very desirable in practical applications. We first categorize PRE$^+$ as the single-hop and multi-hop variant and discuss its potential applications, then we give the definition and security model for the single-hop PRE$^+$, construct a concrete scheme and prove its security. Finally we conclude our paper with many interesting open problems.

1 Introduction

Proxy re-encryption (PRE), allows a semi-trusted proxy to transform a ciphertext originally intended for Alice into the one which can be decrypted by Bob. The proxy in PRE is given a re-encryption key with which he/she can perform the conversion without knowing the corresponding plaintexts. A PRE scheme is *bidirectional* if the proxy can use the re-encryption key to divert ciphertexts from Alice to Bob and vice versa. Otherwise, it is called *unidirectional*. According to how the conversion is performed, PRE can be classified as *multi-hop* and *single-hop*. A multi-hop PRE scheme supports "re-encryption chain", namely any ciphertexts converted by the proxy during the re-encryption phase can be re-encrypted to the ciphertexts of someone else. In other words, the re-encryption phase in multi-hop PRE can be conducted as many times as needed. This differs from a single-hop PRE scheme, where ciphertexts converted by the proxy cannot be re-encrypted again.

The first PRE scheme was proposed by Blaze et al. in Eurocrypt'98 [7]. But due to some undesirable properties of this scheme such as *bidirectional* and *collusion-unresistant* (the delegatee and the proxy can collude to get the delegator's secret key), it does not receive many attention from researchers. In NDSS'05, Ateniese et al. proposed a few new unidirectional PRE schemes, discussed nine desirable properties for PRE, that is, *unidirectional, non-interactive, proxy invisibility, original-access, key optimal, collusion-safe, temporary, non-transitive, non-transferable*, finally explored its many potential applications especially in distributed secure file systems [1,2]. Since then, PRE attracts more and more attention from the researchers. Many wonderful results about PRE have been achieved. From the point view of theoretic construction, many different kinds of PRE have been proposed, such as CCA-secure PRE [8,13,28,31,32,45,44,33], PRE in the

identity based setting [11,16,30], PRE in the attribute based setting [27], PRE in the broadcast setting [12], conditional PRE or type-based PRE [41,12,42,35,21], key private PRE [3], PRE with keyword search [34] etc. From the point view of practical application, PRE has found many interesting applications, such as in encrypted e-mail forwarding [8], key escrow [14], distributed secure file systems [1,2], security in publish/subscribe systems [25], multicast [10], secure certified email mailing lists [23,24], interoperable architecture of DRM [36], access control [37], privacy for public transformation [17], and securely obfuscating re-encryption [19] etc.

1.1 Our Observation

We observe that all these PRE schemes require the delegator (or the delegator and the delegatee cooperatively) to generate the re-encryption keys. However this is not the only way to generate the re-encryption keys. *The encrypter* also has this ability, after all, it has the full control on his plaintext and its corresponding ciphertext. We exploit the possibility of constructing such a PRE scheme, we denote this kind of PRE as PRE^+.

Intuitively, the PRE^+ is the dual of the traditional PRE. In the traditional PRE, the receiver of the ciphertexts (delegator) delegates the decryption right to the delegatee. While in the new PRE^+, encrypter-the sender of the ciphertexts delegates the decryption right to the delegatee. Naturally PRE^+ can be categorized as the *single-hop* and *multi-hop* variants. At first sight, this PRE^+ seems to be no useful for practical applications. The encrypter can directly encrypt the message to the delegatee, why he needs this PRE^+ to delegate the decryption right? The answer lies in the following facts: First, this delegation can improve the computation and communication efficiency greatly compared with the twice encryption method. Secondly, this PRE^+ can solve some long standing open problems in PRE like delegation transferably and fine-grained delegation. Thirdly, considering the multi-hop variants of PRE^+, it runs like broadcast encryption except it relies on some semi-trusted proxies. And these semi-trusted proxies can be easily found in modern networks. Thus it can be well suited for some particular applications. At last, PRE^+ schemes also have many interesting applications like secure cloud computation and multicast etc.

PRE^+ vs. PRE Compared with the traditional PRE, PRE^+ has the following advantages:

- *On fine-grained delegation.* For PRE, all the schemes until now can only achieve time-based (such as [1,2,28,16]) or condition-based (such as [41,12,42,35,21]) fine-grained delegation. But this situation is not so satisfied, for the proxy in these schemes can still re-encrypt the delegator's ciphertext *infinitely* times if and only if it is in the valid time or with the valid condition. Many practical applications *do not* need this property, they prefer the delegator can control on which message the proxy can re-encrypt. But for PRE^+, this problem can be easily solved. In this PRE^+, the value that can control the delegation is the encrypter's ephemeral randomness. If we follow this principle, then the encrypter needs to use the same ephemeral randomness for the delegated messages. If the encrypter wants to allow the proxy to re-encrypt the message, it encrypts the message with the "same" ephemeral randomness. Otherwise it encrypts the message with a "different" ephemeral randomness. Thus we can see that the encrypter can easily achieve message-based fine-grained control on the delegation.
- *On non-transferable delegation.* Non-transferable delegation means that, if the delegator $\mathcal{A}$ delegates the re-encryption right for the delegatee $\mathcal{B}$ to the proxy $\mathcal{P}$, $\mathcal{P}$ and $\mathcal{B}$ can not collude

to delegate, representing $\mathcal{A}$, new re-encryption right for any other users. For PRE, all the schemes until now can not solve this problem naturally. For example, in [28], the authors remarked that constructing systems withstanding the *transfer of delegation* attacks remained as open problems. Although some researchers try to solve this open problem [40,39,18] in the identity based setting, but their proposals all heavily rely on the PKG, which is not a very natural solution. But for PRE$^+$, this problem can be solved very naturally. In this PRE$^+$, the encrypter computes the re-encryption key by using his ephemeral randomness for the messages and the delegatee's public key. The normal ciphertext receiver $\mathcal{A}$ will now be independent with the delegatee $\mathcal{B}$ and the proxy $\mathcal{P}$, now if $\mathcal{B}$ and $\mathcal{P}$ can not collude to derive the ephemeral randomness, it will be non-transferable.

PRE$^+$ vs. Broadcast Encryption The multi-hop PRE$^+$ runs likely the broadcast encryption (BE) , for both primitives can be viewed as the encryption primitive dealing with one encrypter but many receivers. Obviously, multi-hop PRE$^+$ relies heavily on the physical objects, the semi-trusted proxies. But we remark that PRE$^+$ has its own advantages compared with BE for some particular applications.

- *On scalability.* Generically speaking, BE is a primitive tailored for the scenario of one encrypter encrypting messages to many receivers. When designing the BE schemes, researchers focus on its efficiency including the communication cost such as the ciphertext length, public/private key size and the computation cost, while pay little attention on its scalability. When a new user joins or an old user leaves the receiver set, in some cases, the encrypter has to compute and send the ciphertext again for the whole receiver set or the whole system setup has to be rebooted. For multi-hop PRE$^+$, the scalability issue can be solved smoothly. If a new user joins or an old user leaves the receiver set, what need to do is the delegator (encrypter) adjust the new user's nearest proxy's re-encryption key, which is a very easy task.
- *On efficiency.* In current BE schemes, researchers always have to make some trade-off between size of the ciphertext and public/private keys, for example, The best known fully collusion resistant BE scheme is the BGW schemes, which were proposed by Boneh, Gentry and Waters in Eurocrypt'05 [5]. One of the scheme can achieve $\mathcal{O}(\sqrt{n})$ -size ciphertexts and public key, the other can achieve constant size ciphertext, $\mathcal{O}(n)$-size public key and constant size private keys. While in the PRE$^+$ scheme, the constant size of ciphertext and public/private keys can be easily achieved.

1.2 Organization

In Section 2, we give some preliminaries which are necessary for understanding our construction. In Section 3, we give the definition and security notions for PRE$^+$. In Section 4, we give a concrete single-hop PRE$^+$ scheme and prove its security, extends it to a multi-hop variants. At last, We conclude our paper with some interesting open problems 6.

2 Preliminaries

2.1 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of order p for some large prime p. A bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ between these two groups satisfies the following properties:

- Bilinear: We say that a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is bilinear if $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$.
- Non-degenerate: The map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_T$. We observe that $\mathbb{G}$, $\mathbb{G}$ are groups of prime order, this implies that if P is a generator of $\mathbb{G}$ then $e(P, P)$ is a generator of $\mathbb{G}_T$.
- Computable: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in \mathbb{G}$.

We denote BSetup as an algorithm that, on input the security parameter 1^k, it outputs the parameters for a bilinear map as $(p, g, \mathbb{G}, \mathbb{G}_T, e)$, where $p \in \Theta(2^k)$.

2.2 Complexity Assumptions

Let $(p, g, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear group of prime order p and P be its generator. Here we define the extended decisional bilinear Diffie-Hellman (eDBDH) assumption, extended decisional linear (eDL) assumption and variant of discrete logarithm (vDL) assumption.

eDBDH Problem. Let $(p, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathsf{BSetup}(1^k)$ The eDBDH problem is as follows: given given $(g, g^a, g^{ab}, g^{cd/b}, g^{cde/b}, g^{cf}, g^{b/e}, g^{ued/b}g^{sf}, T)$ or random. for some $a, b, c, d, e \in Z_p^*$ and $T \in \mathbb{G}_T$, decide whether $T = e(g, g)^{ad}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving eDBDH problem if

$$|Pr[\mathcal{A}(g, g^a, g^{ab}, g^{cd/b}, g^{cde/b}, g^{cf}, g^{b/e}, g^{ued/b}g^{sf}, e(g, g)^{ad}) = 0]$$
$$-Pr[\mathcal{A}(g, g^a, g^{ab}, g^{cd/b}, g^{cde/b}, g^{cf}, g^{b/e}, g^{ued/b}g^{sf}, T) = 0]| \geq \epsilon$$

where the probability is over the random choices of a, b, c, d, e, u, f, s in Z_p^*, the random choices of T in $\mathbb{G}_T$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}$.

Definition 1. (eDBDH Assumption) *We say that the ϵ-eDBDH assumption holds if no PPT algorithm has advantage at least ϵ in solving the eDBDH problem.*

eDL Problem. Let $(p, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow BSetup(1^k)$. The eDL problem is as follows: Given $(g, g^{\alpha_2}, g^{\alpha_1\alpha_2\alpha_4}, g^{\alpha_3}, g^{\alpha_1\alpha_3\alpha_5}, g^{\alpha_1\alpha_4 u})$, decide whether $T = g^{\alpha_5 - \alpha_4}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving eDL problem if

$$|Pr[\mathcal{A}(g, g^{\alpha_2}, g^{\alpha_1\alpha_2\alpha_4}, g^{\alpha_3}, g^{\alpha_1\alpha_3\alpha_5}, g^{\alpha_1\alpha_4 u}, g^{\alpha_5 - \alpha_4}) = 0]$$
$$-Pr[\mathcal{A}(g, g^{\alpha_2}, g^{\alpha_1\alpha_2\alpha_4}, g^{\alpha_3}, g^{\alpha_1\alpha_3\alpha_5}, g^{\alpha_1\alpha_4 u}, T) = 0]| \geq \epsilon$$

where the probability is over the random choices of $\alpha_1, \alpha_2, \cdots, \alpha_5, u$ in Z_p^*, the random choices of T in $\mathbb{G}_T$, the random choice of $g \in \mathbb{G}$, and the random bits of $\mathcal{A}$.

Definition 2. (eDL Assumption) *We say that the ϵ-eDL assumption holds if no PPT algorithm has advantage at least ϵ in solving the eDL problem.*

vDL Problem. Let $(p, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow BSetup(1^k)$. The vDL problem is as follows: Given $(g^{\beta_1}, g^{u\beta_2/\beta_1}, g^{\beta_2-\beta_3}, g^{\beta_2/\beta_1})$, compute (β_2, β_1), An algorithm $\mathcal{A}$ has advantage ϵ in solving vDL problem if

$$|Pr[\mathcal{A}(g^{\beta_1}, g^{u\beta_2/\beta_1}, g^{\beta_2-\beta_3}, g^{\beta_2/\beta_1}) = (\beta_2, \beta_1)]| \geq \epsilon$$

where the probability is over the random choices of $\beta_1, \beta_2, \beta_3, u$ in Z_p^*, the random choice of $g \in \mathbb{G}$, and the random bits of $\mathcal{A}$.

Definition 3. (vDL Assumption) *We say that the ϵ-vDL assumption holds if no PPT algorithm has advantage at least ϵ in solving the vDL problem.*

3 Definition and Security Model for PRE$^+$

3.1 Definition

Definition 4. *PRE$^+$ (Dual of proxy re-encryption) scheme is a tuple of algorithms* PRE$^+$.KeyGen, PRE$^+$.ReKeyGen, PRE$^+$.Enc, PRE$^+$.ReEnc, PRE$^+$.Dec)

PRE$^+$.KeyGen$(1^k) \rightarrow (pk, sk)$: On input the security parameter 1^k, this algorithm outputs the decrypter's public key/secret key pair (pk_1, sk_1) and the delegatee's public/secret key pair (pk_2, sk_2). Here we emphasis that in PRE$^+$, the delegator is the encrypter, while in traditional PRE the delegator is the decrypter (ciphertext receiver).

PRE$^+$.Enc$(pk_1, m) \rightarrow C_2$: On input a public key pk_1, ephemeral randomness consisting of (r, r') [1], a message $m \in \{0,1\}^n$, this algorithm (the encrypter) outputs a second-level ciphertext C_2.

PRE$^+$.ReKeyGen$(r', pk_1, pk_2) \rightarrow rk_{1\rightarrow 2}$: On input ephemeral randomness r' for pk_1, public key pk_2, this algorithm (the encrypter) outputs a re-encryption key $rk_{1\rightarrow 2}$.

PRE$^+$.ReEnc$(rk_{1\rightarrow 2}, C_2) \rightarrow C_1$: On input a re-encryption key $rk_{1\rightarrow 2}$ and a ciphertext C_2, this algorithm (the proxy) outputs a first-level ciphertext C_1 or the error symbol $\perp$.

PRE$^+$.Dec$_2(sk_2, C_2)$: On input a secret key sk_2 and a second-level ciphertext C_2, this algorithm (the decrypter) outputs a message $m \in \{0,1\}^n$ or $\perp$.

PRE$^+$.Dec$_1(sk_1, C_1)$: On input a secret key sk_1 and a first-level ciphertext C_1, this algorithm (the delegatee) outputs a message $m \in \{0,1\}^n$ or $\perp$.

Roughly speaking, the correctness requires that, for all $(pk_i, sk_i) \leftarrow$ PRE$^+$.KeyGen(1^k) and $(pk_j, sk_j) \leftarrow$ PRE$^+$.KeyGen(1^k), it holds that

$$\text{PRE}^+.\text{Dec}_2(sk_i, \text{PRE}^+.\text{Enc}(pk_i, m)) = m.$$
$$\text{PRE}^+.\text{Dec}_1(sk_j, \text{PRE}^+.\text{ReEnc}(\text{PRE}^+.\text{ReKeyGen}(r', pk_1, pk_2), \text{PRE}^+.\text{Enc}(pk_i, m)) = m.$$

[1] Here we denote the randomness as (r, r'), but actually they can be consisted of many value like (r, u, t) and (r', u', t')

3.2 Security Models

Definition 5. (CCA-security for the Second-level Ciphertext). *A single-use unidirectional* PRE^+ *scheme is CCA-secure for the second-level ciphertext if the advantage of any PPT adversary* $\mathcal{A}$ *in the following game played between a challenger* $\mathcal{C}$ *and* $\mathcal{A}$ *is negligible in the security parameter* k*. Note that we work in the static corruption model, where the adversary should decide the corrupted users before the game starts.*

Setup: The Challenger sets up the system parameters.

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

- $\mathcal{O}_{pk}$: On input an index i, the Challenger takes a security parameter k, and responds by running algorithm $\mathsf{PRE}^+.\mathsf{KeyGen}(1^k)$ to generate a key pair (pk_i, sk_i), gives pk_i to $\mathcal{A}$ and records (pk_i, sk_i) in table T_K.
- $\mathcal{O}_{sk}$: On input pk_i by $\mathcal{A}$, where pk_i is from $\mathcal{O}_{pk}$, if pk_i is corrupted, the Challenger searches pk_i in table T_K and returns sk_i; otherwise, the Challenger returns $\perp$.
- $\mathcal{O}_{rk}$: On input (pk_i, pk_j) by $\mathcal{A}$, where pk_i, pk_j are from $\mathcal{O}_{pk}$, the Challenger returns the re-encryption key $rk_{pk_i \to pk_j} = \mathsf{PRE}^+.\mathsf{ReKeyGen}(r', pk_i, pk_j)$, where sk_i is the secret key corresponding to pk_i.
- $\mathcal{O}_{re}$: On input (pk_i, pk_j, C) by $\mathcal{A}$, where pk_i, pk_j are from $\mathcal{O}_{pk}$, the Challenger returns the re-encrypted ciphertext $C' = \mathsf{PRE}^+.\mathsf{ReEnc}(\mathsf{PRE}^+.\mathsf{ReKeyGen}(r', pk_i, pk_j), C)$, where sk is the secret key corresponding to pk.
- $\mathcal{O}_{dec}$: On input (pk_i, C_i), where pk_i is from $\mathcal{O}_{pk}$, the Challenger returns $\mathsf{PRE}^+.\mathsf{Dec}_1(sk_i, C_i)$ or $\mathsf{PRE}^+.\mathsf{Dec}_2(sk_i, C_i)$, where sk_i is the secret key corresponding to pk_i.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, and a public key pk^* on which it wishes to challenge. There are three constraints on the public key pk^*, (i) it is from $\mathcal{O}_{pk}$; (ii) it is uncorrupted; (iii) if $(pk^*, \bigstar)$ did appear in any query to $\mathcal{O}_{rk}$, then $\bigstar$ is uncorrupted. The Challenger picks a random bit $\mathbf{b} \in \{0, 1\}$ and sets

$$C^* = \mathsf{PRE}^+.\mathsf{Enc}(pk^*, m_{\mathbf{b}})$$

It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ adaptively issues more queries $q_{n_1+1}, \cdots, q_n$, but with the following restrictions.

- $\mathcal{O}_{rk}$: On input (pk_i, pk_j) by $\mathcal{A}$, the following requirements should be all satisfied.
  - pk_i and pk_j are from $\mathcal{O}_{pk}$;
  - if $pk_i = pk^*$, then pk_j is uncorrupted.
- $\mathcal{O}_{re}$: On input (pk_i, pk_j, C_i) by $\mathcal{A}$, the following requirements should be all satisfied.
  - pk_i and pk_j are from $\mathcal{O}_{pk}$;
  - if (pk_i, C_i) is a derivative, the Challenger outputs $\perp$. Otherwise the challenger returns $C' = \mathsf{PRE}^+.\mathsf{ReEnc}(\mathsf{PRE}^+.\mathsf{ReKeyGen}(r', pk_i, pk_j), C_i)$. Derivatives of (pk^*, C^*) are defined as follows:
    1. (pk^*, C^*) is a derivative of itself.
    2. If (pk, C) is a derivative of (pk^*, C^*) and (pk', C') is a derivative of (pk, C), then (pk', C') is a derivative of (pk^*, C^*).
    3. If $\mathcal{A}$ has queried $\mathcal{O}_{re}$ on input (pk, pk', C) and obtained (pk', C'), then (pk', C') is a derivative of (pk, C).

4. If $\mathcal{A}$ has queried $\mathcal{O}_{rk}$ on input (pk, pk'), and $C' = \mathsf{PRE}^+.\mathsf{ReEnc}(\mathcal{O}_{re}(r', pk, pk'), C)$, then (pk', C') is a derivative of (pk, C).

– $\mathcal{O}_{dec}$: On input (pk_i, C_i), if the following requirements are all satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger outputs $\perp$.
  • pk_i is from $\mathcal{O}_{pk}$;
  • (pk_i, C_i) is not a derivative of (pk^*, C^*).

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

We define adversary $\mathcal{A}$'s advantage in attacking PRE^+ as

$$Adv_{PRE^+}^{2CCA} = |\Pr[\mathbf{b} = \mathbf{b}'] - \frac{1}{2}|.$$

Definition 6. (CCA-security for the First-level Ciphertext.) *A single-use unidirectional* PRE^+ *scheme is CCA-secure for the first-level ciphertext if the advantage of any PPT adversary* $\mathcal{A}$ *in the following game played between a challenger* $\mathcal{C}$ *and* $\mathcal{A}$ *is negligible in the security parameter* k*. Note that we work in the static corruption model, where the adversary should decide the corrupted users before the game starts.*

Setup: The Challenger sets up the system parameters.

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

– $\mathcal{O}_{pk}$: Same as the above definition.
– $\mathcal{O}_{sk}$: Same as the above definition.
– $\mathcal{O}_{rk}$: Same as the above definition.
– $\mathcal{O}_{re}$: Same as the above definition.
– $\mathcal{O}_{dec}$: On input (pk_i, C_i), where pk_i is from $\mathcal{O}_{pk}$, the Challenger returns $\mathsf{PRE}^+.\mathsf{Dec}_1(sk_i, C_i)$, where sk_i is the secret key corresponding to pk_i.

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, and two public keys pk^*, pk on which it wishes to challenge. There are two constraints on the public key pk^*, (i) it is from $\mathcal{O}_{pk}$; (ii) it is uncorrupted. The Challenger picks a random bit $\mathbf{b} \in \{0, 1\}$ and sets

$$C^* = \mathsf{PRE}^+.\mathsf{ReEnc}(\mathsf{PRE}^+.\mathsf{ReKeyGen}(r', pk, pk^*), \mathsf{PRE}^+.\mathsf{Enc}(pk, m_{\mathbf{b}}))$$

It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ adaptively issues more queries $q_{n_1+1}, \cdots, q_n$, but with the following restrictions.

– $\mathcal{O}_{rk}$: On input (pk_i, pk_j) by $\mathcal{A}$, the following requirements should be all satisfied.
  • pk_i and pk_j are from $\mathcal{O}_{pk}$;
– $\mathcal{O}_{dec}$: On input (pk_i, C_i), if the following requirement is satisfied, the Challenger responds as in Phase 1; otherwise, the Challenger outputs $\perp$.
  • pk_i is from $\mathcal{O}_{pk}$;

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

We define adversary $\mathcal{A}$'s advantage in attacking PRE^+ as

$$Adv_{PRE^+}^{1CCA} = |\Pr[\mathbf{b} = \mathbf{b}' - \frac{1}{2}|.$$

Remark 1. In the first-level ciphertext security game, the challenge ciphertext is a re-encrypted ciphertext. Note that, in a single-hop unidirectional PRE^+ scheme, since the first level ciphertext cannot be re-encrypted, the adversary should be allowed to obtain any re-encryption keys, even including those from the target public key pk^* to other public keys which are corrupted. Furthermore, since $\mathcal{A}$ is allowed to obtain any re-encryption keys, there is no need to provide the re-encryption oracle and the second level decryption oracle for him.

Definition 7. (Collusion Resistant Security.) *A single-use unidirectional* PRE^+ *is collusion resistant under an adaptive chosen ciphertext attack if no polynomial bounded adversary* $\mathcal{A}$ *has a non-negligible advantage against the Challenger in the following game. Here, we also work in the static corruption model.*

Setup: The Challenger sets up the system parameters.

Find: Almost the same as Phase 1 in the above game, except that there is no re-encryption oracle in this game, since the adversary can get every re-encryption key.

Output: Finally, the adversary $\mathcal{A}$ outputs the randomness which used to generate re-encryption key (According to the above algorithm notation, we can denote it as r'), and wins the game if the corresponding public key pk is uncorrupted.

We define adversary $\mathcal{A}$'s advantage in attacking PRE^+ as

$$Adv^{CR}_{PRE^+} = \Pr[\mathcal{A} \text{ wins}].$$

4 Our Construction

4.1 The Proposed Single-hop PRE$^+$ Scheme

Our construction is based on [22] and the following is our concrete scheme:

1. $\mathsf{Setup}(k)$: On input the security parameter k, the setup algorithm outputs the system parameter $(g, p, \mathbb{G}, \mathbb{G}_T, e) \rightarrow \mathsf{BSetup}(k)$, hash functions H_1, H_2, H_3 and a random generator $g' \in \mathbb{G}$, where $H_1(\cdot) : \{0,1\}^* \rightarrow Z_p^*, , H_2(\cdot) : \mathbb{G}_T \rightarrow Z_p^*, H_3(\cdot) : \mathbb{G}^2 \times \mathbb{G}_T \times \{0,1\}^l \times \mathbb{G}^4 \rightarrow \mathbb{G}$.
2. $\mathsf{KeyGen}(k)$: On input the security parameter k, the key generation algorithm outputs a (public key, private key) pair $(pk_1 = g^{x_1}, sk_1 = x_1)$ for the decrypter, a (public key, private key) pair $(pk_2 = g^{x_2}, sk_2 = x_2)$ for the delegatee where $x_1, x_2 \in Z_p^*$.
3. $\mathsf{Encrypt}(m, pk_1)$: On input a message $m \in \{0,1\}^l$, fixed randomness (t, r_1) and unfixed randomness (r, k_1, π) from Z_p^*, and a public key pk_1, the regular encryption algorithm outputs $C = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9)$ where

$$\begin{aligned} &r = H_1(m, h), c_1 = g^r, c_2 = g^{tr}, c_3 = h \cdot e(g,g)^{rr_1}, \\ &c_4 = m \oplus H_2(h), c_5 = pk_1^{\frac{r_1}{t}}, c_6 = pk_1^{\frac{k_1 r_1}{t}}, c_7 = pk_1^{\pi}, \\ &c_8 = g^{\frac{t}{k_1}}, c_9 = g'^{\frac{k_1 r_1}{t}} H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_1)^{\pi} \end{aligned}$$

   where h is a random element from $\mathbb{G}_T$.
4. $\mathsf{ReKeyGen}(pk_2, \mathsf{Randomness}\ (r_1, t), pk_1)$: The encrypter gives the proxy the re-encryption key

$$rk_{pk_1 \rightarrow pk_2} = (pk_2^{\frac{r_2}{t}}, g^{r_2} g^{-r_1}, g'^{\frac{r_2}{t}}, g^t)$$

   where r_2 is a random number from Z_p^*.

5. $\mathsf{ReEnc}(C, rk_{pk_1 \to pk2})$: On input the ciphertext C, the proxy outputs $C' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7, c'_8, c'_9)$:

$$c'_1 = c_1, c'_2 = c_2, c'_3 = c_3 \cdot e(c_1, g^{r_2} g^{-r_1}) = h \cdot e(g, g)^{r r_2},$$
$$c'_4 = c_4, c'_5 = pk_2^{\frac{r_2}{t}}, c'_6 = pk_2^{\frac{k_2 r_2}{t}}, c'_7 = pk_2^{\pi'}, c'_8 = g^{\frac{t}{k_2}},$$
$$c'_9 = g'^{\frac{k_2 r_2}{t}} H_3(c'_1||c'_2||c'_3||c'_4||c'_5||c'_6||c'_7||c'_8||pk_2)^{\pi'}$$

   where π', k_2. are randomly chosen from Z_p^*.
6. $\mathsf{Decrypt}(C, sk)$: it runs as follows:
   - On input a second-level ciphertext C, and a secret key sk_1, the decryption algorithm is as follows:
     - First check whether

$$e(c_9, pk_1) = e(g', c_6) e(H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_1), c_7)$$

       if it does not hold, output $\perp$ and terminate, otherwise do the following.
     - Compute $h = c_3 / e(c_2, c_5)^{\frac{1}{x}}$.
     - Compute $m = c_4 \oplus H_2(h)$.
     - Compute $r = H_1(m, h)$.
     - Output m, if $c_1 = g^r$, $\perp$ otherwise.
   - On input a first-level ciphertext C', and a secret key sk_2, the decryption algorithm decrypt as the above, for these two level ciphertexts share the same structure.

Remark 2. Roughly speaking, this scheme can be viewed as combining two parts: the BF-IBE part and the Waters signature part. Note that c_4 is computed by the scheme BF [4], and c_6, c_7, c_9 are computed by the strongly unforgeable variant [6] of Waters scheme [38]. Thus our scheme's security roughly can be based on the security of BF-IBE and Waters signature.

Remark 3. By combing randomness (r, r_1, π) and the introduced randomness (k, t), we can instantiate the concept of PRE^+ and prove its security. Why our scheme seems so complicated? After numerous attempts which can be seen in the Appendix 7, we think the current form is necessary. Maybe our scheme can be simplified or more natural schemes in the standard model can be constructed, we leave them as important open problems.

4.2 Security Analysis

Theorem 1. *Our proposal is PRE$^+$-CCA-secure in the random oracle model for the second level ciphertext under assumptions that the eDBDH problem and eDL problem are hard.*

Proof. We incrementally define a sequence of games starting at the real attack (**Game** G_0), and ending up at **Game** G_9, which clearly shows that the adversary cannot break the system. We define E_i to be the event $b = b'$ in Game G_i, where b is the bit involved in the challenge phase, and b' is the output of $\mathcal{A}$ in the Guess phase. Note in our scheme, the two-level ciphertexts share the same structure, thus the decryption oracle for the first level ciphertext and the decryption oracle the second level ciphertext are same, thus in the following Games, we just consider one decryption oracle.

– **Game** G_0**.** This game corresponds to the real attack. By definition,

$$|Pr[E_0] - 1/2| = Adv^{2CCA}_{PRE^+}(k) \tag{1}$$

– **Game** G_1**.** In this game, we modify the hash functions to the random oracles.
  - O_{h_1}: On input a string S, the Challenger checks whether $(S, r^{(1)})$ exists in Table T_{h_1}. If it is, the Challenger returns $r^{(1)}$; otherwise, the challenger chooses a random number $r^{(1)}$ from Z_p^*, returns $r^{(1)}$, and records $(S, r^{(1)})$ into Table T_{h_1}.
  - O_{h_2}: On input a string S, the Challenger checks whether $(S, R^{(2)})$ exists in Table T_{h_2}. If it is, the Challenger returns $R^{(2)}$; otherwise, the Challenger chooses a random number $R^{(2)}$ from $\{0,1\}^l$, returns $R^{(2)}$, and records $(S, R^{(2)})$ into Table T_{h_2}.
  - O_{h_3}: On input a string $c_1||c_2||c_3||c_4||c_5||||c_6||c_7||c_8||pk$, the Challenger checks whether $(c_1||c_2||c_3||c_4||c_5||||c_6||c_7||c_8||pk, R^{(3)}, *)$ exists in Table T_{h_3}. If it is, the Challenger returns $R^{(3)}$; otherwise, the Challenger chooses a random number $r^{(3)}$ from Z_p^*, returns $R^{(3)} = pk^{r^{(3)}}$, and records $(c_1||c_2||c_3||c_4||c_5||||c_6||c_7||c_8||pk, R^{(3)}, r^{(3)})$ in Table T_{h_3}.

– **Game** G_2**.** In this game, we modify O_{h_1} and the challenge phase by setting $h = h^*$, where h is a random number from Z_p^*.
  - O_{h_1}: On input a string $m_b||h^*$, the Challenger aborts. Other performances are the same as those in O_{h_1} which we first mention.
  - Challenge: The Challenger sets $h = h^*$, other performances are the same as those in O_{h_1} which we first mention.
    It is easy to see that only if the adversary never queries O_{h_1} with $m_b||h^*$, **Game** G_2 and **Game** G_1 are indistinguishable. Hence, we have

$$|Pr[E_2 - E_1] \leq \frac{q_{h_1}}{p}$$

    where q_{h_1} is the maximum number of queries to O_{h_1} (including the number the Challenger queries O_{h_1}).

– **Game** G_3**.** In this game, we set $g' = g_{+1}(\neq g) = g^u$, where u is chosen randomly from Z_p^*. The Challenger knows a value $log_g^{g_{-1}} = 1/u$ satisfying $e(g_{-1}, g_{+1}) = e(g, g)$. It is easy to see that **Game** G_3 and **Game** G_2 are indistinguishable.

$$Pr[E_3] = Pr[E_2]$$

– **Game** G_4**.** In this game, we modify the re-encryption oracle and the decryption oracle as follows.
  - O_{re}: On input (pk_i, pk_j, C), if $(pk^*, C^*) = (pk_i, C)$ and pk_j is uncorrupted, the Challenger records $(pk_j, Encrypt(pk_j, O_{dec}(pk_i, C)))$ into Table T_{re}. Other performances are the same as that in Game G_3.
  - O_{dec}: On input (pk_i, C_i), if pk_i is uncorrupted, the Challenger checks whether (pk_i, C_i) exists in Table T_{re}. If yes, the Challenger outputs $\perp$. Other performances are the same as that in Game G_4. Note that Table T_{re} is used to record the derivatives of the second level challenge ciphertext from O_{re}.

  we can easily obtain that

$$Pr[E_4] = Pr[E_3]$$

– **Game** G_5**.** In this game, we modify the decryption oracle as follows.
  - O_{dec}: The Challenger only changes the method of doing the decryption for the uncorrupted public keys as follows, and other performances are the same as that in Game G_4.
    1. Set a set $\mathcal{S} = \phi$.
    2. Find $(S, r^{(1)})$ in Table T_{h_1}, such that $g^{r^{(1)}} = c_1, m'||h' = S, c_4 = m' \oplus H_2(h')$.
    3. If $\mathcal{S} = \phi$, then output $\perp$ and terminate. Otherwise, do the following.
    4. Choose the first pair $(S, r^{(1)})$ from $\mathcal{S}$, and set $\mathcal{S} = \mathcal{S}/(S, r^{(1)})$.
    5. If $h' = c_3/e(c_9/c_7^{r^{(3)}}, c_8)^{r^{(1)}/u}$ and $e(c_9, pk_i) = e(g', c_6)e(H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_i), c_7)$ both hold, output m' and terminate; otherwise, go to Step 3. Note that $r^{(3)}$ is the associated value to $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, pk_i)$ in T_{h_3}, and

$$H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_i) = pk_i^{r^{(3)}}$$

  In Game G_5, the Challenger decrypts the ciphertext not by using the secret key but the random oracles (O_{h_1}, O_{h_2} and O_{h_3}). Hence, Game G_5 and Game G_4 would be different, if the adversary guessed the correct value of $H_1(m||h)$ without querying O_{h_1} or guessed the correct value of $H_2(h)$ without querying O_{h_2} or guessed the correct value of $H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_i)$ without querying O_{h_3}. The probability of this issue is $(q_{h_2}/p + q_{h_3}/2^l + q_{h_3}/p)$ at most. As a result, we have that

$$|Pr[E_5] - Pr[E_4]| \leq q_{dec}(q_{h_2}/p + q_{h_3}/2^l + q_{h_3}/p)$$

  where $q_{h_1}, q_{h_2}, q_{h_3}$ is the maximum number of queries to $O_{h_1}, O_{h_2}, O_{h_3}$ (including the number the Challenger queries $O_{h_1}, O_{h_2}, O_{h_3}$).
– **Game** G_6**.** In this game, we continue to modify the decryption oracle as follows.
  - It is almost the same as that in Game G_5, except that if pk_i is corrupted, and the plaintext is $\perp$ obtained in the previous performances, the Challenger additionally performs the following.
    1. If $e(c_9, pk_i) = e(g', c_6)e(H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_i), c_7)$ holds, do the following steps; otherwise, output $\perp$ and terminate.
    2. Set $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_2'$.
    3. Search $(rk^1_{pk^*\to pk_i}, rk^2_{pk^*\to pk_i}, rk^3_{pk^*\to pk_i}, rk^4_{pk^*\to pk_i})$ in Table T_{rk} (where T_{rk} records the re-encryption keys which have returned to the adversary), and put all $(rk^1_{pk^*\to pk_i}, rk^2_{pk^*\to pk_i}, rk^3_{pk^*\to pk_i}, rk^4_{pk^*\to pk_i})$ into the set $\mathcal{S}_1$.
    4. Search $(S, r^{(1)})$ in Table T_{h_1}, such that $m'||h' = S$, $c_4 = m' \oplus H_2(h')$, and $c_1 = g^{r^{(1)}}$, and then put all $(S, r^{(1)})$ into set $\mathcal{S}_2$ and $\mathcal{S}_2'$.
    5. If $\mathcal{S}_1 = \phi$ or $\mathcal{S}_2 = \phi$, then output $\perp$ and terminate. Otherwise, do the following.
    6. Choose the first pair $(rk^1_{pk^*\to pk_i}, rk^2_{pk^*\to pk_i}, rk^3_{pk^*\to pk_i}, rk^4_{pk^*\to pk_i})$ from $\mathcal{S}_1$, and set $\mathcal{S}_1 = \mathcal{S}_1/(rk^1_{pk^*\to pk_i}, rk^2_{pk^*\to pk_i}, rk^3_{pk^*\to pk_i}, rk^4_{pk^*\to pk_i})$
    7. Choose the first pair $(S, r^{(1)})$ from $\mathcal{S}_2$, and set $\mathcal{S}_2 = \mathcal{S}_2/(S, r^{(1)})$.
    8. If $c_3' = c_3 e(c_9/c_7^{r^{(3)}}, g^{\frac{t}{k_1}})^{\frac{r^{(1)}}{u}}$ where $r^{(3)}$ is the associated value to $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, pk_i)$ in T_{h_3}.
    9. If $\mathcal{S}_1 = \phi$, output $\perp$; otherwise, do the following steps.
    10. If $\mathcal{S}_2 = \phi$, then set $\mathcal{S}_2 = \mathcal{S}_2'$ and go to Step 5; otherwise, go to Step 6.

Note that if the input regular ciphertext is computed from ReEnc, the verifying equation in step 8 must hold. Hence, the modification in this game does not make Game G_6 different from Game G_5. As a result, we have that

$$Pr[E_6] = Pr[E_5]$$

– **Game** G_7. In this game, we continue to modify the re-encryption key generation oracle as follows.
  - O_{rk}: On input (pk_i, pk_j), if $pk_i \neq pk^*$, then the Challenger returns $(pk_j^{r_2/t}, g^{r_2}g^{-r_1}, g'^{r_2/t}, g^t)$ where (r_1, t) are fixed randomness chosen by the challenger in advance, and r_2 is randomly chosen from Z_p^*. Other performances are the same as those in Game G_6.

  It is easy to see that Game G_7 and Game G_6 are indistinguishable. Hence, we have that

$$Pr[E_7] = Pr[E_6]$$

– **Game** G_8. In this game, we modify the re-encryption oracle as follows.
  - O_{rk}: On input (pk_i, pk_j), if $pk_i = pk^*$, the pk_j is uncorrupted, then the Challenger returns (A, B, C, D, pk_j) where A, B, C are random elements from G, and $e(A, g') = e(C, pk_j)$ and D is fixed as g^t. Other performances are the same as those in Game G_7.

  We have the following analysis on the difference between Game G_7 and Game G_8.
  - O_{dec} and O_{re} do not use the secret keys of the uncorrupted users in Game G_7.

  Note that due to the modification in Game G_6, the decryption oracle cannot help to verify the modification on O_{rk}. In Game G_6, it is guaranteed that the first level ciphertexts from $\mathsf{ReEnc}(O_{rk}, C)$ can be decrypted properly.
  Hence, if the adversary without knowing $1/u$ can verify the modification on O_{rk}, we can build an algorithm to solve the eDL hard problem. In particular, set $t = 1/\alpha_1, pk_i = g^{\alpha_2}, r_1 = \alpha_4, pk_i^{r_1/t} = g^{\alpha_1\alpha_2\alpha_4}, pk_j = g^{\alpha_3}, r_2 = \alpha_5, (pk_j)^{r_2/t} = g^{\alpha_1\alpha_3\alpha_5}, (g')^{r_2/t} = g^{\alpha_1\alpha_4 u}$, we can not calculate the value of $g^{r_2}g^{-r_1} = g^{\alpha_5-\alpha_4}$. Here we do not consider the information about (r_2, r_1) leaked from $(c_2, c_3, c_6, c_8, c_9, c_2', c_3', c_6', c_8', c_9')$ for these value all are completely random for randomness $(r, (r, h), k_1, (k_1, \pi), r, (r, h), k_2, (k_2, \pi'))$. Hence we have that

$$|Pr[E_8] - Pr[E_7]| \leq q_{rk} \cdot \epsilon_{eDL}$$

  where q_{rk} is the maximum number of queries to O_{rk}.
– **Game** G_9. In this game, we modify the challenge phase as follows.
  - Challenge: The challenge ciphertext is

$$r = a, c_1^* = g^r = g^a, t = b, c_2^* = g^{tr} = g^{ab}, c_3^* = h^*T, c_4^* = m_b \oplus H_2(h^*), pk_1 = g^c,$$

$$r_1 = d, c_5^* = pk_1^{r_1/t} = g^{cd/b}, k_1 = e, c_6^* = pk_1^{k_1 r_1/t} = g^{cde/b}, \pi = f, c_7 = pk_1^{\pi} = g^{cf},$$

$$c_8 = g^{t/k_1} = g^{b/e}, H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_1) = g^s,$$

$$c_9 = g'^{\frac{k_1 r_1}{t}} H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_1)^{\pi} = g^{ued/b}g^{sf}$$

  Other performances are the same as those in previous game. Note that in the above we implicitly set $H_1(m, h) = a$, and T is a random number. If the adversary can distinguish Game G_9 and Game G_8, we can build an algorithm B that solves the eDBDH problem. The eDBDH

problem is, given $(g^a, g^{ab}, g^{cd/b}, g^{cde/b}, g^{cf}, g^{b/e}, g^{ued/b}g^{sf}, T)$ and decide if $T = e(g,g)^{ad}$ or random. it is clear that Game G_9 is the same as Game G_8. Hence, we have that

$$|Pr[E_9] - Pr[E_8]| \leq \epsilon_{eDBDH}$$

Furthermore, it is clear that in Game G_9

$$Pr[E_9] = 1/2$$

since T is a random element.

As a result, combining the above equations among the Games, we get the theorem.

Theorem 2. *Our proposal is PRE$^+$-CCA-secure in the random oracle model for the first level ciphertext under assumptions that the eDBDH problem and eDL problem are hard.*

We can use the similar method in the proof of the above Theorem 1 to prove this theorem, for the first level ciphertext and the second level ciphertext share the same structure, here we omit the details..

Theorem 3. *Our PRE$^+$ proposal is collusion-resistent under assumption that vDL problem is hard.*

Here when the proxy and the delegatee collude, they can compute $(g^{\frac{r_2}{t}}, g^{r_2}g^{-r_1}, g'^{\frac{r_2}{t}}, g^t)$, but this does not help them compute (r_1, t) under the eDL problem, which is the randomness used to generate re-encryption keys. The vDL problem is, given $(g^{\beta_1}, g^{u\beta_2/\beta_1}, g^{\beta_2-\beta_3}, g^{\beta_2/\beta_1})$, compute (β_2, β_1), which is difficult. Thus our PRE$^+$ proposal is collusion-resistent.

4.3 Extending it to a Multi-hop Variant

Our single-hop PRE$^+$ scheme can be easily extended to a multi-hop variant. Below is the concrete scheme:

1. Setup(1^k): Same as the single-hop scheme.
2. Keygen(1^k): Same as the single-hop scheme.
3. Encrypt(m, pk_{i-1}): Same as the single-hop scheme.
4. ReKeyGen$(pk_i$, Randomness (r_{i-1}, t), $pk_{i-1})$: The encrypter gives the proxy the re-encryption key
$$rk_{pk_i} = (pk_i^{\frac{r_i}{t}}, g^{r_i}g^{-r_{i-1}}, g'^{\frac{r_i}{t}}, g^t)$$
where r_i is a random number from Z_p^*, and the encrypter records r_i for the next hop's re-encryption key generation.
5. ReEnc$(C, rk_{pk_{i-1}\to pk_i})$: Same as the single-hop scheme.
6. Decrypt(C, sk_i): Same as the single-hop scheme.

Remark 4. Note in this scheme the ciphertext remains constant no matter how times it has been re-encrypted. This is a very desirable feature in practical applications. We emphasis the construction of constant-size ciphertext multi-hop CCA-secure traditional PRE scheme remains as a very important open problem until now. We also note the encrypter needs to remember the randomness r_i for the $(i+1)$ hop's re-encryption key generation, which will restrict its employment. We leave construction of new multi-hop PRE$^+$ scheme which need not remember the randomness as an important open problem.

5 Applications

In this section, we describe two typical application of PRE$^+$.

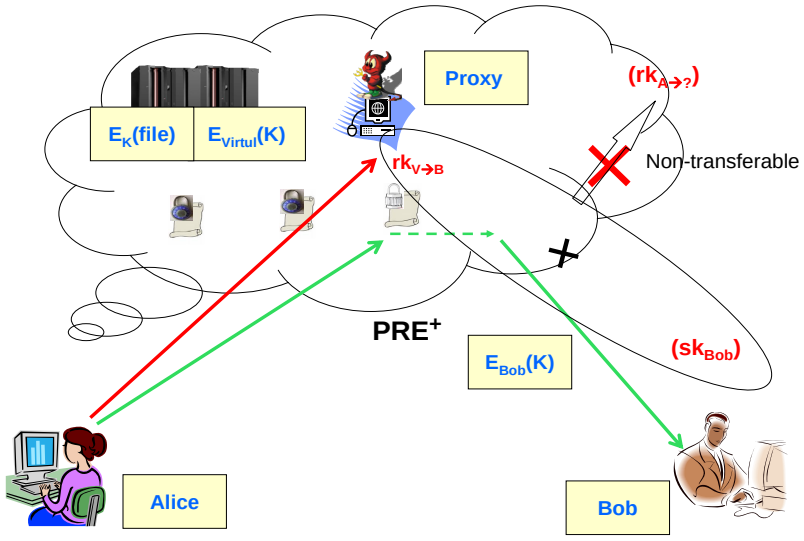

Fig. 1. PRE$^+$ in Secure Cloud Computation

5.1 PRE$^+$ in Secure Cloud Computation

Here we describe a frame of application of PRE$^+$ in secure cloud computation, which can be seen from Fig. 1. In this framework, there are four kinds of parties. The first kind is the client users, denoted as Alice, which provides contents to the Cloud. The second kind is the infrastructure provider, denotes Cloud, which stores and manages client users's providing contents. The third kind is the application users, denoted as Bob, which wants to access Alice's contents. The fourth kind is the access controller, denoted as proxy, which does the transformation process (sometimes the proxy can be the Cloud). In order to protect the privacy, Alice certainly encrypts her contents. Here we assume Alice constructs a virtual user Virtual, which she knows its private/public key. Then Alice encrypts her content under a file symmetrical key K, and encrypts K under the public key of Virtual, sends them to Cloud. Then after knowing Bob's public key, she assigns the re-encryption key $rk_{v\to B}$ to the proxy. When application user Bob wants to access the content of Alice, he relies on the proxy doing the re-encryption from V to B with $rk_{v\to B}$. In this process, the proxy can not know the content, neither he can get any information on private keys of Virtual and Bob.

Readers might argue that the traditional PRE can also do this job. But our point is that, just like the statements in the introduction, our PRE$^+$ framework can handles very well on the **non-transferable and message level based fine-grained delegation issues**. In the current

state, almost there exist no PRE schemes with non-transferable property, and no IBPRE schemes with non-transferable property without the help of PKG [40], while relying on PKG is not a good choice for practical applications. That means, if using PRE in this framework, the proxy and Bob can collude to representing Alice, to make new delegations to new users. Obviously, this contradicts Alice's original intention. Similarly, in the current state, all the PRE schemes can not handle the fine-grained delegation at the **message based level**. This will bring great trouble, Alice has no control on which content will be re-encrypted. She has to incorporate the condition or type or time information in her content if she uses the conditional or type or time based PRE. While in PRE$^+$, Alice can easily control on which content will be re-encrypted and which will be not.

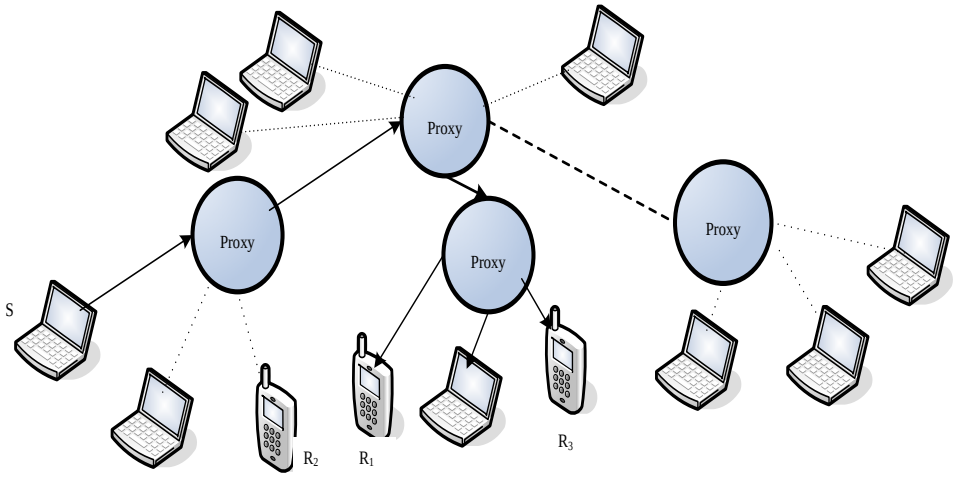

Fig. 2. PRE$^+$ in Secure Multicast

5.2 PRE$^+$ in Secure Wireless Multicast

Here we describe a framework of application of PRE$^+$ in secure wireless multicast. In this framework, the sender encrypts the multicast key to a set of receivers by using PRE$^+$. There are very many natural proxies in the typical wireless network settings, like the routers and switchers. First the sender S encrypts the key to one of the receivers, and the adjacent proxies re-encrypt the ciphertext to their neighbor receivers. And these receivers send the re-encrypted ciphertext to their adjacent proxies, and thus the multicast key will be spread to all the receivers.

Readers might argue that BE and traditional PRE can also do this job. But PRE$^+$ brings some new benefits for this application. First, in the whole process, the ciphertext, public/private keys will remain constant, while the BE can not easily achieve. Second, the scalability can be easily handled in PRE$^+$. If a new user joins the receiver set, the sender S just assigns new proxy re-encryption keys to its adjacent proxies. While in BE, the sender has to compute the whole ciphertexts and send them to the network again, which is a heavy task for wireless setting. While in PRE, the new users's adjacent proxies's re-encryption keys have to be assigned by the old receivers. But sometimes these old receivers are not so friendly, they might refuse to do this or be

malicious. But in PRE^+, the re-encryption keys are all assigned by the sender, and the problems in PRE can not happen. Lastly, there are also existing transferring of delegation problem and fine-grained delegation problem in PRE, while in PRE^+, these problems can be easily handled.

6 Conclusion

In this paper, we introduce a new primitive: PRE^+, which is the dual of PRE, discuss its difference with existing primitives PRE and BE, describe some potential applications of this new primitive. We also give the definition and security model for this new primitive, construct a concrete single-hop scheme and prove its $\mathsf{IND\text{-}CCA}$ security, extend it to a multi-hop scheme.

There are much future work can be explored, such as further finding interesting applications of PRE^+, constructing more kinds of single-hop and multi-hop PRE^+ schemes based on different assumptions, exploring the relationship between PRE^+ and other cryptographic primitives like proxy encryption etc. We leave them as open problems.

7 Acknowledgements

The authors would like to express their gratitude thanks to Dr. Xiaoqi Jia for providing us full vision of [22] and Dr. Jun Shao, Dr. Licheng Wang for many helpful comments. This work is supported by National Natural Science Foundation of China under contract No. 61103230.

References

1. G. Ateniese, K. Fu, M. Green and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *ACM NDSS 2005*, pages 29–43, 2005.
2. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *ACM Transactions on Information and System Security*, no. 1, pages 1–30. 2006.
3. G. Ateniese, K. Benson, and S. Hohenberger. Key private proxy re-encryption. In *CT-RSA 2009*, volume 5473 of *LNCS*, pages 279–294, 2009.
4. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, 2001.
5. D. Boneh, C. Gentry and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275, 2005.
6. D. Boneh, E. Shen, and B. Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *PKC 2006*, volume 3985 of *LNCS*, pages 229-240, 2006.
7. M. Blaze, G. Bleumer and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 127–144, 1998.
8. R. Canetti and S. Hohenberger. Chosen ciphertext secure proxy re-encryption. In *ACM CCS 2007*, pages 185–194, 2007.
9. R. Canetti, S. Halevi and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271, 2003.
10. Y-P.Chiu, C-L. Lei, and C-Y. Huang. Secure multicast using proxy encryption. In *ICICS 2005*, volume 3783 of LNCS, pages 280–290, 2005.
11. C. Chu and W. Tzeng. Identity-based proxy re-encryption without random oracles. In *ISC 2007*, volume 4779 of *LNCS*, pages 189–202, 2007.
12. C. Chu, S. Chow, J.Weng, J. Zhou, and R. H. Deng. Conditional proxy broadcast re-encryption. In *ACISP 2009*, volume 5594 of *LNCS*, pages 327–342, 2009.
13. R. Deng, J. Weng, S. Liu and K. Chen. Chosen ciphertext secure proxy re-encryption without pairing. In *CANS 2008*, volume 5339 of *LNCS*, pages 1–17, 2008.
14. A. Ivan and Y. Dodis. Proxy cryptography revisited. In *Internet Society (ISOC): NDSS 2003*, 2003.

15. L. Fang, W. Susilo and J. Wang. Anonymous conditional proxy re-encryption without random oracles. In *PROVSEC 2009*, volume 5854 of *LNCS*, pages 47–60, 2009.
16. M. Green and G. Ateniese. Identity-based proxy re-encryption. In *ACNS 2007*, volume 4521 of *LNCS*, pages 288–306, 2007.
17. T. S. Heydt-Benjamin, H. Chae, B.Defend, and K. Fu. Privacy for public transportation. In *PET 2006*, volume 4258 of LNCS, pages 1–19, 2005.
18. Y. He, T. Chim, L. Hui and S. Yiu. Non-Transferable Proxy Re-Encryption Scheme for Data Dissemination Control. Available at Cryptology ePrint Archive, Report 2010/192, 2010.
19. S. Hohenberger, G. N. Rothblum, A.Shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. In *TCC 2007*, volume 4392 of LNCS, pages 233–252, 2007.
20. S. Hohenberger, B. Waters. Realizing hash-and-sign signatures under standard assumptions. In *EUROCRYPT 2009*, volume 5479 of LNCS, pages 333–350, 2009.
21. L. Ibraimi, Q. Tang, P. Hartel, and W. Jonker. A type-and-identity-based proxy re-encryption scheme and its application in healthcare. In *SDM 2008*, volume 5159 of LNCS, pages 185–198, 2008.
22. X. Jia, J. Shao, J. Jing and P. Liu. CCA-secure type-based proxy re-encryption with invisible proxy. In *2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010)*, pages 1299–1305.
23. H. Khurana, A. Slagell, and R.Bonilla. Sels: A secure e-mail list service. In *ACM SAC 2005*, pages 306-313, 2005.
24. H. Khurana and H-S.Hahm. Certified mailing lists. In *ASIACCS 2006*, pages 46–58, 2006.
25. H. Khurana and R. Koleva. Scalable security and accounting services for content-based publish subscribe systems. In *International Journal of E-Business Research*, 2(3), 2006.
26. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442, 2004.
27. X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute based proxy re-encryption with delegating capabilities. In *AISACCS 2009*,pages 276–286, 2009.
28. B. Libert and D. Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Pairing 2008*, volume 5209 of *LNCS*, pages 332–353, 2008.
29. B. Libert and D. Vergnaud. Unidirectional chosen ciphertext secure proxy re-encryption. In *PKC 2008*, volume 4939 of *LNCS*, pages 360–379, 2008.
30. T. Matsuo. Proxy re-encryption systems for identity-based encryption. In *Pairing 2007*, volume 4575 of *LNCS*, pages 247–267, 2007.
31. J. Shao and Z. Cao. CCA-secure proxy re-encryption without pairing. In *PKC 2009,* LNCS 5443, pages. 357–376, Springer–Verlag, 2009.
32. J. Shao, Z. Cao, P. Lin. Generic construction for CCA-secure unidirectional proxy re-encryption. In Security and Communication Networks, no. 2, pages 1-16, 2009.
33. J. Shao, Z. Cao and P. Lin. CCA-secure PRE scheme without random oracle. Cryptology ePrint Archive, Report 2010/112, 2010. Available at http://eprint.iacr.org.
34. J. Shao, Z. Cao, X. Liang and H. Lin. Proxy re-encryption with keyword search. In *Information Science 2010*, Doi: 10.1016/j.ins.2010.03.026, 2010.
35. Q. Tang. Type-based proxy re-encryption and its construction. In *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 130–144, 2008.
36. G. Taban, A. A. Cardenas, and V. D. Gligor. Towards a secure and interoperable drm architecture. In *ACM DRM 2006*, pages 69–78, 2006.
37. A. Talmy and O. Dobzinski. Abuse freedom in access control schemes. In *AINA 2006*, pages 77–86, 2006.
38. B. Waters. Efficient identity-based encryption without random oracles. In *Eurocrypt 2005*, volume 3494 of LNCS, pages 114–127, 2005.
39. L. Wang, L. Wang, M. Mambo, E. Okamoto. New identity-based proxy re-encryption schemes to prevent collusion attacks. In *Pairing 2010*, volume 6487 of LNCS, pages 327–346, 2010.
40. X. Wang, X. Yang, F. Li. On the Role of PKG for Proxy Re-encryption in the Identity Based Setting. Available at Cryptology ePrint Archive, Report 2008/410, 2008.
41. J. Weng, R. H. Deng, C. Chu, X. Ding, and J. Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *ACM ASIACCS 2009*, pages 322–332, 2009.
42. J. Weng, Y. Yang, Q. Tang, R. Deng, and F. Bao. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *ISC 2009*, volume 5735 of *LNCS*, pages 151–166, 2008.
43. J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen and F. Bao CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. In Science China Information Sciences, 53:593-606, 2010.

44. J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen and F. Bao CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. Cryptology ePrint Archive, Report 2010/265, 2010. Available at http://eprint.iacr.org.
45. S. Chow, J. Weng, Y. Yang, R. Deng Efficient unidirectional proxy re-encryption. In *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 316–332, 2010.

Appendix

7.1 First Attempt

1. $\mathsf{Setup}(k)$: On input the security parameter k, the setup algorithm outputs the system parameter $(g, p, \mathbb{G}, \mathbb{G}_T, e) \rightarrow \mathsf{BSetup}(k)$, hash functions H_1, H_2, H_3, H_4, H_5 and a random generator $g' \in \mathbb{G}$, where $H_1(\cdot) : \{0,1\}^* \rightarrow Z_p^*, , H_2(\cdot) : \mathbb{G}_T \rightarrow Z_p^*, H_3(\cdot) : \mathbb{G}^2 \times \mathbb{G}_T \times \{0,1\}^l \times \mathbb{G}^2 \rightarrow \mathbb{G}$.
2. $\mathsf{KeyGen}(k)$: On input the security parameter k, the key generation algorithm outputs a (public key, private key) pair $(pk_1 = g^{x_1}, sk_1 = x_1)$ for the delegator, a (public key, private key) pair $(pk_2 = g^{x_2}, sk_2 = x_2)$ for the delegatee where $x_1, x_2 \in Z_p^*$.
3. $\mathsf{Encrypt}(m, pk_1)$: On input a message $m \in \{0,1\}^l$, and a public key pk_1, the regular encryption algorithm outputs $C = (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$,

$$r = H_1(m, h), c_1 = g^r, c_2 = g^{tr}, c_3 = h \cdot e(g,g)^{rr_1}$$

$$c_4 = m \oplus H_2(h), c_5 = pk_1^{\frac{r_1}{t}}, c_6 = pk_1^{\pi}$$

$$c_7 = g'^{\frac{r_1}{t}} H_3(c_1||c_2||c_3||c_4||c_5||c_6||pk_1)^{\pi}$$

   where t, r_1, π are random numbers from Z_p^* and h is a random element from $\mathbb{G}_T$.
4. $\mathsf{ReKeyGen}(pk_2, \mathsf{Randomness}\ (r_1, t), pk_1)$: The encryptor gives the proxy the re-encryption key

$$rk_{pk_1 \rightarrow pk_2} = (pk_2^{\frac{r_2}{t}}, g^{r_2} g^{-r_1}, g'^{\frac{r_2}{t}})$$

5. $\mathsf{ReEnc}(C, rk_{pk_1 \rightarrow pk2})$: On input the ciphertext C, the proxy outputs $C' = (c_1', c_2', c_3', c_4', c_5', c_6', c_7')$:

$$c_1' = c_1, c_2' = c_2, c_4' = c_4, c_5' = pk_2^{\frac{r_2}{t}}, c_6' = pk_2^{\pi'}$$

$$c_3' = c_3 \cdot e(c_1, g^{r_2} g^{-r_1}) = h \cdot e(g,g)^{rr_2}$$

$$c_7' = g'^{\frac{r_2}{t}} H_3(c_1'||c_2'||c_3'||c_4'||c_5'||c_6'||pk_2)'^{\pi'}$$

   where π' is a random number from Z_p^*
6. $\mathsf{Decrypt}(C, sk)$: it runs as follows:
   - On input a second-level ciphertext C, and a secret key sk_1, the decryption algorithm is as follows:
     - First check whether

$$e(c_7, pk_1) = e(g', c_5) e(H_3(c_1||c_2||c_3||c_4||c_5||c_6||pk_1), c_6)$$

       if it does not hold, output $\perp$ and terminate, otherwise do the following.
     - Compute $h = c_3 / e(c_2, c_3)^{\frac{1}{x}}$.
     - Compute $m = c_4 \oplus H_2(h)$.
     - Compute $r = H_1(m, h)$.

  - Output m, if $c_1 = g^r$, $\perp$ otherwise.
- On input a first-level ciphertext C', and a secret key sk_2, the decryption algorithm decrypt as the above case.

Remark 5. This scheme is not secure, for $g'^{\frac{r_2}{t}}$ is part of the re-encryption key, and also it is an important part of

$$c'_7 = g'^{\frac{r_2}{t}} H_3(c'_1||c'_2||c'_3||c'_4||c'_5||c'_6||pk_2)'^{\pi'}$$

If the adversary gets $g'^{\frac{r_2}{t}}$ from re-encryption key generation oracle, the variant of Waters Signature is not secure any more.

7.2 Second Attempt

1. $\mathsf{Setup}(k)$: On input the security parameter k, the setup algorithm outputs the system parameter $(g, p, \mathbb{G}, \mathbb{G}_T, e) \rightarrow \mathsf{BSetup}(k)$, hash functions H_1, H_2, H_3, H_4, H_5 and a random generator $g' \in \mathbb{G}$, where $H_1(\cdot) : \{0,1\}^* \rightarrow Z_p^*, , H_2(\cdot) : \mathbb{G}_T \rightarrow Z_p^*, H_3(\cdot) : \mathbb{G}^2 \times \mathbb{G}_T \times \{0,1\}^l \times \mathbb{G}^2 \rightarrow \mathbb{G}$.
2. $\mathsf{KeyGen}(k)$: On input the security parameter k, the key generation algorithm outputs a (public key, private key) pair $(pk_1 = g^{x_1}, sk_1 = x_1)$ for the delegator, a (public key, private key) pair $(pk_2 = g^{x_2}, sk_2 = x_2)$ for the delegatee where $x_1, x_2 \in Z_p^*$.
3. $\mathsf{Encrypt}(m, pk_1)$: On input a message $m \in \{0,1\}^l$, and a public key pk_1, the regular encryption algorithm outputs $C = (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$,

$$r = H_1(m, h), c_1 = g^r, c_2 = g^{tr}, c_3 = h \cdot e(g, g)^{rr_1}$$

$$c_4 = m \oplus H_2(h), c_5 = pk_1^{\frac{r_1}{t}}, c_6 = pk_1^{\frac{k_1 r_1}{t}}, c_7 = pk_1^{\pi}$$

$$c_8 = g'^{\frac{k_1 r_1}{t}} H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||pk_1)^{\pi}$$

   where t, r_1, k_1, π are random numbers from Z_p^* and h is a random element from $\mathbb{G}_T$.
4. $\mathsf{ReKeyGen}(pk_2, \mathsf{Randomness}\ (r_1, t), pk_1)$: The encrypter gives the proxy the re-encryption key

$$rk_{pk_1 \rightarrow pk_2} = (pk_2^{\frac{r_2}{t}}, g^{r_2} g^{-r_1}, g'^{\frac{r_2}{t}})$$

   where r_2 is a random number from Z_p^*.
5. $\mathsf{ReEnc}(C, rk_{pk_1 \rightarrow pk2})$: On input the ciphertext C, the proxy outputs $C' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7)$:

$$c'_1 = c_1, c'_2 = c_2, c'_4 = c_4, c'_5 = pk_2^{\frac{r_2}{t}}, c'_6 = pk_2^{\frac{k_2 r_2}{t}}, c'_7 = pk_2^{\pi'}$$

$$c'_3 = c_3 \cdot e(c_1, g^{r_2} g^{-r_1}) = h \cdot e(g, g)^{rr_2}$$

$$c'_8 = g'^{\frac{k_2 r_2}{t}} H_3(c'_1||c'_2||c'_3||c'_4||c'_5||c'_6||c'_7||pk_2)^{\pi'}$$

   where π', k are randomly chosen from Z_p^*.
6. $\mathsf{Decrypt}(C, sk)$: it runs as follows:
   - On input a second-level ciphertext C, and a secret key sk_1, the decryption algorithm is as follows:
     - First check whether

$$e(c_8, pk_1) = e(g', c_5) e(H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||pk_1), c_7)$$

       if it does not hold, output $\perp$ and terminate, otherwise do the following.

  - Compute $h = c_3/e(c_2, c_5)^{\frac{1}{x}}$.
  - Compute $m = c_4 \oplus H_2(h)$.
  - Compute $r = H_1(m, h)$.
  - Output m, if $c_1 = g^r$, $\perp$ otherwise.
- On input a first-level ciphertext C', and a secret key sk_2, the decryption algorithm decrypt as the above case.

Remark 6. This scheme seems to be secure, but we can not prove its security. Here we roughly give the reason. Just like the security proof in the right scheme, but when we go to Game G_5, we encounter this step:

- If $h' = c_3/e(c_9/c_7^{r^{(3)}}, g^{t/k_1})^{r^{(1)}/u}$ and $e(c_9, pk_i) = e(g', c_6)e(H_3(c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8||pk_i), c_7)$ both hold, output m' and terminate; otherwise, go to Step 3.

But in this scheme, the ciphertext does not contain g^{t/k_1}, the Challenger can not smoothly run the decryption oracle without g^{t/k_1}. To add g^{t/k_1} in the ciphertext, we need add g^t in the re-encryption key, just like the right scheme in the main body.