Fully Secure Proxy Re-Encryption without Random Oracles

Jun Shao chn.junshao@gmail.com

Zhenfu Cao* zfcao@cs.sjtu.edu.cn

Licheng Wang wanglc.cn@gmail.com

Xiaohui Liang liangxh127@sjtu.edu.cn

Department of Computer Science and Engineering, Shanghai Jiao Tong University
800 Rd. Dongchuan, Shanghai, 200240, P. R. China

June 27, 2007

Abstract

In a proxy re-encryption scheme, a semi-trusted proxy, with some additional information, can transform a ciphertext under Alice's public key into a new ciphertext under Bob's public key on the same message, but cannot learn any information about the messages encrypted under the public key of either Alice or Bob. In this paper, we propose two new unidirectional proxy re-encryption schemes, where a proxy can transform a ciphertext for Alice into a new ciphertext for Bob, but not vice versa. Note that, unidirectional proxy re-encryption is more powerful than bidirectional one, since a bidirectional scheme can always be implemented by an unidirectional one. Furthermore, these two schemes can be proved *in the standard model*, chosen-ciphertext secure based on Decisional Bilinear Inverse Diffie-Hellman assumption and master key secure based on Extended Discrete Logarithm assumption. To our best knowledge, our proposals are the first fully secure (CCA-secure and master key secure) proxy re-encryption schemes in the standard model.

Key words: proxy re-encryption, unidirectional, the standard model, fully secure.

1 Introduction

In many applications, including encrypted email forwarding [6], distributed file systems [1, 2], the DRM of Apple's iTunes [21] and the interoperable DRM architecture [23], it is desired that the data is encrypted under pk_1 can be re-encrypted under pk_2. One nice solution to this problem is *proxy re-encryption*, introduced by Blaze, Bleumer and Strauss at EUROCRYPT 1998 [6], where a semi-trusted proxy, with some additional information, can transform a ciphertext under Alice's public key into a new ciphertext under Bob's public key on the same message, but cannot learn any information about the messages encrypted under the public key of either Alice or Bob.

In [6], Blaze *et. al.* proposed a concrete proxy re-encryption scheme, named BBS scheme, based on ElGamal public key scheme [13], where the proxy can transform ciphertexts from Alice to Bob, and vice versa. That is, the BBS scheme is bidirectional. Unfortunately, the BBS scheme suffers from collusion attacks, i.e., Alice (Bob) can collude with the proxy to reveal Bob's (Alice's) secret key. Furthermore, no unidirectional proxy re-encryption scheme was proposed in [6]. Jakobsson [18], and Zhou *et. al.* [24] gave a partial solution to the problems in [6] by proposing a quorum-based protocol where the proxy is divided into sub-components.

*Corresponding Author.

In [17], based on key sharing technique, Ivan and Dodis proposed a generic construction for unidirectional proxy re-encryption, where the delegator's (Alice's) secret key is divided into two parts, one is sent to the proxy, and the other is sent to the delegatee (Bob). Although this generic construction has advantages over the previous schemes, there are also several disadvantages. (1) Besides his own secret key, the delegatee (Bob) has to store an additional secret to decrypt re-encrypted ciphertext for every delegator. (2) Though the delegator (Alice) cannot collude with the proxy to reveal the delegatee's (Bob's) secret key, the delegatee (Bob) can collude with the proxy to reveal the delegator's (Alice's) secret key. Ateniese *et. al.* [1, 2] proposed an improvement of Ivan-Dodis construction, which removes the above disadvantages.

However, as mentioned in [10], the above proxy re-encryption schemes achieve at most chosen plaintext attacks (CPA) security. In fact, applications often require security against chosen ciphertext attacks (CCA). Recently, based on another key sharing technique, Green and Ateniese proposed the first CPA and CCA secure ID-based unidirectional proxy re-encryption schemes in the random oracle model [15]. However, since they employed the key sharing technique, the schemes in [15] suffer from the following attack as the schemes in [17]: the delegatee (Bob) can collude with the proxy to reveal the delegator's (Alice's) secret key. In a concurrent and independent work [10], Canetti and Hohenberger proposed the first CCA-secure bidirectional proxy re-encryption scheme, named CH scheme, in the standard model. Their result is very nice. To achieve CCA-secure, Canetti and Hohenberger applied the Canetti, Halevi and Katz paradigm [12] (which is for transforming any selective-identity, CPA-secure ID-based encryption scheme into a CCA-secure encryption scheme) with somewhat modification. That is, adding an element Z to the ciphertext (X, Y) of a selective-identity, CPA-secure ID-based encryption scheme, such that the second part of the ciphertext Y and Z will be signed by a strongly-unforgeable one-time signature scheme, and Z allows anyone to check that unsigned first part of the ciphertext X wasn't mutated in any meaningful way. To give the proof in the standard model, they replaced the random oracles with specific concrete hash functions, in particular, the ones in [11, 4]. However, like previous bidirectional proxy re-encryption schemes, the CH scheme suffers from collusion attacks. In [10], Canetti and Hohenberger did not propose any CCA-secure unidirectional proxy re-encryption scheme in the standard model, but left it as an open problem[1]. Note that, unidirectional proxy re-encryption is more powerful than bidirectional one, since a bidirectional scheme can always be implemented by an unidirectional one.

1.1 Our Contributions

We present two proxy re-encryption schemes secure against chosen-ciphertext attacks. Our schemes are unidirectional, and efficient enough to be used in practice. We use Canetti-Hohenberger technique [10] (replacing random oracles with concrete hash fucntions) to prove them secure under the Decisional Bilinear Inverse Diffie-Hellman assumption in the standard model. Our construction is a combination of the techniques due to Fujisaki-Okamoto [14] and Canetti-Hohenberger [10]. That is, we use Canetti-Hohenberger technique (using a strongly unforgeable one-time signature scheme) to achieve public verifiability for original ciphertexts, and use Fujisaki-Okamoto technique and Canetti-Hohenberger technique to achieve CCA-security for both original ciphertexts and re-encrypted ciphertexts.

Unlike other CCA-secure proxy re-encryption schemes [15, 10], our proposals can resist the collusion attacks, i.e., Alice (Bob) and the proxy cannot cooperate to reveal Bob's (Alice's) secret key. We name this security as master key security, whose formal definition is given in Section 2.

Obviously, our proposals answer the first open problem proposed by Canetti and Hohenberger in [10].

[1]In [10], Canetti and Hohenberger proposed four open problems on proxy re-encryption, such that building (1) unidirectional CCA-secure schemes in the standard model, (2) multi-hop, unidirectional schemes, (3) unidirectional or CCA-secure scheme without bilinear groups, (4) secure obfuscations of CCA-secure re-encryption or other key translation schemes.

1.2 Paper Organization

The remaining paper is organized as follows. In Section 2, we review the definitions of proxy re-encryption and its security model against chosen-ciphertext attacks. And then, we review the Bilinear groups and the underlying complexity assumptions in Section 3. In what follows, we present our schemes and their security proofs. Finally, we conclude the paper in Section 5.

2 Definitions

The similar definitions can be found in [1, 2, 15, 10].

Definition 1 (Unidirectional PRE) *An unidirectional proxy re-encryption scheme* PRE *is a tuple of PPT algorithms* (KeyGen, ReKeyGen, Enc, ReEnc, Dec)*:*

- $\texttt{KeyGen}(1^k) \rightarrow (pk, sk)$. On input the security parameter 1^k, the key generation algorithm KeyGen outputs a public key pk and a secret key sk.
- $\texttt{ReKeyGen}(sk_1, pk_2) \rightarrow rk_{1\rightarrow 2}$. On input a secret key sk_1 and a public key pk_2, the re-encryption key generation algorithm ReKeyGen output an unidirectional re-encryption key $rk_{1\rightarrow 2}$.
- $\texttt{Enc}(pk, m) \rightarrow C$. On input a public key pk and a message m in the message space, the encryption algorithm Enc outputs a ciphertext C.
- $\texttt{ReEnc}(rk_{1\rightarrow 2}, C_1) \rightarrow C_2$. On input a re-encryption key $rk_{1\rightarrow 2}$ and a ciphertext C_1, the re-encryption algorithm ReEnc outputs an re-encrypted ciphertext C_2 or "Reject".
- $\texttt{Dec}(sk, C) \rightarrow m$. On input a secret key sk and a ciphertext C, the decryption algorithm Dec outputs a message m in the message space or "Reject".

Correctness. The correctness property has two requirements. For any message m in the message space and any key pairs $(pk, sk), (pk', sk') \leftarrow \texttt{KeyGen}(1^k)$. Then the following two conditions must hold:

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

Chosen Ciphertext Security for Unidirectional Proxy Re-Encryption.[2] We say that an unidirectional proxy re-encryption scheme PRE is semantically secure against an adaptive chosen ciphertext attack if no polynomial bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following Uni-PRE-CCA game.

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

- **Public key generation oracle** $\mathcal{O}_{pk}$: On input an index i,[3] the Challenger takes a security parameter k, and responds by running algorithm $\texttt{KeyGen}(1^k)$ to generate a key pair (pk_i, sk_i), gives pk to $\mathcal{A}$ and records (pk_i, sk_i) in table T_K.
- **Secret key generation oracle** $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, where pk is from $\mathcal{O}_{pk}$, the Challenger searches pk in table T_K and returns sk.
- **Re-encryption key generation oracle** $\mathcal{O}_{ReKeyGen}$: On input (pk, pk') by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, the Challenger returns the re-encryption key $rk_{pk\rightarrow pk'} = \texttt{ReKeyGen}(sk, pk')$, where sk is the secret key corresponding to pk.

[2]This security notion is a modification of that in [10, 15].
[3]This index is just used to distinguish the different public keys.

- **Re-encryption oracle** $\mathcal{O}_{ReEnc}$: On input (pk, pk', C) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, the Challenger returns the re-encrypted ciphertext $C' = \texttt{ReEnc}(\texttt{ReKeyGen}(sk, pk'), C)$, where sk is the secret key corresponding to pk.
- **Decryption oracle** $\mathcal{O}_{Dec}$: On input (pk, C), where pk is from $\mathcal{O}_{pk}$, the Challenger returns $\texttt{Dec}(sk, C)$, where sk is the secret key corresponding to pk.

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

Challenge: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, and a public key pk^* on which it wishes to be challenged. There are two constraints on the public key pk^*, one is that it did not appear in any query to $\mathcal{O}_{sk}$ in Phase 1, the other is that if $(pk^*, \bigstar)$ did appear in any query to $\mathcal{O}_{ReKeyGen}$, then $\bigstar$ did not appear in any query to $\mathcal{O}_{sk}$.

The Challenger picks a random bit $b \in \{0, 1\}$ and sets $C^* = \texttt{Enc}(pk^*, m_b)$. It sends C^* as the challenge to $\mathcal{A}$.

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

- **Public key generation oracle** $\mathcal{O}_{pk}$: The Challenger responds as in Phase 1.
- **Secret key generation oracle** $\mathcal{O}_{sk}$: On input pk by $\mathcal{A}$, where pk is from $\mathcal{O}_{pk}$ and $pk \neq pk^*$, and if (pk^*, pk) is not a query to $\mathcal{O}_{ReKeyGen}$, and if (pk', pk, C') is not a query to $\mathcal{O}_{ReEnc}$, where (pk', C') is a `derivative`[4] of (pk^*, C^*), the Challenger responds as in Phase 1.
- **Re-encryption key generation oracle** $\mathcal{O}_{ReKeyGen}$: On input (pk, pk') by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$ and if $pk = pk^*$, then pk' is not a query to $\mathcal{O}_{sk}$, the Challenger responds as in Phase 1.
- **Re-encryption oracle** $\mathcal{O}_{ReEnc}$: On input (pk, pk', C) by $\mathcal{A}$, where pk, pk' are from $\mathcal{O}_{pk}$, and if (pk, C) is a `derivative` of (pk^*, C^*), then pk' is not a query to $\mathcal{O}_{sk}$, the Challenger responds as in Phase 1.
- **Decryption oracle** $\mathcal{O}_{Dec}$: On input (pk, C), where pk is from $\mathcal{O}_{pk}$, and if (pk, C) is not a `derivative` of (pk^*, C^*), the Challenger responds as in Phase 1.

These queries may be also asked adaptively.

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

We refer to such an adversary $\mathcal{A}$ as an Uni-PRE-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking `PRE` as the following function of the security parameter k:

$$Adv_{\texttt{PRE},\mathcal{A}} = |\Pr[b = b'] - 1/2|.$$

Using the Uni-PRE-CCA game, we can define chosen ciphertext security for unidirectional proxy re-encryption schemes.

[4]The definition is from that in [10]. 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 O_{ReEnc} on input (pk, pk', C) and obtained (pk', C'), then (pk', C') is a derivative of (pk, C).
4. If $\mathcal{A}$ has queried $O_{ReKeyGen}$ on input (pk, pk'), and $C' = \texttt{ReEnc}(O_{ReKeyGen}(pk, pk'), C)$, then (pk', C') is a derivative of (pk, C).

Definition 2 (Uni-PRE-CCA security) *We say that the unidirectional proxy re-encryption scheme* `PRE` *is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time Uni-PRE-CCA adversary $\mathcal{A}$ the function $Adv_{\mathtt{PRE},\mathcal{A}}$ is negligible. As shorthand, we say that* `PRE` *is Uni-PRE-CCA secure.*

Definition 3 (Uni-PRE-MK security) [5] *We say that an unidirectional proxy re-encryption scheme* `PRE` *has master key security if for any polynomial bounded adversary $\mathcal{A}$, the following probability is negligible.*

$$\Pr[\mathcal{A}(pk_1, pk_2, sk_2, rk_{1\to 2}, rk_{2\to 1}) = sk_1 | (sk_1, pk_1)(sk_2, pk_2) \leftarrow \mathtt{KeyGen}(1^k)]$$

Definition 4 (Uni-PRE-Full security) *We say that an unidirectional proxy re-encryption scheme* `PRE` *has full security if and only if it is Uni-PRE-CCA secure and Uni-PRE-MK secure.*

3 Preliminaries

3.1 Bilinear Groups

In this subsection, we briefly review the definitions about bilinear maps and bilinear map groups, which follow that in [7, 8].

1. $\mathbb{G}$ and $\mathbb{G}_T$ are two (multiplicative) cyclic groups of prime order q;
2. g is a generator of $\mathbb{G}$;
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups as above. A *admissible bilinear map* is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

1. *Alternation*: For all $P, Q \in \mathbb{G}$, $e(P, Q) = e(Q, P)^{-1}$;
2. *Bilinearity*: For all $P, Q, R \in \mathbb{G}$, $e(P \cdot Q, R) = e(P, R) \cdot e(Q, R)$ and $e(P, Q \cdot R) = e(P, Q) \cdot e(P, R)$.
3. *Non-degeneracy*: If $e(P, Q) = 1$ for all $Q \in G_1$, then $P = \mathcal{O}$, where $\mathcal{O}$ is a point at infinity.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists a group $\mathbb{G}_T$ and an efficiently computable bilinear map as above. We denote `BSetup` as an algorithm that, on input the security parameter 1^k, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_T, e)$, where $q \in \Theta(2^k)$.

3.2 Complexity Assumptions

The security of the schemes proposed in this paper are based on the Decisional Bilinear Inverse Diffie-Hellman assumption (DBIDH), and the Extended Discrete Logarithm assumption (EDL).

DBIDH Problem. Let $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow$`BSetup`$(1^k)$. The DBIDH problem is as follows: Given $\langle g, g^a, g^{ab} \rangle$ for some $a, b \in \mathbb{Z}_q$ and $Q \in \mathbb{G}_T$, decide whether $Q = e(g, g)^b$. An algorithm $\mathcal{A}$ has advantage ε in solving DBIDH problem if

$$|\Pr[\mathcal{A}(g, g^a, g^{ab}, e(g, g)^b) = 0] - \Pr[\mathcal{A}(g, g^a, g^{ab}, Q) = 0]| \geq \varepsilon$$

where the probability is over the random choice of a, b in $\mathbb{Z}_q$, the random choice of Q in $\mathbb{G}_T$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}$.

[5]This security notion is from [1, 2].

Definition 5 (DBIDH Assumption) *We say that the ε-DBIDH assumption holds if no PPT algorithm has advantage at least ε in solving the DBIDH problem.*

The DBIDH assumption is used in [1, 2] to build an unidirectional proxy re-encryption. Furthermore, one can easily show that DBIDH problem is harder than q-BDHI problem, which is used to constructed many schemes without random oracles [3, 4].

EDL Problem. Let $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \texttt{BSetup}(1^k)$. The DBIDH problem is as follows: Given $\langle g, g^a, g^{1/a} \rangle$ for some $a \in \mathbb{Z}_q$, compute a. An algorithm $\mathcal{A}$ has advantage ε in solving DBIDH problem if

$$\Pr[\mathcal{A}(g, g^a, g^{1/a}) = a] \geq \varepsilon$$

where the probability is over the random choice of a in $\mathbb{Z}_q$, the random choice of Q in $\mathbb{G}_T$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}$.

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

It is easy to see that the EDL problem is easier than the Discrete Logarithm problem (DL), and the Generalized Diffie-Hellman problem (GenDH)[6] is easier than the EDL problem.

4 Fully Secure Unidirectional Proxy Re-Encryption Constructions

Following the approach of Canetti-Hohenberger [10], we first propose a simple construction in the random oracle model, and then we use the same method of [10] to replace the random oracles with concrete hash functions.

4.1 Unidirectional PRE Construction, $\prod_{Uni-RO}$, in the Random Oracle Model

Notations and Configuration. Let 1^k be the security parameter and $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \texttt{BSetup}(1^k)$, and $\texttt{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a strongly unforgeable one-time signature scheme, where $l = l(k)$ denotes the length of the verification keys output by $\mathcal{G}(1^k)$. Moreover, we assume that any given key in $\texttt{Sig}$ has a negligible chance of being sampled[7]. Let $H_1 : \{0,1\}^{\leq l} \rightarrow \mathbb{G}$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$ be three hash functions, where k_1 is another security parameter, and we will treat H_1 as a random oracle and other hash functions as any concrete collision-resistant hash function, such as SHA-1.

Define the algorithm *Check* on input a ciphertext tuple (A, B, C, D, E, F, S) and a key pk as follows:

1. Run $\mathcal{V}(A, (C, D, E, F), S)$ to verify signature S on message (C, D, E, F) with respect to key A.
2. Check that $e(B, H_1(A)) = e(pk, D)$.
3. If any of these checks fails, output 0; else output 1.

Scheme $\prod_{Uni-RO} = (\texttt{KeyGen}, \texttt{ReKeyGen}, \texttt{Enc}, \texttt{ReEnc}, \texttt{Dec})$ is described as follows:

Key Generation (KeyGen): On input 1^k, select random $x \in \mathbb{Z}_q$. Set $pk = g^x$ and $sk = x$.

Re-Encryption Key Generation (ReKeyGen): On input a public key pk_Y and a secret key $sk_X = x$, output the unidirectional re-encryption key $rk_{X \rightarrow Y} = (pk_Y)^{1/x} = g^{y/x}$.

Encryption (Enc): On input pk and a message $m \in \{0,1\}^n$, do:

[6]Many cryptographic constructions rely on GenDH assumption, including [22, 20, 5, 19, 9].

[7]It requires that $\mathcal{G}$ has super-logarithmic minimum entropy.

1. Select a one-time signature key pair as $\mathcal{G}(1^k) \rightarrow (svk, ssk)$. Set $A = svk$.
2. Select two random numbers $r \in \mathbb{Z}_q$, $\sigma \in \mathbb{G}_T$ and compute

$$B = pk^r,\ C = e(g,g)^r \cdot \sigma,\ D = H_1(A)^r,\ E = H_2(\sigma) \oplus m,\ F = H_3(\sigma||m).$$

3. Run the signing algorithm $\mathcal{S}(ssk, (C, D, E, F))$, where the message to sign is the tuple (C, D, E, F), and denote the signature S.
4. Output the ciphertext (A, B, C, D, E, F, S).

Re-Encryption (`ReEnc`): On input a re-encryption key $rk_{X\rightarrow Y}$ and a ciphertext $K = (A, B, C, D, E, F, S)$ under key pk_X, if $Check(K, pk_X) = 0$, output "Reject" and terminate; otherwise, re-encrypt the ciphertext to be under key pk_Y as:

1. Compute $B' = e(B, rk_{X\rightarrow Y}) = e(g,g)^{yr}$.
2. Output the new ciphertext $(A, B, (B', pk_X), C, D, E, F, S)$.

Decryption (`Dec`): On input a secret key sk and any ciphertext K, parse K,

Case $K = (A, B, C, D, E, F, S)$**:** If $Check(K, g^{sk}) = 0$, output "Reject" and terminate; otherwise, compute $\sigma = C/e(B,g)^{1/sk}$, $m = E \oplus H_2(\sigma)$.

Case $K = (A, B, (B', pk_X), C, D, E, F, S)$**:** If $Check(K', pk_X) = 0$, where $K' = (A, B, C, D, E, F, S)$, output "Reject" and terminate, otherwise, compute $\sigma = C/B'^{1/sk}$, $m = E \oplus H_2(\sigma)$.

If $F = H_3(\sigma||m)$, output m; otherwise, output "Reject" and terminate.

Correctness. The correctness property is easily observable.

Theorem 1 *If the DBIDH assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$*, then scheme* $\prod_{Uni-RO}$ *is Uni-PRE-CCA secure in the random oracle model.*

Proof. If there exist an adversary $\mathcal{A}$ can break $\prod_{Uni-RO}$, then we can construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve DBIDH problem, i.e., on DBIDH input (g, g^a, g^{ab}, Q), $\mathcal{B}$ decides if $Q = e(g,g)^b$ or not. $\mathcal{B}$ sets up the global parameters for $\mathcal{A}$ as follows: the description of the strongly unforgeable one-time signature `Sig`$= (\mathcal{G}, \mathcal{S}, \mathcal{V})$, the groups $\langle g \rangle = \mathbb{G}, \mathbb{G}_T$, their prime order q, and the mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, $(svk^*, ssk^*) \leftarrow \mathcal{G}(1^k)$, $A^* = svk^*$. The system parameters are $(q, g, \mathbb{G}, \mathbb{G}_T, e, H_1, H_2, H_3)$, where H_1 is a random oracle. The security parameter is $k \geq |q|$.

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

Hash oracle. On input r, $\mathcal{B}$ first checks whether pair (r, R) exists in table T_H, if yes, $\mathcal{B}$ responds $\mathcal{A}$ with Y. If not, $\mathcal{B}$ does:

$$H_1(r) = \begin{cases} (g^a), & \text{if } r = A^*; \\ g^s, & \text{otherwise, where } s \text{ is chosen randomly from } \mathbb{Z}_q. \end{cases}$$

respond $\mathcal{A}$ with $H_1(r)$ and record the pair $(r, H_1(r), s)$ in table T_H.

Phase 1. $\mathcal{B}$ builds the following oracles.

- $\mathcal{O}_{pk}$: $\mathcal{B}$ first selects a random $coin \in \{0,1\}$ so that $\Pr[coin = 0] = \delta$ for some δ[8]. And then, $\mathcal{B}$ picks a random $x_i \in \mathbb{Z}_q$. If $coin = 0$, $\mathcal{B}$ computes $pk_i = g^{x_i}$; otherwise, $\mathcal{B}$ computes $pk_i = (g^a)^{x_i}$. At last, $\mathcal{B}$ records the tuple $(pk_i, x_i, coin_i)$ in T_K, and responds $\mathcal{A}$ with pk_i.

[8]It can be determined by the same method of that in [7, 8, 16].

- $\mathcal{O}_{sk}$: On input pk_i, $\mathcal{B}$ checks whether pk_i exists in T_K, if not, $\mathcal{B}$ aborts. Otherwise, if $coin_i = 1$, $\mathcal{B}$ reports *failure* and aborts. If $coin_i = 0$, $\mathcal{B}$ responds $\mathcal{A}$ with x_i, and records pk_i in table T_{sk}.
- $\mathcal{O}_{ReKeyGen}$: On input (pk_i, pk_j), $\mathcal{B}$ checks whether pk_i and pk_j both exist in T_K, if not, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ does the following performances.
  - If $coin_i = coin_j$, $\mathcal{B}$ responds $\mathcal{A}$ with g^{x_j/x_i}, and records (pk_i, pk_j) in table T_{rk}.
  - If $coin_i = 0$ and $coin_j = 1$, $\mathcal{B}$ responds $\mathcal{A}$ with $(pk_j)^{1/x_i}$, and records (pk_i, pk_j) in table T_{rk}.
  - If $coin_i = 1$ and $coin_j = 0$, $\mathcal{B}$ reports *failure* and aborts.
- $\mathcal{O}_{ReEnc}$: On input (pk_i, pk_j, K), $\mathcal{B}$ checks whether pk_i and pk_j both exist in table T_K, if not, $\mathcal{B}$ aborts. Otherwise, if $Check(K, pk_i) = 0$, then the ciphertext is not well-formed, $\mathcal{B}$ returns "Reject" and aborts; otherwise, $\mathcal{B}$ parses $K = (A, B, C, D, E, F, S)$, and does the following performances.
  - If $coin_i = 1$ and $coin_j = 0$, $\mathcal{B}$ finds $(A, H_2(A), s)$ in table T_F, and computes $e(D^{x_j/s}, g) = e((H_2(A)^r)^{x_j/s}, g) = e((g^s)^{rx_j/s}, g) = e(g^{rx_j}, g) = e(g, g)^{rx_j} = B'$. At last, $\mathcal{B}$ responds $\mathcal{A}$ with $(A, B, (B', pk_i), C, D, E, F, S)$.
  - Otherwise, $\mathcal{B}$ calls $\mathcal{O}_{ReKeyGen}$ on input (pk_i, pk_j) to get the re-encryption key $rk_{i\to j}$, executes $\texttt{ReEnc}(rk_{i\to j}, K)$, and responds $\mathcal{A}$ with the result.
- $\mathcal{O}_{Dec}$: On input (pk_i, K), $\mathcal{B}$ checks whether pk_i exists in table T_K, if not, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ does the following performances.
  - If $coin_i = 0$, then $sk_i = x_i$, $\mathcal{B}$ responds $\mathcal{A}$ with $\texttt{Dec}(sk_i, K)$.
  - If $coin_i = 1$, $\mathcal{B}$ parse K and finds $(A, F(A), s)$ in table T_F,

    Case $K = (A, B, C, D, E, F, S)$**:** If $Check(K, pk_i) = 0$, $\mathcal{B}$ outputs "Reject" and aborts.

    Case $K = (A, B, (B', pk_X), C, D, E, F, S)$**:** If $Check(K', pk_X) = 0$, where $K' = (A, B, C, D, E, F, S)$, or $B' \neq e(pk_i, D^{1/s})$, $\mathcal{B}$ outputs "Reject" and aborts.

    $\mathcal{B}$ computes $\sigma = C/e(D^{1/s}, g) = C/e((H_1(A)^r)^{1/s}, g) = C/(g^r, g) = C/e(g, g)^r$, $m = E \oplus H_2(\sigma)$. If $F = H_3(\sigma||m)$, $\mathcal{B}$ outputs m; otherwise, $\mathcal{B}$ outputs "Reject" and aborts.

Challenge: At some point, $\mathcal{A}$ outputs a challenge tuple (pk^*, m_0, m_1). If pk^* is not in table T_K, or in table T_{sk}, or (pk^*, pk_i) is in table T_{urk}, and pk_i is in table T_{sk}, then $\mathcal{B}$ aborts. If the value of *coin*, corresponding to pk^*, is 0, $\mathcal{B}$ reports *failure* and aborts. Otherwise, $\mathcal{B}$ responds choosing a random $d \in \{0, 1\}$ and setting:

$$A^* = svk^*,\ B^* = (g^{ab})^{x^*} = (pk^*)^b,\ C^* = Q \cdot \sigma,\ D^* = g^{ab} = F(A^*)^b,$$

$$E^* = H_2(\sigma) \oplus m_d,\ F^* = H_3(\sigma||m_d),\ S^* = \mathcal{S}(ssk^*, (C^*, D^*, E^*, F^*)),$$

where x^* is in the same item with pk^* in table T_K. $\mathcal{B}$ returns $K^* = (A^*, B^*, C^*, D^*, E^*, S^*)$ to $\mathcal{A}$, and records (pk^*, K^*) in table T_{re}.

Phase 2: $\mathcal{B}$ builds the following oracles.

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

- $\mathcal{O}_{ReEnc}$: On input (pk_i, pk_j, K), if (pk_i, K) is in table T_{re}, and pk_j is in table T_{sk}, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ responds as in Phase 1. At last, if (pk_i, K) is in table T_{re}, then $\mathcal{B}$ records (pk_j, K') in table T_{re}, where K' is the resulting re-encrypted ciphertext.
- $\mathcal{O}_{Dec}$: On input (pk_i, K), if (pk_i, K) is in table T_{re}, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ responds as in Phase 1.

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

If any of the *failure* conditions (below) are false, $\mathcal{B}$ can use $\mathcal{A}$ to solve DBIDH problem.

1. The value of *coin* corresponding to pk^* is 0.
2. For each of $\mathcal{A}$'s queries $\mathcal{O}_{sk}(pk_i)$, where $pk_i \neq pk^*$, $coin_i = 1$.
3. For each of $\mathcal{A}$'s queries $\mathcal{O}_{ReKeyGen}(pk_i, pk_j)$, where $pk_i \neq pk^*$, $coin_i = 1$ and $coin_j = 0$.

Suppose $\mathcal{A}$ makes a total of q_{sk} queries to secret key generation oracle, and q_{rk} queries to re-encryption key generation oracle. Then the probability that $\mathcal{B}$ does not abort in phases 1 or 2 is $\delta^{q_{sk}} \cdot (1-(1-\delta)\delta)^{q_{rk}}$. The probability that it does not abort during the challenge step is $1-\delta$. Therefore, the probability that $\mathcal{B}$ does not abort during the simulation is $\delta^{q_{sk}}(1-\delta)(1-\delta+\delta^2)^{q_{rk}}$. Now, we assume that $q_{max} = \max\{q_{sk}, q_{rk}\}$, then we have $\delta^{q_{sk}}(1-\delta)(1-\delta+\delta^2)^{q_{rk}} \geq \delta^{q_{max}}(1-\delta)(1-\delta+\delta^2)^{q_{max}}$.

According to result of [7, 8], $\delta^{q_{max}}(1-\delta)$ reaches maximal value, i.e., $\frac{1}{e(1+q_{max})}$, when $\delta = \frac{q_{max}}{1+q_{max}}$. At this time, the value of the rest part $(1-\delta+\delta^2)^{q_{max}}$ approximate $1/e$ when q_{max} is large enough. Thus, we have

$$\delta^{q_{max}}(1-\delta)(1-\delta+\delta^2)^{q_{max}} \geq \frac{1}{e^2(1+q_{max})}$$

Now, we can finish this proof. ■

Theorem 2 *If the EDL assumption holds in $(\mathbb{G}, \mathbb{G}_T)$, then the scheme $\prod_{Uni-RO}$ has Uni-PRE-MK security.*

Proof. One can easily show that an algorithm for against master key security in $\mathbb{G}$ (given $(g, g^a, g^b, b, g^{a/b}, g^{b/a})$, compute a) gives an algorithm for solving EDL problem $\mathbb{G}$ (given $(g, g^{1/a}, g^a)$, compute a). ■

Combining the above two theorems, we have

Theorem 3 *Under DBIDH assumption and EDL assumption, $\prod_{Uni-RO}$ has Uni-PRE-Full security in the random oracle model.*

4.2 Unidirectional PRE Construction, $\prod_{Uni}$, without Random Oracles

In this subsection, we apply the technique of [10] to convert $\prod_{Uni-RO}$ to $\prod_{Uni}$, the latter one can be proved in the standard model. The only difference between $\prod_{Uni-RO}$ and $\prod_{Uni}$ is the function H_1. In $\prod_{Uni-RO}$, $H_1(y) \overset{def}{=} g_2^y \cdot g_3$[9] instead of a random oracle, where g_2 and g_3 are two random numbers in $\mathbb{G}$.

Theorem 4 *If the DBIDH assumption holds in $(\mathbb{G}, \mathbb{G}_T)$, then scheme $\prod_{Uni}$ is Uni-PRE-CCA secure in the standard model.*

[9]In fact, as mentioned in [10], we use y instead of $\tilde{y}$ for simplicity, where $\tilde{y}$ is a fixed one-to-one representation of y in $\mathbb{Z}_q$. This one-to-one mapping from y to $\tilde{y}$ can be implemented by an additional hash function.

Proof. If there exist an adversary $\mathcal{A}$ break $\prod_{Uni}$, then we can construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solves DBIDH problem, i.e., on DBIDH input (g, g^a, g^{ab}, Q), $\mathcal{B}$ decides if $Q = e(g,g)^b$ or not. $\mathcal{B}$ sets up the global parameters for $\mathcal{A}$ as follows: the description of the groups $\langle g \rangle = \mathbb{G}, \mathbb{G}_T$, their prime order q, and the mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, $(svk^*, ssk^*) \leftarrow \mathcal{G}(1^k)$, $A^* = svk^*$, $g_2 = g^{\alpha_1}$, $g_3 = g^{a\alpha_2 - \alpha_1 A^*}$, where α_1 and α_2 are two random numbers from $\mathbb{Z}_q$. The system parameters are $(q, g, g_2, g_3, \mathbb{G}, \mathbb{G}_T, e, H_1, H_2, H_3)$.

The security parameter is $k \geq |q|$.

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

Phase 1. $\mathcal{B}$ builds the following simulations.

- $\mathcal{O}_{pk}$: The same as in the proof of Theorem 1.
- $\mathcal{O}_{sk}$: The same as in the proof of Theorem 1.
- $\mathcal{O}_{ReKeyGen}$: The same as in the proof of Theorem 1.
- $\mathcal{O}_{ReEnc}$: On input (pk_i, pk_j, K), $\mathcal{B}$ checks whether pk_i and pk_j both exist in table T_K, if not, $\mathcal{B}$ aborts. Otherwise, if $Check(K, pk_i) = 0$, then the ciphertext is not well-formed, $\mathcal{B}$ outputs "Reject" and aborts; otherwise, $\mathcal{B}$ parses $K = (A, B, C, D, E, F, S)$, and does the following performances.
  - If $coin_i = 1$ and $coin_j = 0$, $\mathcal{B}$ computes:
  $$t = \frac{D}{B^{\alpha_2/x_i}},\ \lambda = \frac{1}{\alpha_1(A - A^*)}.$$
  Then $\mathcal{B}$ gets $B' = e((t^\lambda)^{x_j}, g)$, and responds $\mathcal{A}$ with $(A, B, (B', pk_i), C, D, E, F, S)$. Note that when $A \neq A^*$,[10] then $\mathcal{B}$ can solve for $t^\lambda = g^r$ since:
  $$t = \frac{H_1(A)}{(pk_i^r)^{\alpha_2/x_i}} = \frac{g_2^{rA} g_3^r}{pk_i^{r\alpha_2/x_i}} = \frac{(g^{\alpha_1})^{rA}(g^{a\alpha_2 - \alpha_1 A^*})^r}{(g^{ax_i})^{r\alpha_2/x_i}} = \frac{g^{r\alpha_1(A-A^*)+ra\alpha_2}}{g^{ra\alpha_2}} = g^{r\alpha_1(A-A^*)}.$$
  - Otherwise, $\mathcal{B}$ calls $\mathcal{O}_{ReKeyGen}$ on input (pk_i, pk_j) to get the re-encryption key $rk_{i\to j}$, executes $\texttt{ReEnc}(rk_{i\to j}, K)$, and responds $\mathcal{A}$ with the result.
- $\mathcal{O}_{Dec}$: On input (pk_i, K), $\mathcal{B}$ checks whether pk_i exists in table T_K, if not, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ does the following performances.
  - If $coin_i = 0$, then $sk_i = x_i$, $\mathcal{B}$ responds $\mathcal{A}$ with $\texttt{Dec}(sk_i, K)$.
  - If $coin_i = 1$, $\mathcal{B}$ parses K,

    Case $K = (A, B, C, D, E, F, S)$**:** If $Check(K, pk_i) = 0$, $\mathcal{B}$ outputs "Reject" and aborts; otherwise, $\mathcal{B}$ solves for g^r as it does in $\mathcal{O}_{ReEnc}$, and computes $\sigma = C/e(g^r, g)$, $m = E \oplus H_2(\sigma)$. If $F = H_3(\sigma||m)$, $\mathcal{B}$ outputs m; otherwise, $\mathcal{B}$ outputs "Reject" and aborts.

    Case $K = (A, B, (B', pk_X), C, D, E, F, S)$**:** If $Check(K', pk_X) = 0$, where $K' = (A, B, C, D, E, F, S)$, $\mathcal{B}$ outputs "Reject" and aborts; otherwise, $\mathcal{B}$ does

    * If the value of $coin_X$, corresponding to pk_X, is 0, then $\mathcal{B}$ computes $g^r = B^{\frac{1}{x_X}}$, and checks $B' \overset{?}{=} e(g^r, pk_i)$. If not, $\mathcal{B}$ outputs "Reject" and aborts; otherwise, $\mathcal{B}$ returns the result of $\texttt{Dec}(x_X, K')$, where $K' = (A, B, C, D, E, F, S)$.

[10] We assume that any given key of the one-time signature scheme has a negligible chance of being sampled, hence, the probability of the event $A = A^*$ is negligible.

* If the value of $coin_X$, corresponding to pk_X, is 1, then $\mathcal{B}$ solves for g^r as it does in $\mathcal{O}_{ReEnc}$, and checks $B' \overset{?}{=} e(g^r, pk_i)$. If not, $\mathcal{B}$ outputs "Reject" and aborts; otherwise, $\mathcal{B}$ computes $\sigma = C/e(g^r, g)$, $m = E \oplus H_2(\sigma)$. If $F = H_3(\sigma||m)$, $\mathcal{B}$ outputs m; otherwise, $\mathcal{B}$ outputs "Reject" and aborts.

Challenge: At some point, $\mathcal{A}$ outputs a challenge tuple (pk^*, m_0, m_1). If pk^* is not in table T_K, or pk^* is table T_{sk}, or (pk^*, pk_i) is in table T_{rk}, and pk_i is in table T_{sk}, then $\mathcal{B}$ aborts. If the value of $coin^*$, corresponding to pk^*, is 0, $\mathcal{B}$ reports *failure* and aborts. Otherwise, $\mathcal{B}$ responds choosing a random $d \in \{0,1\}$ and setting:

$$A^* = svk^*,\ B^* = (g^{ab})^{x^*} = (pk^*)^b,\ C^* = Q \cdot \sigma,$$

$$D^* = (g^{ab})^{\alpha_2} = ((g^{\alpha_1})^{A^*} \cdot g^{a\alpha_2 - \alpha_1 A^*})^b = (g_2^{A^*} \cdot g_3)^b = H_1(A^*)^b,$$

$$E^* = H_2(\sigma) \oplus m_d,\ F^* = H_3(\sigma||m_d),\ S^* = \mathcal{S}(ssk^*, (C^*, D^*, E^*, F^*)),$$

where x^* is in the same item with pk^* in table T_K. $\mathcal{B}$ returns $K^* = (A^*, B^*, C^*, D^*, E^*, S^*)$ to $\mathcal{A}$, and records (pk^*, K^*) in table T_{re}.

Phase 2: $\mathcal{B}$ builds the following simulations.

- $\mathcal{O}_{pk}$: The same as in the proof of Theorem 1.
- $\mathcal{O}_{sk}$: The same as in the proof of Theorem 1.
- $\mathcal{O}_{ReKeyGen}$: On input (pk_i, pk_j), if $pk_i = pk^*$, and pk_j is in table T_{sk}, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ responds as in Phase 1.
- $\mathcal{O}_{ReEnc}$: On input (pk_i, pk_j, K), if (pk_i, K) is in table T_{re}, and pk_j is in table T_{sk}, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ responds as in Phase 1. At last, if (pk_i, K) is in table T_{re}, then $\mathcal{B}$ records (pk_j, K') in table T_{re}, where K' is the resulting re-encrypted ciphertext.
- $\mathcal{O}_{Dec}$: On input (pk_i, K), if (pk_i, K) is in table T_{re}, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ responds as in Phase 1.

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

The rest analysis follows the previous proof. ■

Similar with $\prod_{Uni-RO}$, we have the following theorems.

Theorem 5 *If the EDL assumption holds in $(\mathbb{G}, \mathbb{G}_T)$, then scheme $\prod_{Uni}$ has Uni-PRE-MK security.*

Theorem 6 *Under DBIDH assumption and EDL assumption, $\prod_{Uni}$ has Uni-PRE-Full security in the standard model.*

4.2.1 Discussion of scheme $\prod_{Uni}$

The scheme $\prod_{Uni}$ is the first unidirectional CCA-secure scheme without random oracles, which answers the first problem asked by Canetti and Hohenberger in [10]. However, it still has three disadvantages as follows:

- The delegator can also decrypt the re-encrypted ciphertext without any more information.[11]

[11] In [1, 2], the authors considered this disadvantage as a good property, named *original-access*. However, we point that non-original-access scheme is more powerful than original-access one, since original-access one can always be implemented by non-original-access one with the re-encrypted ciphertext containing the original ciphertext.

- The size of ciphertext is expanded after re-encryption.
- The delegatee knows the delegator from the re-encrypted ciphertext.

Can we remove the above disadvantages to construct a new scheme which is also CCA-secure without random oracles? Fortunately, the answer is "Yes". In next subsection, we propose such a scheme, named $\prod'_{Uni}$, which also applies Canetti-Hohenberger technique.

4.3 A Variation $\prod'_{Uni}$

Notations and Configuration. $(q, g, \mathbb{G}, \mathbb{G}_T, e, \mathtt{Sig})$ are the same as those in $\prod_{Uni}$, g_2 and g_3 are two random numbers of $\mathbb{G}$. Let $H_1 : \{0,1\}^* \to \mathbb{Z}_q$, $H_3 : \{0,1\}^* \to \{0,1\}^n$ be two hash functions, and $H_2(y) \overset{def}{=} g_2^y \cdot g_3$.

Define the algorithm *Check* on input a ciphertext tuple (A, B, C, D, E, S) and a key pk as follows:

1. Run $\mathcal{V}(A, (C, D, E), S)$ to verify signature S on message (C, D, E) with respect to key A.
2. Check that $e(B, H_2(A)) = e(pk, D)$.
3. If any of these checks fail, output 0; else output 1.

Scheme $\prod'_{Uni} = (\mathtt{KeyGen}, \mathtt{ReKeyGen}, \mathtt{Enc}, \mathtt{ReEnc}, \mathtt{Dec})$ is described as follows:

Key Generation (`KeyGen`): The same as that in $\prod_{Uni}$.

Re-Encryption Key Generation (`ReKeyGen`): The same as that in $\prod_{Uni}$.

Encryption (`Enc`): On input pk and a message $m \in \mathbb{G}_T$, do:

1. Select a one-time signature key pair as $\mathcal{G}(1^k) \to (svk, ssk)$. Set $A = svk$.
2. Select a random number $\sigma \in \mathbb{G}_T$ and compute

$$r = H_1(\sigma || m),\ B = pk^r,\ C = e(g, g)^r \cdot \sigma,\ D = H_2(A)^r,\ E = H_3(\sigma) \oplus m.$$

3. Run the signing algorithm $\mathcal{S}(ssk, (C, D, E))$, where the message to sign is the tuple (C, D, E), and denote the signature S.
4. Output the ciphertext (A, B, C, D, E, S).

Re-Encryption (`ReEnc`): On input a re-encryption key $rk_{X \to Y}$ and a ciphertext $K = (A, B, C, D, E, F, S)$ under key pk_X, if $Check(K, pk_X) = 0$, output "Reject" and terminate; otherwise, re-encrypt the ciphertext to be under key pk_Y as:

1. Compute $B' = e(B, rk_{X \to Y}) = e(g, g)^{yr}$.
2. Output the new ciphertext (A, B', C, D, E, S).

Decryption (`Dec`): On input a secret key sk and any ciphertext K, parse $K = (A, B, C, D, E, S)$,

Case $B \in \mathbb{G}$**:** If $Check(K, g^{sk}) = 0$, output "Reject" and terminate; otherwise, compute $\sigma = C/e(B, g)^{1/sk}$, $m = E \oplus H_2(\sigma)$. If $B = pk^{H_1(\sigma||m)}$ and $D = H_2(A)^{H_1(\sigma||m)}$, output m; otherwise, output "Reject" and terminate.

Case $B \in \mathbb{G}_T$**:** Compute $\sigma = C/B^{1/sk}$, $m = E \oplus H_2(\sigma)$. If $B = e(pk, g)^{H_1(\sigma||m)}$ and $D = H_2(A)^{H_1(\sigma||m)}$, output m; otherwise, output "Reject" and terminate.

Correctness. The correctness property is easily observable.

The main difference between $\prod_{Uni}$ and $\prod'_{Uni}$ is that in $\prod'_{Uni}$, $r = H_1(\sigma||m)$ instead of r is a random number in $\mathbb{Z}_q$.

Similar with $\prod_{Uni}$, we have the following three theorems on $\prod'_{Uni}$.

Theorem 7 *If the DBIDH assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$*, then scheme* $\prod'_{Uni}$ *is an Uni-PRE-CCA secure in the standard model.*

Theorem 8 *If the EDL assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$*, then scheme* $\prod'_{Uni}$ *has Uni-PRE-MK security.*

Theorem 9 *Under DBIDH assumption EDL assumption,* $\prod'_{Uni}$ *has Uni-PRE-Full security in the standard model.*

5 Conclusions

In this paper, we proposed two new unidirectional proxy re-encryption schemes, which are efficient enough to be used in practice. Furthermore, they have other nice properties, including master key security (collusion-resistance), CCA-secure in the standard model, no additional secret the delegatee needs to store to decrypt the re-encrypted ciphertexts, non-interactive re-encryption key generation.

This work is the answer to the first open problem proposed by Canetti and Hohenberger in [10], and we are working on solving other open problems.

References

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applicactions to Secure Distributed Storage. In *Internet Society (ISOC): NDSS 2005*, pp. 29-43, 2005. 1, 2, 5, 3.2, 11

[2] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applicactions to Secure Distributed Storage. *ACM Transactions on Information and System Security (TISSEC)*, 9 (1), 2006, pp. 1-30. 1, 2, 5, 3.2, 11

[3] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, LNCS 3027, pp: 56-73, 2004. 3.2

[4] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT 2004*, LNCS 3027, pp. 223-238, 2004. 1, 3.2

[5] E. Biham, D. Boneh, and O. Reingold. Breaking generalized Diffie- Hellman modulo a composite is no easier than factoring. *IPL*, 70, 1999, pp. 83-87. 6

[6] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT 1998*, LNCS 1403, pp. 127-144, 1998. 1

[7] D. Boneh, M. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001*, LNCS 2139, pp. 231-229, 2001. 3.1, 8, 4.1

[8] D. Boneh, M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32 (3), 2003, pp. 586-615. 3.1, 8, 4.1

[9] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324, 2003, pp. 71-90. 6

[10] R. Canetti, S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. 2007. Cryptology ePrint Archieve: Report 2007/171. 1, 1.1, 1, 2, 2, 4, 4, 4.2, 9, 4.2.1, 5

[11] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, LNCS 2656, pp. 255-271, 2003. 1

[12] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT 2004*, LNCS 3027, pp. 207-222, 2004. 1

[13] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31/4, 1985, pp. 469-472. 1

[14] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO 1999*, LNCS 1666, pp. 537-554, 1999. 1.1

[15] M. Green and G. Ateniese. Identity-Based Proxy Re-encryption. 2006. Cryptology ePrint Archieve: Report 2006/473. 1, 1.1, 2, 2

[16] C. Gentry and A. Silverberg. Hierachical ID-Based Cryptography. In *ASIACRYPT 2002*, LNCS 2501, pp. 548-566, 2002. 8

[17] A. Ivan and Y. Dodis. Proxy Cryptography Revisited. In *Internet Society (ISOC): NDSS 2003*, 2003. 1

[18] M. Jakobsson. On Quorum Controlled Asymmetric Proxy Re-encryption. In *PKC 1999*, LNCS 1560, pp. 112-121, 1999. 1

[19] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO 2002*, LNCS 2442, pp. 597-612 2002. 6

[20] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *STOC 1997*, pp. 458-467, 1997. 6

[21] T. Smith. DVD Jon: buy DRM-less tracks from Apple iTunes, March 18, 2005. Available at http://www.theregister.co.uk/2005/03/18/itunes_pymusique/. 1

[22] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to group communication. In *ACM CCS 1996*, pp. 31-37, 1996. 6

[23] G. Taban, A.A. Cárdenas and V.D. Gligor. Towards a Secure and Interoperable DRM Architecture. In *ACM DRM 2006*, pp. 69-78, 2006. 1

[24] L. Zhou, M.A. Marsh, F.B. Schneider, and A. Redz. Distributed Blinding for Distributed ElGamal Re-encryption. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05) - Volume 00*, pp. 824-834, 2005. 1