

# Chosen-Ciphertext Secure Certificateless Proxy Re-Encryption

Chul Sur<sup>1</sup>, Chae Duk Jung<sup>2</sup>, Youngho Park<sup>2</sup>, and Kyung Hyune Rhee<sup>3</sup>

<sup>1</sup> Department of Computer Science, Pukyong National University,  
599-1, Daeyeon3-Dong, Nam-Gu, Busan 608-737, Republic of Korea  
`kahlil@pknu.ac.kr`

<sup>2</sup> Department of Information Security, Pukyong National University  
`{jcd0205,pyhoya}@pknu.ac.kr`

<sup>3</sup> Department of IT Convergence and Application Engineering,  
Pukyong National University  
`khrhee@pknu.ac.kr`

**Abstract.** In this paper we introduce the notion of certificateless proxy re-encryption and also give precise definitions for secure certificateless proxy re-encryption schemes. We present a concrete scheme based on bilinear pairing, which enjoys the advantages of certificateless public key cryptography while providing the functionalities of proxy re-encryption. Moreover, the proposed scheme is unidirectional and compatible with current certificateless encryption deployments. Finally, we show that our scheme has chosen ciphertext security in the random oracle model.

**Keywords:** Certificateless Public Key Encryption, Proxy Re-Encryption, Chosen Ciphertext Security, Bilinear Pairing.

## 1 Introduction

In a proxy re-encryption (PRE) scheme, a semi-trusted proxy is allowed to transform a ciphertext under Alice's public key into a new ciphertext under Bob's public key on the same message. However, the proxy cannot learn any information about the messages encrypted under the public key of either Alice or Bob. PRE has received plenty of attention from the research community as fundamental cryptographic function to solve key management problems in various practical applications such as distributed file systems, secure email forwarding, and interoperable DRM systems in recent years [3,4,10].

In [7], Blaze *et al.* introduced the notion of PRE and proposed a concrete PRE scheme, where the message and secret keys are kept hidden from the proxy. However, their scheme is bidirectional. That is, the information released to divert ciphertexts from Alice to Bob can also be used to transform ciphertexts in the opposite direction [3,10]. Note that, it is obvious that unidirectional PRE is more powerful than bidirectional one since any unidirectional scheme can be easily transformed to a bidirectional one. Moreover, their scheme suffers from collusion attack that Alice (Bob) can collude with the proxy to reveal Bob's (Alice's) secret

key. Ateniese *et al.* [3] presented the first constructions of unidirectional proxy re-encryption based on bilinear pairing. The schemes prevent the proxy from colluding with Bob to expose Alice's secret key. However, their schemes only achieve chosen plaintext attack (CPA) security.

As pointed out in [10], chosen plaintext security is clearly not enough for many applications that require security against chosen ciphertext attack (CCA). Canetti and Hohenberger [10] addressed the problem of obtaining a PRE scheme that is secure against chosen ciphertext attack. They then provided a formal security model for secure PRE schemes and presented a construction that has chosen ciphertext security in the standard model. However, their construction is still bidirectional and vulnerable to collusion attack like previous bidirectional PRE schemes. Recently, Libert and Vergnaud [15] presented the first construction of unidirectional proxy re-encryption that has chosen ciphertext security in the standard model. In [13], Green and Ateniese addressed the problem of identity-based PRE and presented a unidirectional scheme that has chosen ciphertext security in the random oracle model [6]. However, the scheme suffers from the collusion attack owing to the key sharing technique employed in the scheme. That is, Bob can collude with the proxy to reveal Alice's secret key.

Even though a number of PRE schemes have been proposed in the literature, all prior PRE schemes are constructed based on either traditional public key encryption (PKE) [3,7,10,15,16] or identity-based encryption (IBE) [13]. However, it is well recognized that traditional PKE suffers from the issues associated with certificate management such as revocation and IBE has inherent key escrow problem. To alleviate the aforementioned problems in traditional PKE and IBE, the concept of certificateless public key cryptography (CL-PKC), which combines the best aspects of traditional PKE (i.e., key escrow free) and of IBE (i.e., implicit certification), was introduced in [1]. The topic of CL-PKC has undergone quite rapid development with many schemes being proposed for encryption, signature, authenticated key agreement, and so on.

**Our Contribution.** In this paper, we introduce the notion of certificateless proxy re-encryption that enjoys the best aspects of traditional PKE and of IBE while providing the functionalities of proxy re-encryption. Upon taking into consideration of both security notions of certificateless encryption and proxy re-encryption, we provide a formal security model for constructing secure certificateless proxy re-encryption schemes and present a concrete scheme based on bilinear pairing. The proposed scheme is unidirectional and compatible with existing certificateless encryption deployments. Finally, we show that our scheme is secure against adaptive chosen ciphertext attack in the random oracle model.

The rest of the paper is organized as follows. The next section gives precise definition and security model for certificateless proxy re-encryption schemes. In Section 3, we present the first construction of certificateless proxy re-encryption based on bilinear pairing and then prove that the proposed scheme has chosen ciphertext security in the random oracle model under reasonable complexity assumption. Finally, we conclude the paper in Section 4.

## 2 Certificateless Proxy Re-Encryption

In this section, we present a definition of a certificateless proxy re-encryption (CL-PRE) scheme and provide a formal security model so as to construct secure CL-PRE schemes.

### 2.1 Definition

We incorporate certificateless encryption (CLE) [1,2] into proxy re-encryption (PRE) [10,13] for constructing a new certificateless proxy re-encryption scheme. A model for certificateless proxy re-encryption is specified as follows:

**Definition 1 (CL-PRE).** *A unidirectional certificateless proxy re-encryption scheme is a 9-tuple of algorithms which are the following:*

- **Setup**( $k$ ): Takes a security parameter  $k$  and returns a randomly chosen master key  $mk$  and a list of public parameters  $\mathbf{params}$ .
- **Partial-Private-Key-Extract**( $\mathbf{params}, mk, ID_A$ ): Takes a list of public parameters  $\mathbf{params}$ , an identifier for user  $A$ ,  $ID_A \in \{0, 1\}^*$  and the master key  $mk$  as inputs, returns a partial private key  $d_A$ .
- **Set-Secret-Value**( $\mathbf{params}, ID_A$ ): Given a list of public parameters  $\mathbf{params}$  and an identifier for user  $A$ , returns a randomly chosen secret value  $x_A$  for that user.
- **Set-Private-Key**( $\mathbf{params}, d_A, x_A$ ): Given a list of public parameters  $\mathbf{params}$ , a user's partial private key  $d_A$  and secret value  $x_A$ , outputs a private key  $sk_A$ .
- **Set-Public-Key**( $\mathbf{params}, x_A$ ): Takes a list of public parameters  $\mathbf{params}$  and a user's secret value  $x_A$  as inputs, outputs a corresponding public key  $pk_A$ .
- **Encrypt**( $m, \mathbf{params}, ID_A, pk_A$ ): Takes a message  $m$ , a list of public parameters  $\mathbf{params}$ , an identifier  $ID_A$  of the receiver, and a corresponding public key  $pk_A$  as inputs. It outputs a ciphertext  $C_A$  or a distinguished symbol  $\perp$ .
- **Set-Proxy-Re-Encryption-Key**( $\mathbf{params}, ID_A, pk_A, sk_A, ID_B, pk_B$ ): Takes a list of public parameters  $\mathbf{params}$ , a user  $A$ 's identifier  $ID_A$  and a public/private key pair  $(pk_A, sk_A)$ , a user  $B$ 's identifier  $ID_B$  and a public key  $pk_B$  as inputs. It outputs a unidirectional re-encryption key  $rk_{A \rightarrow B}$ .
- **Re-Encrypt**( $\mathbf{params}, rk_{A \rightarrow B}, C_A$ ): Takes a list of public parameters  $\mathbf{params}$ , a re-encryption key  $rk_{A \rightarrow B}$ , and a ciphertext  $C_A$  as inputs. It outputs a re-encrypted ciphertext  $C_B$  or a distinguished symbol  $\perp$ .
- **Decrypt**( $\mathbf{params}, sk_{ID}, C_{ID}$ ): Given a list of public parameters  $\mathbf{params}$ , a ciphertext  $C_{ID}$ , and the receiver  $ID$ 's private key  $sk_{ID}$ , it outputs a message  $m$  or a distinguished symbol  $\perp$ .

For completeness, it is obviously required that the following two propositions must hold for any message  $m$  in the message space  $\mathcal{M}$ :

- $\text{Decrypt}(\mathbf{params}, sk_A, \text{Encrypt}(m, \mathbf{params}, ID_A, pk_A)) = m$ .
- $\text{Decrypt}(\mathbf{params}, sk_B, \text{Re-Encrypt}(\mathbf{params}, rk_{A \rightarrow B}, C_A)) = m$ .

where  $sk_{ID} \leftarrow \text{Set-Private-Key}(\mathbf{params}, d_{ID}, x_{ID})$  and  $rk_{A \rightarrow B} \leftarrow \text{Set-Proxy-Re-Encryption-Key}(\mathbf{params}, ID_A, pk_A, sk_A, ID_B, pk_B)$ .

## 2.2 Security Model

Here we provide a security model to prove the adaptive chosen ciphertext security of our CL-PRE scheme. To give a precise security notion for the proposed scheme, we take into account both security notions of certificateless encryption [1,2] and proxy re-encryption [10,16]. Following the security model in [1,2], there are two kinds of adversaries, named Type I and Type II adversaries, that represent a malicious third party and an honest-but-curious key generation center (KGC) in our security model, respectively. In addition, we consider the notion of derivative ciphertexts [10,16] to prevent the adversaries from breaking our CL-PRE scheme by means of re-encryption queries and re-encryption key queries corresponding to the challenge identity  $ID^*$  and the challenge ciphertext  $C^*$ .

We then define two different types of “indistinguishability of encryptions under chosen ciphertext attack” (IND-CCA) games against the Type I and the Type II adversaries, respectively. The first game between the Type I adversary (denoted by  $\mathcal{A}_I$ ) and the challenger is defined as follows:

**Setup:** The challenger takes a security parameter  $k$  and runs the Setup algorithm. It gives the resulting public parameters  $\text{params}$  to  $\mathcal{A}_I$  and keeps the master key  $\text{mk}$  to itself.

**Phase 1:**  $\mathcal{A}_I$  issues queries  $q_1, \dots, q_m$  adaptively where query  $q_i$  is one of:

- **Extraction query** on  $ID_i$ . The challenger responds by running algorithm Partial-Private-Key-Extraction to generate the partial private key  $d_{ID_i}$  for  $ID_i$ .
- **Private Key query** on  $ID_i$ . If the public key for  $ID_i$  has not been replaced, the challenger responds by running algorithm Set-Private-Key to generate the private key  $sk_{ID_i}$  for  $ID_i$ .
- **Public Key query** on  $ID_i$ . The challenger responds by running algorithm Set-Public-Key to generate the public key  $pk_{ID_i}$  for  $ID_i$ .
- **Replace Public Key query** on the public key for  $ID_i$ .  $\mathcal{A}_I$  can repeatedly replace the public key  $pk_{ID_i}$  for  $ID_i$  with any value  $pk'_{ID_i}$  of its choice.
- **Re-Encryption Key query** on  $(ID_i, ID_j)$ . The challenger responds by running algorithm Set-Proxy-Re-Encryption-Key to generate the re-encryption key  $rk_{i \rightarrow j}$ .
- **Re-Encryption query** on  $(ID_i, ID_j, C_{ID_i})$ . The challenger responds by running algorithm Re-Encrypt to transform the ciphertext  $C_{ID_i}$  into the re-encrypted ciphertext  $C_{ID_j}$  using the re-encryption key  $rk_{i \rightarrow j}$ .
- **Decryption query** on  $(ID_i, C_{ID_i})$ . The challenger responds by running algorithm Decrypt to decrypt the ciphertext  $C_{ID_i}$  using the private key  $sk_{ID_i}$ . Even though the public key for  $ID_i$  may be replaced, the challenger is forced to respond with a correct answer as in [1,2].

**Challenge:** Once  $\mathcal{A}_I$  decides that Phase 1 is over, it outputs two equal length plaintexts  $m_0, m_1 \in \mathcal{M}$  and an identity  $ID^*$  of uncorrupted private key, on which it wishes to be challenged. In particular,  $ID^*$  may not have been submitted to both Replace Public Key and Extraction queries. Moreover,  $\mathcal{A}_I$  is restricted to the choice of  $ID^*$  such that trivial decryption is not possible

using keys extracted in Phase 1, e.g., by using re-encryption keys to transform  $C_{ID^*}$  into  $C_{ID_i}$  for which  $\mathcal{A}_I$  holds a decryption key. The challenger picks a random bit  $b \in \{0, 1\}$  and computes the challenge ciphertext  $C^* = \text{Encrypt}(m_b, \text{params}, \text{ID}^*, pk_{ID^*})$ , where  $pk_{ID^*}$  is the public key currently associated with  $\text{ID}^*$ . It sends  $C^*$  as the challenge to  $\mathcal{A}_I$ .

**Phase 2:**  $\mathcal{A}_I$  issues more queries  $q_{m+1}, \dots, q_n$  adaptively where  $q_i$  is one of:

- **Decryption query** on  $(\text{ID}_i, C_{ID_i})$ . If  $(\text{ID}_i, C_{ID_i})$  is not a *challenge derivative*, the challenger responds as in Phase 1. The definition of challenge derivative is as follows [10,16]:
  1.  $(\text{ID}^*, C^*)$  is a challenge derivative of itself.
  2. If  $(\text{ID}_i, C_{ID_i})$  is a challenge derivative,  $\mathcal{A}_I$  has issued the Re-Encryption query on  $(\text{ID}_i, \text{ID}_j, C_{ID_i})$  to receive a new ciphertext  $C_{ID_j}$ , then  $(\text{ID}_j, C_{ID_j})$  is a challenge derivative.
  3. If  $(\text{ID}_i, C_{ID_i})$  is a challenge derivative,  $\mathcal{A}_I$  has issued the Re-Encryption Key query on  $(\text{ID}_i, \text{ID}_j)$  to obtain a re-encryption key  $rk_{i \rightarrow j}$ , and  $C_{ID_j} = \text{Re-Encrypt}(\text{params}, rk_{i \rightarrow j}, C_{ID_i})$ , then  $(\text{ID}_j, C_{ID_j})$  is a challenge derivative.
- **Extraction query** on  $\text{ID}_i$ . The challenger responds as in Phase 1 except that  $\mathcal{A}_I$  has issued the Replace Public Key query on  $\text{ID}_i$  and a challenge derivative  $(\text{ID}_i, C_{ID_i})$  exists.
- **Private Key query** on  $\text{ID}_i \neq \text{ID}^*$  where a challenge derivative  $(\text{ID}_i, C_{ID_i})$  does not exist. The challenger responds as in Phase 1.
- **Public Key query** on  $\text{ID}_i$ . The challenger responds as in Phase 1.
- **Replace Public Key query** on the public key for  $\text{ID}_i$ . The challenger responds as in Phase 1.
- **Re-Encryption Key query** on  $(\text{ID}_i, \text{ID}_j)$ . The challenger responds as in Phase 1 except that  $\text{ID}_i = \text{ID}^*$  and  $\mathcal{A}_I$  has issued the Private Key query on  $\text{ID}_j$ .
- **Re-Encryption query** on  $(\text{ID}_i, \text{ID}_j, C_{ID_i})$ . The challenger responds as in Phase 1 except that  $\mathcal{A}_I$  has issued the Private Key query on  $\text{ID}_j$  and  $(\text{ID}_i, C_{ID_i})$  is a challenge derivative.

**Guess:** Finally,  $\mathcal{A}_I$  outputs a guess  $b' \in \{0, 1\}$ .  $\mathcal{A}_I$  wins if  $b' = b$ .

We define the advantage of  $\mathcal{A}_I$  in breaking the CL-PRE scheme as

$$\text{Adv}(\mathcal{A}_I) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

**Definition 2.** We say that a unidirectional CL-PRE scheme is  $(t, \epsilon)$ -adaptive chosen ciphertext (IND-CCA) secure against the Type I adversary if for any  $t$ -time adversary  $\mathcal{A}_I$  we have that  $\text{Adv}(\mathcal{A}_I) < \epsilon$ .

We now define the second game between the Type II adversary (denoted by  $\mathcal{A}_{II}$ ) and the challenger as follows:

**Setup:** The challenger takes a security parameter  $k$  and runs the Setup algorithm. It gives the resulting public parameters  $\text{params}$  and the master key  $\text{mk}$  to  $\mathcal{A}_{II}$ .

**Phase 1:**  $\mathcal{A}_{II}$  issues queries  $q_1, \dots, q_m$  adaptively where query  $q_i$  is one of:

- **Private Key query** on  $ID_i$ . The challenger responds by running algorithm Set-Private-Key to generate the private key  $sk_{ID_i}$  for  $ID_i$ .
- **Public Key query** on  $ID_i$ . The challenger responds by running algorithm Set-Public-Key to generate the public key  $pk_{ID_i}$  for  $ID_i$ .
- **Re-Encryption Key query** on  $(ID_i, ID_j)$ . The challenger responds by running algorithm Set-Proxy-Re-Encryption-Key to generate the re-encryption key  $rk_{i \rightarrow j}$ .
- **Re-Encryption query** on  $(ID_i, ID_j, C_{ID_i})$ . The challenger responds by running algorithm Re-Encrypt to transform the ciphertext  $C_{ID_i}$  into the re-encrypted ciphertext  $C_{ID_j}$  using the re-encryption key  $rk_{i \rightarrow j}$ .
- **Decryption query** on  $(ID_i, C_{ID_i})$ . The challenger responds by running algorithm Decrypt to decrypt the ciphertext  $C_{ID_i}$  using the private key  $sk_{ID_i}$ .

**Challenge:** Once  $\mathcal{A}_{II}$  decides that Phase 1 is over, it outputs two equal length plaintexts  $m_0, m_1 \in \mathcal{M}$  and an identity  $ID^*$  of uncorrupted private key, on which it wishes to be challenged. In particular,  $\mathcal{A}_{II}$  is restricted to the choice of  $ID^*$  such that trivial decryption is not possible using keys extracted in Phase 1, e.g., by using re-encryption keys to transform  $C_{ID^*}$  into  $C_{ID_i}$  for which  $\mathcal{A}_{II}$  holds a decryption key. The challenger picks a random bit  $b \in \{0, 1\}$  and computes the challenge ciphertext  $C^* = \text{Encrypt}(m_b, \text{params}, ID^*, pk_{ID^*})$ . It sends  $C^*$  as the challenge to  $\mathcal{A}_{II}$ .

**Phase 2:**  $\mathcal{A}_{II}$  issues more queries  $q_{m+1}, \dots, q_n$  adaptively where  $q_i$  is one of:

- **Decryption query** on  $(ID_i, C_{ID_i})$ . If  $(ID_i, C_{ID_i})$  is not a challenge derivative, the challenger responds as in Phase 1.
- **Private Key query** on  $ID_i \neq ID^*$ . If a challenge derivative  $(ID_i, C_{ID_i})$  does not exist. The challenger responds as in Phase 1.
- **Public Key query** on  $ID_i$ . The challenger responds as in Phase 1.
- **Re-Encryption Key query** on  $(ID_i, ID_j)$ . The challenger responds as in Phase 1 except that  $ID_i = ID^*$  and  $\mathcal{A}_{II}$  has issued the Private Key query on  $ID_j$ .
- **Re-Encryption query** on  $(ID_i, ID_j, C_{ID_i})$ . The challenger responds as in Phase 1 except that  $\mathcal{A}_{II}$  has issued the Private Key query on  $ID_j$  and  $(ID_i, C_{ID_i})$  is a challenge derivative.

**Guess:** Finally,  $\mathcal{A}_{II}$  outputs a guess  $b' \in \{0, 1\}$ .  $\mathcal{A}_{II}$  wins if  $b' = b$ .

We define the advantage of  $\mathcal{A}_{II}$  in breaking the CL-PRE scheme as

$$\text{Adv}(\mathcal{A}_{II}) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

**Definition 3.** We say that a unidirectional CL-PRE scheme is  $(t, \epsilon)$ -adaptive chosen ciphertext (IND-CCA) secure against the Type II adversary if for any  $t$ -time adversary  $\mathcal{A}_{II}$  we have that  $\text{Adv}(\mathcal{A}_{II}) < \epsilon$ .

### 3 Chosen-Ciphertext Secure CL-PRE Scheme

In this section, we present the first construction of CL-PRE based on bilinear pairing, which gets rid of key escrow problem inherent in identity-based PRE schemes as well as the certificate revocation problem in traditional public key based PRE ones. We apply the transformation technique of Libert and Quisquater [14], which is a modification of Fujisaki-Okamoto conversion [12] by hashing the recipient's identity and public key along with the message and the random string in the encryption algorithm, to our construction. As pointed out in [14], Fujisaki-Okamoto conversion is not enough to prevent an attacker from breaking the scheme by using public key replacement oracles in the certificateless setting. We then prove that the proposed scheme is secure against an adaptive chosen ciphertext attack (IND-CCA) in the random oracle model.

#### 3.1 Bilinear Pairing and Complexity Assumption

We start by providing a brief overview of bilinear pairing and related computational problem on which our CL-PRE scheme is based. We use the following standard notation [8,9] to describe bilinear pairing:

1.  $\mathbb{G}_1$  and  $\mathbb{G}_2$  are two (multiplicative) cyclic groups of prime order  $q$ .
2.  $g$  is a generator of  $\mathbb{G}_1$ .
3.  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  is a bilinear map.

Let  $\mathbb{G}_1$  and  $\mathbb{G}_2$  be two groups as above. A bilinear map is a map  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  with the following properties:

1. Bilinear : for all  $u, v \in \mathbb{G}_1$  and  $a, b \in \mathbb{Z}_q^*$ , we have  $e(u^a, v^b) = e(u, v)^{ab}$ .
2. Non-degenerate :  $e(g, g) \neq 1_{\mathbb{G}_2}$ .

Note that  $e(\cdot, \cdot)$  is symmetric since  $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$ .

The security of our scheme relies on the intractability of the following problem [8]. The  $p$ -Bilinear Diffie Hellman Inversion ( $p$ -BDHI) problem is defined as follows: given a tuple  $(g, g^\alpha, \dots, g^{\alpha^p}) \in \mathbb{G}_1^{p+1}$  as input, compute  $e(g, g)^{1/\alpha} \in \mathbb{G}_2$ . An algorithm  $\mathcal{B}$  has advantage  $\epsilon$  in solving the  $p$ -BDHI problem if

$$\Pr[\mathcal{B}(g, g^\alpha, \dots, g^{\alpha^p}) = e(g, g)^{1/\alpha}] \geq \epsilon$$

Where the probability is over the random choice of generator  $g \in \mathbb{G}_1$ , the random choice of  $\alpha \in \mathbb{Z}_q^*$ , and the random bits of  $\mathcal{B}$ .

**Definition 4.** We say that the  $(t, p, \epsilon)$ -BDHI assumption holds in  $\mathbb{G}_1$  if no  $t$ -time algorithm has advantage at least  $\epsilon$  in solving the  $p$ -BDHI problem in  $\mathbb{G}_1$ .

### 3.2 Construction

The detailed description of the scheme is as follows:

**Setup:** Given security parameters  $k$  and  $k_0$ , this algorithm performs as follows:

1. Choose a  $k$ -bit prime number  $q$ , bilinear map groups  $(\mathbb{G}_1, \mathbb{G}_2)$  of order  $q$ , and random generators  $g, h \in \mathbb{G}_1$ .
2. Pick a KGC's master secret key  $\alpha \in \mathbb{Z}_q^*$  uniformly at random and compute a public key  $g_1 = g^\alpha \in \mathbb{G}_1$ .
3. Choose cryptographic hash functions  $H_1: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ ,  $H_2: \mathbb{G}_2^2 \rightarrow \{0, 1\}^{n+k_0}$ ,  $H_3: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ ,  $H_4: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ ,  $H_5: \{0, 1\}^* \rightarrow \mathbb{G}_1$ , and compute the group elements  $g_2 = e(g, g)$ ,  $g_3 = e(g, h) \in \mathbb{G}_2$ .

The public parameters are

$$\text{params} := \{\mathbb{G}_1, \mathbb{G}_2, e, g, h, g_1, g_2, g_3, H_1, H_2, H_3, H_4, H_5\}$$

where message space is  $\mathcal{M} := \{0, 1\}^n$ .

**Partial-Private-Key-Extract:** This algorithm takes **params** and an identifier  $\text{ID}_A$  for a user  $A$  as inputs, computes  $h_A = H_1(\text{ID}_A) \in \mathbb{Z}_q^*$  and extracts  $A$ 's partial private key  $d_A = g^{1/(\alpha+h_A)} \in \mathbb{G}_1$ .

**Set-Secret-Value:** Given **params** and  $\text{ID}_A$  as inputs, this algorithm selects  $a_1, a_2 \in \mathbb{Z}_q^*$  uniformly at random and sets  $x_A = (a_1, a_2) \in \mathbb{Z}_q^{*2}$  which is returned as user  $A$ 's secret value.

**Set-Private-Key:** Given **params**, user  $A$ 's partial private key  $d_A$  and secret value  $x_A$ , this algorithm returns the pair  $sk_A = (d_A, x_A) \in \mathbb{G}_1 \times \mathbb{Z}_q^{*2}$  as a private key.

**Set-Public-Key:** This algorithm takes **params** and user  $A$ 's secret value  $x_A$  as inputs, and produces user  $A$ 's public key  $pk_A = (g_3^{a_1}, g^{a_2}) \in \mathbb{G}_2 \times \mathbb{G}_1$ .

**Encrypt:** To encrypt a message  $m \in \mathcal{M}$  using the identifier  $\text{ID}_A$  and the public key  $pk_A = (g_3^{a_1}, g^{a_2})$ , the sender performs the following steps:

1. Check that  $pk_A$  is in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , if not output  $\perp$ .
2. Compute  $h_A = H_1(\text{ID}_A) \in \mathbb{Z}_q^*$ .
3. Choose a random  $\sigma \in \{0, 1\}^{k_0}$  and compute  $r = H_4(m||\sigma||\text{ID}_A||pk_A) \in \mathbb{Z}_q^*$ .
4. Compute the ciphertext  $C = (C_1, C_2, C_3, C_4)$ :

$$C = ((g^{h_A} \cdot g_1)^r, h^r, (m||\sigma) \oplus H_2(g_2^r || (g_3^{a_1})^r), u^r)$$

where  $u = H_5(\text{ID}_A||pk_A||C_1||C_2||C_3)$ .

**Set-Proxy-Re-Encryption-Key:** Given user  $B$ 's identifier  $\text{ID}_B$  and public key  $pk_B = (g_3^{b_1}, g^{b_2})$ , and user  $A$ 's identifier  $\text{ID}_A$  and public/private key pair  $(pk_A, sk_A)$ , this algorithm performs as follows:

1. Choose a random  $s \in \mathbb{Z}_q^*$  and compute  $\mu = H_3(g_2^s || \text{ID}_A || pk_A || \text{ID}_B || pk_B) \in \mathbb{Z}_q^*$ .
2. Compute  $h_A = H_1(\text{ID}_A) \in \mathbb{Z}_q^*$  and  $h_B = H_1(\text{ID}_B) \in \mathbb{Z}_q^*$ .
3. Set the proxy re-encryption key  $rk_{A \rightarrow B} = (rk_{A \rightarrow B}^{(1)}, rk_{A \rightarrow B}^{(2)}, rk_{A \rightarrow B}^{(3)}) = (g^{\mu/(\alpha+h_A)}, (g^{h_B} \cdot g_1)^s, (g^{b_2})^{a_1})$ .

**Re-Encrypt:** Given a re-encryption key  $rk_{A \rightarrow B}$  and a ciphertext  $C = (C_1, C_2, C_3, C_4)$  under the identifier  $\text{ID}_A$  and the public key  $pk_A$ , the proxy performs the following steps:

1. Set  $u = H_5(\text{ID}_A || pk_A || C_1 || C_2 || C_3)$  and compute  $h_A = H_1(\text{ID}_A) \in \mathbb{Z}_q^*$ .
2. Check that  $e(C_1, u) \stackrel{?}{=} e((g^{h_A} \cdot g_1), C_4)$  and  $e(C_2, u) \stackrel{?}{=} e(h, C_4)$ , if not output  $\perp$ .
3. Compute  $C'_1 = e(C_1, rk_{A \rightarrow B}^{(1)})$  and set  $C''_1 = rk_{A \rightarrow B}^{(2)}$ .
4. Compute  $C'_2 = e(C_2, rk_{A \rightarrow B}^{(3)})$ .
5. Set the new ciphertext  $C' = (C'_1, C''_1, C'_2, C_3, \text{ID}_A, pk_A)$ .

### Decrypt

- To decrypt non re-encrypted ciphertext  $C$ , the receiver  $A$  performs as follows:
  1. Parse  $C$  as  $(C_1, C_2, C_3, C_4)$ .
  2. Compute  $h_A = H_1(\text{ID}_A) \in \mathbb{Z}_q^*$ .
  3. Compute  $\omega = e(C_1, d_A) || e(g, C_2)^{a_1}$  and then  $(m || \sigma) = C_3 \oplus H_2(\omega)$ .
  4. If  $C_1 = (g^{h_A} \cdot g_1)^r$  and  $C_2 = h^r$ , where  $r = H_4(m || \sigma || \text{ID}_A || pk_A)$ , return  $m$  as the message. Otherwise, output  $\perp$ .
- To decrypt re-encrypted ciphertext  $C'$ , the receiver  $B$  performs as follows:
  1. Parse  $C'$  as  $(C'_1, C''_1, C'_2, C_3, \text{ID}_A, pk_A)$ .
  2. Compute  $\kappa = e(C'_1, d_B)$  and  $\mu = H_3(\kappa || \text{ID}_A || pk_A || \text{ID}_B || pk_B)$ .
  3. Compute  $\omega = C_3^{1/\mu} || C'_2^{1/b_2}$  and then  $(m || \sigma) = C_3 \oplus H_2(\omega)$ .
  4. If  $C'_1 = g_2^{\mu \cdot r}$  and  $C'_2 = (g_3^{a_1})^{b_2 \cdot r}$ , where  $r = H_4(m || \sigma || \text{ID}_A || pk_A)$ , return  $m$  as the message. Otherwise, output  $\perp$ .

The consistency of the construction is easy to check as follows:

- For the receiver  $A$ , we have

$$\begin{aligned}
 e(C_1, d_A) || e(g, C_2)^{a_1} &= e((g^{h_A} \cdot g^\alpha)^r, g^{1/(\alpha+h_A)}) || e(g, h^r)^{a_1} \\
 &= e(g^{(h_A+\alpha) \cdot r}, g^{1/(\alpha+h_A)}) || e(g, h)^{a_1 \cdot r} \\
 &= e(g, g)^r || e(g, h)^{a_1 \cdot r} \\
 &= g_2^r || (g_3^{a_1})^r
 \end{aligned}$$

- For the receiver  $B$ , we have

$$\begin{aligned}
 C_1^{1/\mu} || C'_2^{1/b_2} &= e(g, g)^{\mu \cdot r / \mu} || e(g, h)^{a_1 \cdot b_2 \cdot r / b_2} \\
 &= e(g, g)^r || e(g, h)^{a_1 \cdot r} \\
 &= g_2^r || (g_3^{a_1})^r
 \end{aligned}$$

### 3.3 Security Analysis

We prove the security of our CL-PRE scheme under the  $p$ -BDHI assumption described in Section 3.1.

**Theorem 1.** *Suppose the hash functions  $H_i$  ( $i = 1, 2, 3, 4, 5$ ) are modeled as random oracles, the above CL-PRE scheme is secure in the sense of IND-CCA described in Section 2.2 under the  $p$ -BDHI assumption.*

To prove the theorem, the proof separately considers both kinds of adversaries to establish the chosen ciphertext security of the above CL-PRE scheme. We now prove Theorem 1 by combining the following two lemmas.

**Lemma 1.** *Suppose that a Type I IND-CCA adversary  $\mathcal{A}_I$  has advantage  $\epsilon$  over our CL-PRE scheme when running time in a time  $\tau$ , asking at most  $q_{H_i}$  queries ( $H_i: i=1,2,3,4,5$ ), at most  $q_{EX}$  extraction queries, at most  $q_{SK}$  private key queries, at most  $q_{PK}$  public key queries, at most  $q_R$  replace queries, at most  $q_{RK}$  re-encryption key queries, at most  $q_{RE}$  re-encryption queries, and at most  $q_D$  decryption queries. There exists an algorithm  $\mathcal{B}$  to solve the  $p$ -BDHI problem with advantage*

$$\epsilon' \geq \frac{1}{q_{H_2}} \left( \frac{\epsilon}{2q_{H_1}} - \frac{q_{H_4} + (q_{H_2} + q_{H_4})q_D}{2^{k_0}} - \frac{q_{RE} + 2q_D}{q} \right)$$

The proof of Lemma 1 is found in Appendix A.

**Lemma 2.** *Suppose that a Type II IND-CCA adversary  $\mathcal{A}_{II}$  has advantage  $\epsilon$  over our CL-PRE scheme when running time in a time  $\tau$ , asking at most  $q_{H_i}$  queries ( $H_i: i=1,2,3,4,5$ ), at most  $q_{SK}$  private key queries, at most  $q_{PK}$  public key queries, at most  $q_{RK}$  re-encryption key queries, at most  $q_{RE}$  re-encryption queries, and at most  $q_D$  decryption queries. There exists an algorithm  $\mathcal{B}$  to solve the  $p$ -BDHI problem with advantage*

$$\epsilon' \geq \frac{1}{q_{H_2}} \left( \frac{\epsilon}{q_{H_1}} - \frac{q_{H_4} + (q_{H_2} + q_{H_4})q_D}{2^{k_0}} - \frac{q_{RE} + 2q_D}{q} \right)$$

The proof of Lemma 2 will be found in the full version of the paper due to the space limitation.

## 4 Conclusion

In this paper, we have introduced the notion of certificateless proxy re-encryption and also provided precise definitions for constructing secure certificateless proxy re-encryption schemes. We have presented a concrete scheme based on bilinear pairing, which enjoys the advantages of certificateless public key cryptography while providing the functionalities of proxy re-encryption. The proposed scheme is unidirectional and compatible with existing certificateless encryption deployments.

## Acknowledgements

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2008-521-D00454).

## References

1. Al-Riyami, S.S., Paterson, K.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Al-Riyami, S.S., Paterson, K.: CBE from CL-PKE: A generic construction and efficient scheme. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 398–415. Springer, Heidelberg (2005)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)* 9(1), 1–30 (2006)
4. Ateniese, G., Benson, K., Hohenberger, S.: Key-Private Proxy Re-Encryption, *Cryptography ePrint Archive, Report 2008/463* (2008)
5. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: *ACM CCS' 93*, pp. 62–73 (1993)
7. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
8. Boneh, D., Boyen, X.: Efficient selective-id secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
9. Boneh, D., Franklin, M.: Identity-based encryption from the weil paring. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
10. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption, *Cryptography ePrint Archive, Report 2007/171* (2007)
11. Deng, R.H., Weng, J., Liu, S., Chen, K.: Chosen-ciphertext secure proxy re-encryption without pairings. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 1–17. Springer, Heidelberg (2008)
12. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)
13. Green, M., Ateniese, G.: Identity-based proxy re-encryption, *Cryptography ePrint Archive, Report 2006/473* (2006)
14. Libert, B., Quisquater, J.: On constructing certificateless cryptosystem from identity based encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006)
15. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
16. Shao, J., Cao, Z.: CCA-secure proxy re-encryption without pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)

## A Proof of Lemma 1

*Proof.* Let  $\mathcal{A}_I$  be an adversary which has advantage  $\epsilon$  in attacking the IND-CCA security of our CL-PRE scheme. We show how to build an algorithm  $\mathcal{B}$  that uses  $\mathcal{A}_I$  to solve the  $p$ -BDHI problem in  $\mathbb{G}_1$ . Given as input a random tuple  $(g, g^\alpha, \dots, g^{\alpha^p}) \in \mathbb{G}_1^{p+1}$ ,  $\mathcal{B}$ 's goal is to extract  $e(g, g)^{1/\alpha}$  for some unknown  $\alpha$ . As in [1,14], we start by distinguishing two kinds of Type I adversaries:

- Type I-A adversary : may replace the public key for the challenge identity  $ID^*$  as some point, but cannot ask for the corresponding partial private key.
- Type I-B adversary : may ask for the partial private key of the challenge identity  $ID^*$  at some point, but cannot replace the corresponding public key.

Therefore, since  $\mathcal{B}$  should guess which kind of Type I adversary will be, it chooses a random bit  $c \in \{0, 1\}$ . If  $c = 0$ ,  $\mathcal{B}$  bets on a Type I-A adversary. Otherwise, bets on a Type I-B adversary.  $\mathcal{B}$  selects an index  $\ell$  with  $1 \leq \ell \leq q_{H_1}$  and an element  $\psi_\ell \in \mathbb{Z}_q^*$  uniformly at random.  $\mathcal{B}$  also chooses  $w_1, \dots, w_{\ell-1}, w_{\ell+1}, \dots, w_{q_{H_1}} \in \mathbb{Z}_q^*$  and computes  $\psi_i = \psi_\ell - w_i$  for  $i = 1, \dots, \ell - 1, \ell + 1, \dots, q_{H_1}$ .

Depending on the value of  $c$ ,  $\mathcal{B}$  now works by interacting with  $\mathcal{A}_I$  as follows:

**Setup:** To generate public parameters **params**,  $\mathcal{B}$  does the followings:

- Case  $c = 0$ .  $\mathcal{B}$  builds a generator  $h \in \mathbb{G}_1$  for which it knows  $p - 1$  pairs of the form  $(\psi_i, h^{1/(\psi_i + \beta)})$  for random  $i \neq \ell$ . This work is done as follows:
  1. Let  $f(z)$  be the polynomial  $f(z) = \prod_{i=1, i \neq \ell}^p (z + w_i)$ . Expand the terms of  $f(z)$  to get  $f(z) = \sum_{j=0}^{p-1} c_j z^j$ . The constant term  $c_0$  is non-zero.
  2. Compute  $h = \prod_{j=0}^{p-1} (g^{\alpha^j})^{c_j} = g^{f(\alpha)}$  and  $u = \prod_{j=1}^p (g^{\alpha^j})^{c_{j-1}} = g^{\alpha f(\alpha)}$ . Note that  $u = h^\alpha$ .
  3. Construct  $p - 1$  pairs  $(w_i, h_i = h^{1/(\alpha + w_i)})$  for any  $i \in \{1, \dots, p\} \setminus \{\ell\}$ . To see this, write  $f_i(z) = f(z)/(z + w_i) = \sum_{i=0}^{p-2} d_i z^i$ . Then  $h_i = h^{1/(\alpha + w_i)} = g^{f_i(\alpha)} = \prod_{i=0}^{p-2} (g^{\alpha^i})^{d_i}$ .
  4. Compute the public key of the KGC as  $g_1 = u^{-1} \cdot h^{-\psi_\ell} = h^{-\alpha - \psi_\ell}$  for unknown master key  $mk := \beta = -\alpha - \psi_\ell \in \mathbb{Z}_q^*$ , and set  $v = g^\alpha$  as another generator.
  5. Set  $(\psi_i, h^{1/(\psi_i + \beta)}) = (\psi_i, h_i^{-1})$  for all  $i \in \{1, \dots, p\} \setminus \{\ell\}$ .
- Case  $c = 1$ .  $\mathcal{B}$  builds generators  $h = g^{\alpha^{p-1}}, v = g^\alpha \in \mathbb{G}_1$ . Then, it chooses a random  $\beta \in \mathbb{Z}_q^*$  as the KGC's master key and computes the corresponding public key  $g_1 = h^\beta \in \mathbb{Z}_q^*$ .

Finally,  $\mathcal{B}$  generates public parameters **params** =  $(\mathbb{G}_1, \mathbb{G}_2, e, h, v, g_1, e(h, h), e(h, v), H_1, H_2, H_3, H_4, H_5)$  and gives **params** to  $\mathcal{A}_I$ . Here  $H_i (i = 1, 2, 3, 4, 5)$  are random oracles controlled by  $\mathcal{B}$  as described below:

**$H_1$ -queries:** For query on input  $ID_i$ ,  $\mathcal{B}$  responds with  $\psi_i$  and increments an index  $i$ .

**$H_2$ -queries:** For query on input  $\gamma_i \in \mathbb{G}_2^2$  of the form  $(\gamma_{i,1} || \gamma_{i,2})$ ,  $\mathcal{B}$  maintains a  $H_2$ -list of tuples  $(\gamma_i, \zeta_i)$  as below:

1. If  $\gamma_i$  appears in the  $H_2$ -list, then  $\mathcal{B}$  responds with  $\zeta_i$ .

2. Otherwise,  $\mathcal{B}$  chooses  $\zeta_i \in \{0, 1\}^{n+k_0}$  uniformly at random and adds  $(\gamma_i, \zeta_i)$  to the  $H_2$ -list and returns  $\zeta_i$  to  $\mathcal{A}_I$ .

**$H_3$ -queries:** For query on input  $\tau_{i,j} \in \{0, 1\}^*$  of the form  $(\kappa_{i,j} || \text{ID}_i || pk_i || \text{ID}_j || pk_j)$ ,  $\mathcal{B}$  maintains a  $H_3$ -list of tuples  $(\tau_{i,j}, \mu_{i,j})$  as below:

1. If  $\tau_{i,j}$  appears in the  $H_3$ -list, then  $\mathcal{B}$  responds with  $\mu_{i,j}$ .
2. Otherwise,  $\mathcal{B}$  chooses  $\mu_{i,j} \in \mathbb{Z}_q^*$  uniformly at random and adds  $(\tau_{i,j}, \mu_{i,j})$  to the  $H_3$ -list and returns  $\mu_{i,j}$  to  $\mathcal{A}_I$ .

**$H_4$ -queries:** For query on input  $\lambda_i \in \{0, 1\}^*$  of the form  $(m_i || \sigma_i || \text{ID}_i || pk_i)$ ,  $\mathcal{B}$  maintains a  $H_4$ -list of tuples  $(m_i, \sigma_i, \text{ID}_i, pk_i, r_i)$  as below:

1. If  $\lambda_i$  appears in the  $H_4$ -list, then  $\mathcal{B}$  responds with  $r_i$ .
2. Otherwise,  $\mathcal{B}$  chooses  $r_i \in \mathbb{Z}_q^*$  uniformly at random and adds  $(m_i, \sigma_i, \text{ID}_i, pk_i, r_i)$  to the  $H_4$ -list and returns  $r_i$  to  $\mathcal{A}_I$ .

**$H_5$ -queries:** For query on input  $\eta_i \in \{0, 1\}^*$  of the form  $(\text{ID} || pk || C_1 || C_2 || C_3)$ ,  $\mathcal{B}$  maintains a  $H_5$ -list of tuples  $(\eta_i, y_i, \rho_i)$  as below:

1. If  $\eta_i$  appears in the  $H_5$ -list, then  $\mathcal{B}$  responds with  $\rho_i$ .
2. Otherwise,  $\mathcal{B}$  chooses a random  $y_i \in \mathbb{Z}_q^*$  and adds  $(\eta_i, y_i, \rho_i = v^{y_i})$  to the  $H_5$ -list and returns  $\rho_i$  to  $\mathcal{A}_I$ .

**Phase 1:**  $\mathcal{A}_I$  issues queries  $q_1, \dots, q_m$  where query  $q_i$  is one of:

- Extraction queries.  $\mathcal{B}$  responds to  $\mathcal{A}_I$ 's query on  $\text{ID}_i$  as follows:
  - Case  $c = 0$ . If  $i = \ell$ ,  $\mathcal{B}$  stops the simulation and outputs "failure". Otherwise, it responds with  $h^{1/(\psi_i + \beta)}$  corresponding to  $H_1(\text{ID}_i) = \psi_i$ .
  - Case  $c = 1$ .  $\mathcal{B}$  responds with  $d_i$  by running the Partial-Private-Key-Extract algorithm as it knows the master key.
- Public key queries. For a public key query on  $\text{ID}_i$ ,  $\mathcal{B}$  checks whether the  $pk$ -list contains a tuple  $(\text{ID}_i, x_i = (x_{i,1}, x_{i,2}), pk_i)$  for this input.
  1. If it does, the previously defined value is returned.
  2. Otherwise,  $\mathcal{B}$  does the followings:
    - Case  $c = 0$ .  $\mathcal{B}$  picks random  $x_{i,1}, x_{i,2} \in \mathbb{Z}_q^*$  and computes  $pk_i = (e(h, v)^{x_{i,1}}, h^{x_{i,2}})$ , then returns it to  $\mathcal{A}_I$ .
    - Case  $c = 1$ . If  $i = \ell$ ,  $\mathcal{B}$  picks a random  $\delta \in \mathbb{G}_1$  and responds with  $pk_i = (e(g, g), \delta)$  which is equal to  $(e(h, v)^{1/\alpha^p}, h^x)$  for unknown  $(1/\alpha^p, x)$ . Otherwise,  $\mathcal{B}$  responds with  $pk_i$  as in the simulation  $c = 0$ .
  3.  $\mathcal{B}$  then adds  $(\text{ID}_i, x_i, pk_i)$  to the  $pk$ -list (If  $i = \ell$  and  $c = 1$ ,  $x_i$  is  $\perp$  as an unknown value).
- Private key queries. For query on input  $\text{ID}_i$ ,  $\mathcal{B}$  performs as follows:
  1. If  $i = \ell$ ,  $\mathcal{B}$  stops the simulation and outputs "failure".
  2. If  $i \neq \ell$ ,  $\mathcal{B}$  recovers the corresponding tuple from the  $pk$ -list. If the  $pk$ -list contain the tuple  $(\text{ID}_i, x_i, pk_i)$ ,
    - Case  $c = 0$ .  $\mathcal{B}$  sets  $sk_i = (d_i, x_i)$  as it knows the partial private key and sends it to  $\mathcal{A}_I$ .
    - Case  $c = 1$ .  $\mathcal{B}$  extracts the partial private key itself using the master key and returns  $sk_i = (d_i, x_i)$  to  $\mathcal{A}_I$ .
  3. Otherwise,  $\mathcal{B}$  makes a public key query on  $\text{ID}_i$ , then simulates as the above process.

- Replace public key queries. When  $\mathcal{A}_I$  queries on input  $(ID_i, pk'_i)$ ,  $\mathcal{B}$  does the followings:
  1. If  $i = \ell$  and  $c = 1$ ,  $\mathcal{B}$  stops the simulation and outputs "failure".
  2. Otherwise,  $\mathcal{B}$  checks that  $pk'_i$  is in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . If does, replaces the corresponding tuple in the  $pk$ -list with  $(ID_i, \perp, pk'_i)$ .
- Re-encryption key queries.  $\mathcal{B}$  responds to  $\mathcal{A}_I$ 's query on  $(ID_i, ID_j)$  as follows:
  1. If  $i = \ell$ ,  $\mathcal{B}$  stops the simulation and outputs "failure".
  2. If  $i \neq \ell$ ,  $\mathcal{B}$  checks whether the  $rk$ -list contains a tuple  $(ID_i, ID_j, rk_{i \rightarrow j}, s_{i,j})$  for this input.
    - If it does, the previously defined value is returned.
    - Otherwise,  $\mathcal{B}$  does followings:
      - (a) Recover the corresponding tuples  $(ID_i, x_i = (x_{i,1}, x_{i,2}), pk_i)$  and  $(ID_j, x_j = (x_{j,1}, x_{j,2}), pk_j)$  from the  $pk$ -list.
      - (b) Choose a random  $s_{i,j} \in \mathbb{Z}_q^*$  and evaluate  $H_3(e(h, h)^{s_{i,j}} || ID_i || pk_i || ID_j || pk_j)$  to obtain  $\mu_{i,j}$ .
      - (c) Simulate an extraction query on  $ID_i$  to obtain the corresponding partial private key  $d_i = h^{1/(\beta+\psi_i)}$ .
      - (d) If  $j = \ell$  and  $c = 1$ , set the re-encryption key  $rk_{i \rightarrow j} = (rk_{i \rightarrow j}^{(1)}, rk_{i \rightarrow j}^{(2)}, rk_{i \rightarrow j}^{(3)}) = (h^{\mu_{i,j}/(\beta+\psi_i)}, (h^{\psi_j} \cdot g_1)^{s_{i,j}}, \delta^{x_{i,1}})$ . Otherwise, set the re-encryption key  $rk_{i \rightarrow j} = (rk_{i \rightarrow j}^{(1)}, rk_{i \rightarrow j}^{(2)}, rk_{i \rightarrow j}^{(3)}) = (h^{\mu_{i,j}/(\beta+\psi_i)}, (h^{\psi_j} \cdot g_1)^{s_{i,j}}, (h^{x_{j,2}})^{x_{i,1}})$ .
      - (e) Return the re-encryption key  $rk_{i \rightarrow j}$  to  $\mathcal{A}_I$  and add  $(ID_i, ID_j, rk_{i \rightarrow j}, s_{i,j})$  to the  $rk$ -list.
- Re-encryption queries. For query on input  $(ID_i, ID_j, C)$ ,  $\mathcal{B}$  parses  $C$  as  $(C_1, C_2, C_3, C_4)$ . Then,  $\mathcal{B}$  recovers the corresponding tuples  $(ID_i, x_i, pk_i)$  and  $(ID_j, x_j, pk_j)$  from the  $pk$ -list and evaluates  $H_5(ID_i || pk_i || C_1 || C_2 || C_3)$  to obtain  $\rho_i$ .  $\mathcal{B}$  checks that  $e(C_1, \rho_i) \stackrel{?}{=} e((h^{\psi_i} \cdot g_1), C_4)$  and  $e(C_2, \rho_i) \stackrel{?}{=} e(v, C_4)$ . If not,  $\mathcal{B}$  returns  $\perp$ . Otherwise,
  1. If  $i = \ell$ ,  $\mathcal{B}$  does the followings:
    - Look up the corresponding tuple  $(m, \sigma, ID_i, pk_i, r)$  in the  $H_4$ -list such that  $C_1 = (h^{\psi_i} \cdot g_1)^r$  and  $C_2 = v^r$ . If there exists no such tuple, return  $\perp$ .
    - Choose a random  $s \in \mathbb{Z}_q^*$  and evaluate  $H_3(e(h, h)^s || ID_i || pk_i || ID_j || pk_j)$  to obtain  $\mu_{i,j}$ .
    - Set  $C'_1 = e(h, h)^{\mu_{i,j} \cdot r}$ ,  $C''_1 = (h^{\psi_j} \cdot g_1)^s$ , and  $C'_2 = e(h, v)^{x_{i,1} \cdot x_{j,2} \cdot r}$  in case  $c = 0$  or  $C'_2 = e(g, g)^{x_{j,2} \cdot r}$  in case  $c = 1$ .
    - Output  $C' = (C'_1, C''_1, C'_2, C_3, ID_i, pk_i)$  to  $\mathcal{A}_I$ .
  2. If  $i \neq \ell$ ,  $\mathcal{B}$  simulates a re-encryption key query on  $(ID_i, ID_j)$  to obtain  $rk_{i \rightarrow j}$ , then outputs  $C' = \text{Re-Encrypt}(params, rk_{i \rightarrow j}, C)$  to  $\mathcal{A}_I$ .
- Decryption queries.  $\mathcal{B}$  responds to  $\mathcal{A}_I$ 's query on  $(ID_i, C)$  as below:
  1. If  $i = \ell$ ,  $\mathcal{B}$  parses  $C$ ,
    - Case  $C = (C_1, C_2, C_3, C_4)$ .  $\mathcal{B}$  does followings:

- (a) Look up the corresponding tuples  $(m, \sigma, \text{ID}_i, pk_i, r)$  and  $(\gamma, \zeta)$  in the  $H_4$ -list and the  $H_2$ -list, respectively, such that

$$C_1 = (h^{\psi_i} \cdot g_1)^r, \quad C_2 = v^r, \quad C_3 = (m \parallel \sigma) \oplus \zeta$$

If there exist no such tuples, return  $\perp$ .

- (b) Check that  $C_4 \stackrel{?}{=} H_5(\text{ID}_i \parallel pk_i \parallel C_1 \parallel C_2 \parallel C_3)^r$ . If not return  $\perp$ , otherwise output  $m$  to  $\mathcal{A}_I$ .
- $C = (C'_1, C''_1, C'_2, C_3, \text{ID}_j, pk_j)$ .  $\mathcal{B}$  does followings:
 

(a) Recover the corresponding tuple  $(\text{ID}_j, \text{ID}_i, rk_{j \rightarrow i}, s_{j,i})$  from the  $rk$ -list.

(b) Check that  $C''_1 \stackrel{?}{=} rk_{j \rightarrow i}^{(2)}$ . If not return  $\perp$ .

(c) Recover the corresponding tuples  $(\text{ID}_j, x_j, pk_j)$  and  $(\text{ID}_i, x_i, pk_i)$  from the  $pk$ -list, then evaluate  $H_3(e(h, h)^{s_{j,i}} \parallel \text{ID}_j \parallel pk_j \parallel \text{ID}_i \parallel pk_i)$  to obtain  $\mu$ .

(d) Look up the corresponding tuples  $(m, \sigma, \text{ID}_j, pk_j, r)$  and  $(\gamma, \zeta)$  in the  $H_4$ -list and the  $H_2$ -list, respectively, such that

$$C'_1 = e(h, h)^{\mu \cdot r}, \quad C'_2 = e(h, v)^{x_{j,1} \cdot x_{i,2} \cdot r}, \quad C_3 = (m \parallel \sigma) \oplus \zeta$$

in case  $c = 0$  or

$$C'_1 = e(h, h)^{\mu \cdot r}, \quad C'_2 = e(\delta, v)^{x_{j,1} \cdot r}, \quad C_3 = (m \parallel \sigma) \oplus \zeta$$

in case  $c = 1$ . If there exist no such tuples, return  $\perp$ . Otherwise, output  $m$  to  $\mathcal{A}_I$ .

2. If  $i \neq \ell$ ,  $\mathcal{B}$  simulates a private key query on  $\text{ID}_i$  to obtain  $sk_i$ , then outputs  $m = \text{Decrypt}(params, sk_i, C)$  to  $\mathcal{A}_I$ .

**Challenge:** Once  $\mathcal{A}_I$  decides that Phase 1 is over, it outputs two messages  $m_0, m_1 \in \mathcal{M}$  and  $\text{ID}^*$  on which it wishes to be challenged. If  $\text{ID}^* \neq \text{ID}_\ell$ ,  $\mathcal{B}$  stops the simulation and outputs "failure". Otherwise,  $\mathcal{B}$  picks a random bit  $b \in \{0, 1\}$  and a random  $\sigma^* \in \{0, 1\}^{k_0}$ . Finally,  $\mathcal{B}$  builds a challenge ciphertext for  $\text{ID}^*$  and the current public key  $pk_{\text{ID}^*}$  according to the value of  $c$ .

- Case  $c = 0$ .  $\mathcal{B}$  picks a random  $l \in \mathbb{Z}_q^*$  and a random string  $\xi \in \{0, 1\}^{n+k_0}$ .  $\mathcal{B}$  then defines the challenged ciphertext to be

$$C^* = (C_1^*, C_2^*, C_3^*, C_4^*) = (h^{-l}, g^l, \xi, (g^y)^l)$$

where  $y$  is obtained from evaluating  $H_5(\text{ID}^* \parallel pk_{\text{ID}^*} \parallel C_1^* \parallel C_2^* \parallel C_3^*)$ .

Define  $r = l/\alpha$ . Since  $\beta = -\alpha - \psi_\ell$ , we have that

$$\begin{aligned} C_1^* &= h^{-l} = (h^{-\alpha})^{l/\alpha} = h^{(\psi_\ell + \beta) \cdot r} = h^{\psi_\ell \cdot r} \cdot h^{\beta \cdot r} = (h^{H_1(\text{ID}^*)} \cdot g_1)^r, \\ C_2^* &= g^l = (g^\alpha)^{l/\alpha} = v^r, \\ C_4^* &= (g^y)^l = (g^{\alpha \cdot y})^{l/\alpha} = (v^y)^{l/\alpha} = \rho^r \end{aligned}$$

Without issuing a  $H_2$ -query on the input  $(e(h, h)^r \parallel e(h, v)^{x_{\ell,1} \cdot r})$ ,  $\mathcal{A}_I$  is unable to recognize that  $C^*$  is not an encryption of  $m_0$  or  $m_1$  and such an event would provide  $\mathcal{B}$  with the searched  $p$ -BDHI solution.

- Case  $c = 1$ .  $\mathcal{B}$  picks a random  $l \in \mathbb{Z}_q^*$  and a random string  $\xi \in \{0, 1\}^{n+k_0}$ .  $\mathcal{B}$  then defines the challenged ciphertext to be

$$C^* = (C_1^*, C_2^*, C_3^*, C_4^*) = ((g^{\alpha^{p-2}})^{\psi_\ell \cdot l} \cdot (g^{\alpha^{p-2}})^{\beta \cdot l}, g^l, \xi, (g^y)^l)$$

where  $y$  is obtained from evaluating  $H_5(\text{ID}^* || pk_{ID}^* || C_1^* || C_2^* || C_3^*)$ . Define  $r = l/\alpha$ . Then, we have that

$$\begin{aligned} C_1^* &= (g^{\alpha^{p-2}})^{\psi_\ell \cdot l} \cdot (g^{\alpha^{p-2}})^{\beta \cdot l} = (h^{\psi_\ell})^{l/\alpha} \cdot (h^\beta)^{l/\alpha} = (h^{H_1(ID^*)} \cdot g_1)^r, \\ C_2^* &= g^l = (g^\alpha)^{l/\alpha} = v^r, \\ C_4^* &= (g^y)^l = (g^{\alpha \cdot y})^{l/\alpha} = (v^y)^{l/\alpha} = \rho^r \end{aligned}$$

Without issuing a  $H_2$ -query on the input  $(e(h, h)^r || e(g, g)^r)$ ,  $\mathcal{A}_I$  is unable to recognize that  $C^*$  is not an encryption of  $m_0$  or  $m_1$  and such an event would provide  $\mathcal{B}$  with the searched  $p$ -BDHI solution.

Finally,  $\mathcal{B}$  returns  $C^*$  as the challenge ciphertext to  $\mathcal{A}_I$ .

**Phase 2:**  $\mathcal{A}_I$  issues more queries.  $\mathcal{B}$  responds as Phase 1 with the restrictions described in Section 2.2.

**Guess:**  $\mathcal{A}_I$  outputs its guess  $b' \in \{0, 1\}$  which is ignored.

To produce a result,  $\mathcal{B}$  chooses a random tuple  $(\gamma_1 || \gamma_2, \zeta)$  from the  $H_2$ -list with probability  $1/q_{H_2}$ . We distinguish two cases according to the value of  $c$ .

- Case  $c = 0$ . We have  $\gamma_1 = e(h, h)^r = e(g, g)^{f(\alpha)^2 \cdot l/\alpha}$  and  $\mathcal{B}$  can extract the  $p$ -BDHI solution by noting that, if  $\gamma^* = e(g, g)^{1/\alpha}$ , then

$$\gamma_1 = e(g, g)^{f(\alpha)^2 \cdot l/\alpha} = \left( \gamma^{*(c_0^2)} \cdot e\left(\prod_{i=0}^{p-2} g^{c_{i+1} \cdot \alpha^i}, \prod_{j=0}^{p-1} g^{c_j \cdot \alpha^j}\right) \cdot e\left(\prod_{i=0}^{p-2} g^{c_{i+1} \cdot \alpha^i}, g^{c_0}\right) \right)^l$$

- Case  $c = 1$ . We have  $\gamma_2 = e(g, g)^r = e(g, g)^{l/\alpha}$  and  $\mathcal{B}$  can extract the  $p$ -BDHI solution by noting that  $\gamma^* = \gamma_2^{1/l}$ .

Let us analyze the simulation. The main idea of our analysis is borrowed from [5,11]. We first evaluate the simulations of the random oracles. From the constructions of  $H_1, H_3$  and  $H_5$ , It is obvious that the simulations of  $H_1, H_3$  and  $H_5$  are perfect. Let  $\text{AskH}_2^*$  be the event that  $(e(h, h)^r || e(h, v)^{x_\ell, 1^r})$  in case  $c = 0$  or  $(e(h, h)^r || e(g, g)^r)$  in case  $c = 1$  has been queried to  $H_2$ , and  $\text{AskH}_4^*$  be the event that  $(m_b || \sigma^* || \text{ID}^* || pk_{ID}^*)$  has been queried to  $H_4$ . The simulations of  $H_2$  and  $H_4$  are also perfect, as long as  $\text{AskH}_2^*$  and  $\text{AskH}_4^*$  did not occur, where  $b$  and  $\sigma^*$  are chosen by  $\mathcal{B}$  in the Challenge phase.

The simulated challenge ciphertext is identically distributed as the real one from the construction.

Next, we analyze the simulation of the re-encryption oracle. This simulation is also perfect, unless  $\mathcal{A}_I$  can submit valid original ciphertexts without querying

hash function  $H_4$ . Let  $\text{ReEncErr}$  denote this event. However, since  $H_4$  acts as a random oracle and  $\mathcal{A}_I$  issues at most  $q_{RE}$  re-encryption queries, we have  $\Pr[\text{ReEncErr}] \leq \frac{q_{RE}}{q}$ .

The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. However, these errors are not significant as shown below: Suppose a ciphertext  $C$  has been queried to the decryption oracle. Even if  $C$  is valid, there is a possibility that  $C$  can be produced without querying  $(g_2^r || (g_3^{x_1})^r)$  to  $H_2$ , where  $r = H_4(m || \sigma || \text{ID} || pk)$ . Let  $\text{Valid}$  be an event that  $C$  is valid. Let  $\text{AskH}_2$  and  $\text{AskH}_4$  be the events that  $(g_2^r || (g_3^{x_1})^r)$  has been queried to  $H_2$  and  $(m || \sigma || \text{ID} || pk)$  has been queried to  $H_4$ , respectively. We then have that

$$\begin{aligned} \Pr[\text{Valid} | \neg \text{AskH}_2] &= \Pr[\text{Valid} \wedge \text{AskH}_4 | \neg \text{AskH}_2] + \Pr[\text{Valid} \wedge \neg \text{AskH}_4 | \neg \text{AskH}_2] \\ &\leq \Pr[\text{AskH}_4 | \neg \text{AskH}_2] + \Pr[\text{Valid} | \neg \text{AskH}_4 \wedge \neg \text{AskH}_2] \\ &\leq \frac{q_{H_4}}{2^{k_0}} + \frac{1}{q} \end{aligned}$$

and similarly we have  $\Pr[\text{Valid} | \neg \text{AskH}_4] \leq \frac{q_{H_2}}{2^{k_0}} + \frac{1}{q}$ . Therefore, we have that

$$\begin{aligned} \Pr[\text{Valid} | (\neg \text{AskH}_2 \vee \neg \text{AskH}_4)] &\leq \Pr[\text{Valid} | \neg \text{AskH}_2] + \Pr[\text{Valid} | \neg \text{AskH}_4] \\ &\leq \frac{q_{H_2} + q_{H_4}}{2^{k_0}} + \frac{2}{q} \end{aligned}$$

Let  $\text{DecErr}$  be the event that  $\Pr[\text{Valid} | (\neg \text{AskH}_2 \vee \neg \text{AskH}_4)]$  happens during the entire simulation. Then, since  $\mathcal{A}_I$  issues at most  $q_D$  decryption queries, we have

$$\Pr[\text{DecErr}] \leq \frac{(q_{H_2} + q_{H_4})q_D}{2^{k_0}} + \frac{2q_D}{q}$$

If  $\mathcal{B}$  does not abort during the simulation, it is clear that the simulations of the other oracles are perfect. Now, let us calculate the probability that  $\mathcal{B}$  does not abort during the simulation. Let  $\neg \text{Abort}$  denote this event. As the proof technique in [1,14], we define the following events:

- $\mathcal{H}$ :  $\mathcal{A}_I$  chooses  $\text{ID}_\ell$  as the challenge identity  $\text{ID}^*$ .
- $\mathcal{F}_0$ :  $\mathcal{A}_I$  extracts the partial private key for  $\text{ID}_\ell$  at some point.
- $\mathcal{F}_1$ :  $\mathcal{A}_I$  replaces the public key of  $\text{ID}_\ell$  at some point.

Following the above events,  $\mathcal{B}$  could abort for one of the following reasons:

1. Because  $c = 0$  and the event  $\mathcal{F}_0$  occurs during the simulation.
2. Because  $c = 1$  and the event  $\mathcal{F}_1$  occurs during the simulation.
3. Because of a private key query for the identity  $\text{ID}_\ell$ . Let  $\mathcal{F}_2$  denote this event.
4. Because of a re-encryption key query for  $(\text{ID}_\ell, \text{ID}_j)$ . Let  $\mathcal{F}_3$  denote this event.
5. Because  $\mathcal{A}_I$  chooses a challenge identity  $\text{ID}^* \neq \text{ID}_\ell$ . Let  $\neg \mathcal{H}$  denote this event.

We also name the event  $(c = i) \wedge \mathcal{F}_i$  as  $\mathcal{H}_i$  for  $i = 0, 1$ . It is obvious that the event  $\mathcal{H}$  implies the events  $\neg\mathcal{F}_2$  and  $\neg\mathcal{F}_3$ . Therefore, the probability that  $\mathcal{B}$  does not abort during the simulation is

$$\begin{aligned} \Pr[\neg\text{Abort}] &= \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 \wedge \neg\mathcal{F}_2 \wedge \neg\mathcal{F}_3 \wedge \mathcal{H}] \\ &= \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 \wedge \mathcal{H}] = \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}] \cdot \Pr[\mathcal{H}] \\ &= \frac{1}{q_{H_1}} \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}] \end{aligned}$$

Since the events  $\mathcal{H}_0$  and  $\mathcal{H}_1$  are mutually exclusive, we have that

$$\Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}] = 1 - \Pr[\mathcal{H}_0 | \mathcal{H}] - \Pr[\mathcal{H}_1 | \mathcal{H}]$$

On the other hand, we have that

$$\Pr[\mathcal{H}_i | \mathcal{H}] = \Pr[(c = i) \wedge \mathcal{F}_i | \mathcal{H}] = \frac{1}{2} \Pr[\mathcal{F}_i | \mathcal{H}]$$

because the event  $\mathcal{F}_i | \mathcal{H}$  is independent of the event  $c = i$  for  $i = 0, 1$ . Finally, as  $\Pr[\mathcal{F}_0 \wedge \mathcal{F}_1 | \mathcal{H}] = 0$ , this implies that  $\Pr[\mathcal{F}_0 | \mathcal{H}] + \Pr[\mathcal{F}_1 | \mathcal{H}] \leq 1$ . Therefore, we have that

$$\begin{aligned} \Pr[\neg\text{Abort}] &= \frac{1}{q_{H_1}} \left( 1 - \frac{1}{2} \Pr[\mathcal{F}_0 | \mathcal{H}] - \frac{1}{2} \Pr[\mathcal{F}_1 | \mathcal{H}] \right) \\ &\geq \frac{1}{2q_{H_1}} \end{aligned}$$

Let  $\text{Good}$  denote the event  $(\text{AskH}_2^* \vee (\text{AskH}_4^* | \neg\text{AskH}_2^*) \vee \text{ReEncErr} \vee \text{DecErr}) | \neg\text{Abort}$ . If  $\text{Good}$  does not happen, due to the randomness of the output of the random oracle  $H_2$ , it is obvious that  $\mathcal{A}_I$  cannot gain any advantage greater than  $1/2$  in guessing  $b$ . Namely, we have  $\Pr[b = b' | \neg\text{Good}] = \frac{1}{2}$ . Hence, by splitting  $\Pr[b = b']$ , we have that

$$\begin{aligned} \Pr[b = b'] &= \Pr[b = b' | \neg\text{Good}] \Pr[\neg\text{Good}] + \Pr[b = b' | \text{Good}] \Pr[\text{Good}] \\ &\leq \frac{1}{2} \Pr[\neg\text{Good}] + \Pr[\text{Good}] = \frac{1}{2} + \frac{1}{2} \Pr[\text{Good}] \end{aligned}$$

and

$$\Pr[b = b'] \geq \Pr[b = b' | \neg\text{Good}] \Pr[\neg\text{Good}] = \frac{1}{2} - \frac{1}{2} \Pr[\text{Good}]$$

We then have that

$$\left| \Pr[b = b'] - \frac{1}{2} \right| \leq \frac{1}{2} \Pr[\text{Good}]$$

By definition of the advantage for the IND-CCA adversary  $\mathcal{A}_I$ , we have that

$$\begin{aligned} \epsilon &= |2 \times \Pr[b = b'] - 1| \\ &\leq \Pr[\text{Good}] = \Pr[(\text{AskH}_2^* \vee (\text{AskH}_4^* | \neg\text{AskH}_2^*) \vee \text{ReEncErr} \vee \text{DecErr}) | \neg\text{Abort}] \\ &= \frac{(\Pr[\text{AskH}_2^*] + \Pr[\text{AskH}_4^* | \neg\text{AskH}_2^*] + \Pr[\text{ReEncErr}] + \Pr[\text{DecErr}])}{\Pr[\neg\text{Abort}]} \end{aligned}$$

Since  $\Pr[\text{AskH}_4^* | \neg \text{AskH}_2^*] \leq \frac{q_{H_4}}{2^{k_0}}$ ,  $\Pr[\text{ReEncErr}] \leq \frac{q_{RE}}{q}$ ,  $\Pr[\text{DecErr}] \leq \frac{(q_{H_2} + q_{H_4})q_D}{2^{k_0}} + \frac{2q_D}{q}$  and  $\Pr[\neg \text{Abort}] \geq \frac{1}{2q_{H_1}}$ , we obtain

$$\begin{aligned} \Pr[\text{AskH}_2^*] &\geq \epsilon \cdot \Pr[\neg \text{Abort}] - \Pr[\text{AskH}_4^* | \neg \text{AskH}_2^*] - \Pr[\text{ReEncErr}] - \Pr[\text{DecErr}] \\ &\geq \frac{\epsilon}{2q_{H_1}} - \frac{q_{H_4}}{2^{k_0}} - \frac{q_{RE}}{q} - \frac{(q_{H_2} + q_{H_4})q_D}{2^{k_0}} - \frac{2q_D}{q} \\ &\geq \frac{\epsilon}{2q_{H_1}} - \frac{q_{H_4} + (q_{H_2} + q_{H_4})q_D}{2^{k_0}} - \frac{q_{RE} + 2q_D}{q} \end{aligned}$$

Finally, notice that  $\mathcal{B}$  solves the  $p$ -BDHI problem with probability  $1/q_{H_2}$  if  $\text{AskH}_2^*$  happens. Consequently, we obtain

$$\epsilon' \geq \frac{1}{q_{H_2}} \Pr[\text{AskH}_2^*] \geq \frac{1}{q_{H_2}} \left( \frac{\epsilon}{2q_{H_1}} - \frac{q_{H_4} + (q_{H_2} + q_{H_4})q_D}{2^{k_0}} - \frac{q_{RE} + 2q_D}{q} \right)$$

□