Multiple and Unlinkable Public Key Encryption without Certificates

Soyoung Park[1],Sang-Ho Lee[1] and Joohan Lee[1]

[1] University of Central Florida, 4000 Blvd Central Florida, FL, USA
[2] Ewha Womans University, 11-1 Daehyundong, Seodaemoongu, Seoul, Korea

Abstract. We newly propose a multiple and unlinkable identity-based public key encryption scheme. Unlike the traditional public key encryption and identity-based encryption schemes, our scheme allows the use of a various number of identity-based public keys in different groups or applications while keeping a single decryption key so that the decryption key can decrypt every ciphertexts encrypted with those public keys. Also our scheme removes the use of certificates as well as the key escrow problem so it is functional and practical. Since our public keys are unlinkable, the user's privacy can be protected from attackers who collect and trace the user information and behavior using the known public keys. Furthermore, we suggest a decryption key renewal protocol to strengthen the security of the single decryption key. Finally, we prove the security of our scheme against the adaptive chosen-ciphertext attack under the random oracle model.

1 Introduction

1.1 Background and Related Work

With the ever increasing Internet user population, widely available large scale Internet-based services and the advent of pervasive computing environment, more and more daily human activities are relying on various types of communication devices and applications over wired or wireless networks all the time. Tremendous personal data have been being transmitted over the networks even without our consciousness. Especially, the pervasive computing environment makes attackers capable of gathering and accessing private and sensitive information more easily. So we need stronger and more flexible security equipments and protocols which can protect privacy against those attackers.

In this paper we propose a new type of public key encryption scheme designed to provide data confidentiality and personal privacy in upcoming ubiquitous environment where heterogeneous networks and services coexist or can be tied together.

(1) The progress of current public key cryptography

Since the inventions of Public Key Cryptography (PKC) in 1976[4] and 1978[9] respectively, a tremendous amount of public key based cryptographic protocols and algorithms for data encryption, digital signatures, secure e-commerce, electronic payments, etc. have been proposed. The traditional public key cryptogra-

phy makes the key management easier because each user just keeps a single public key for encryptions regardless of the number of people on the network with whom the user wishes to communicate securely. However, because the public keys are to be delivered to all the communicating parties over the open network channels or to be stored in a public repository where anyone can access, the traditional PKC has its own problem that the public keys can be easily attacked and forged.

Therefore, a certificate that contains the value of a public key, key holder, key type, etc., and is digitally signed by a trustworthy party (Certificate Authority, or CA) is used to serve as a proof for the correctness of the public key[7]. But the use of certificates causes other complicated certificate-related management problems including certificate revocation, storage and distribution. As a consequence, the authentication and management of public keys became the main barrier to the progress of the conventional public key cryptography.

Identity (ID)-based cryptography (IBC)[3] solved the certificate management problems efficiently by deriving a public key directly from standard identities such as name, IP address, cell phone number, various E-mail addresses and so on, which eliminates the need to obtain certificates for such identities. Instead, the corresponding private key (or decryption key) should be created by the Key Generation Center (KGC) which is a trusted third party for key certification. Thus, the IBC has the key escrow problem inherently in that the KGC can know every user's decryption key to decrypt any ciphertexts.

In order to remove the certificate problem and the key escrow problem at the same time, combined cryptosystems which utilize the advantages of both PKC and IBC have been suggested. Certificate-based cryptosystem[6] and certificateless public key cryptosystem[1] use public keys created from both identities and user created public values and the associated decryption keys created by user itself and the KGC collaboratively. Since both user-own-selected secret value and the KGC's master secret are required to make the decryption key, the combined schemes not only solve the key escrow problem but also provide implicit certification for the public keys. Researches on easy certification for public keys are still ongoing.

Identity-based public keys give a big advantage over the public key managements including public key creation, distribution and certification. But we argue that using a (universal) single public-private key pair cannot be assumed for applications in the ubiquitous environment.

(2) The need of multiple identities

We are using a number of online and off-line services with different identities required to identify ourselves in our real life. Thus individuals keep many identities such as various e-mail addresses, cell phone number, phone number, Web IDs, a social security number, etc. and use them for their different purposes. For an example, Alice can use her social security number to identify herself for e-government services or her company email address for e-banking services.

Different electronic services may ask for different types of identities. On the other hand, Alice may not want to provide her social security number as her

identity for some untrusted services. Hence, the use of a single universal identity is not appropriate for the pervasive computing. In addition it can cause a serious personal privacy problem that an attacker can collect and trace user information by linking user behaviors with the known single identity and then reveal potentially unintended sensitive information.

Therefore, we need to maintain multiple distinct identities and thus we should be able to create multiple public keys based on those identities for secure communication.

(3) The limitation of current public key cryptography and pseudonym systems
The conventional PKC including IBC cannot be applied straightforward to our multiple public key environment since those systems need to create a unique pair of public key and private key essentially. There is a one-to-one mapping relationship between a public key and its private key. If we want to use multiple public keys in different service groups, then we should keep the exact same number of private keys in a secure manner as well. Strengthening the security, we may increase secret values. However, it makes the key management more difficult losing the benefits of easy key management conventional public key cryptosystems offer. Therefore, a new approach is required to realize our model without increasing secret values.

Recently, pseudonym systems[8][13] have been suggested for protecting personal privacy. The basic goal of these approaches is the same as our scheme in that multiple random pseudonyms are used for user identities and public keys. However, the fundamental difference is that the pseudonym systems are designed for user anonymity. In those pseudonyms, we cannot find out the pseudonyms holders. Hence, they need to use anonymous credential systems[10][14][11] for the proofs of those pseudonyms. On the contrary, we use existing recognizable identities instead of random pseudonyms and certify the correctness of public keys using those identities. Thus, what we aim for is different from that of pseudonym systems.

1.2 Contribution

We introduce the concept of a multiple and unlinkable public key encryption (MU-PKE) scheme which makes it available to operate multiple certified public keys without certificates while keeping a single decryption key. The MU-PKE scheme is one of the PKC and IBC combined public key cryptosystems described previously. We assume a Key Generation Center (KGC) which is a trusted third party such as the KGC described in CL-PKE[1] for implicit key certification. The KGC plays the role of making partial public keys and partial decryption keys of its users.

Functionalities The MU-PKE basically provides the following functionalities.

- **Multiple Public Keys Associated with a Single Decryption Key:** The MU-PKE scheme allows that each user can maintain multiple identity-based public keys which will be published to different service areas where the

base identities have already been commonly used. On the other hand, the user just keeps a single decryption key, regardless of the number of public keys, which can decrypt any ciphertext encrypted in one of the user's public keys. Therefore it still makes the key management easy in spite of keeping various public keys.
At first, each user chooses a random secure private key and a secure master identity in addition to the user's base identities. In order to make a decryption key, the KGC first creates a partial decryption key on the master identity and then the user completes a final decryption key with her private key. Every single public key is also created in a similar way. The KGC first creates a partial public key on a base identity from which the user wants to create the related public key and then the user makes a final public key on the base identity with her master identity and private key. Since the same master identity is used for making both decryption key and public keys, the single decryption key can decrypt any ciphertexts encrypted in public keys created from the common master identity.

- **Certificateless and No Key Escrow:** Each user cannot make her valid public keys or decryption key on her own. User need partial key information which can be generated by the only KGC to complete the final public keys and decryption key. Since the KGC's key certification is integrated into each key there is no need to use certificates. Also, since the user chosen private key is used for generating the decryption key the MU-PKE scheme eliminates the key escrow problem.
- **Unlinkable Public Keys:** Each public key of a single user is differentiated from other public keys by the value of its base identity. The value of the master identity which is commonly used in creating every public key is hidden in the public key. Thus, if an attacker may acquire some pairs of public key and its base identity but cannot recognize the identity holders from those identities, he cannot distinguish if those pairs came from the same user or different users. Hence this property guarantees the personal privacy that is supposed to be protected by using a lot of heterogeneous identities in different service groups or applications.
- **Decryption Key Renewal:** Using a single decryption key can be vulnerable. The exposure of the single decryption key can result in significant damage over the entire services for which the associated public keys are used. So we need to refresh the decryption key periodically. We propose a periodic decryption key update protocol in Section 5.

Our MU-PKE scheme is built from a modified bilinear map such as the Weil pairing on elliptic curves[3]. We define an adaptive chosen-ciphertext security for a MU-PKE scheme and analyze the security of the proposed MU-PKE scheme against the adaptive chosen-ciphertext attack under the random oracle model.

Application As a typical application, our MU-PKE can be applied for email encryption such as PGP[15] under existing e-mail systems. Suppose that a user, Alice is using three emails by different providers such as corporate e-mail, gmail

and hotmail. Alice can create three independent public keys, each of which is created from the individual e-mail address and is used for encryption in the associated e-mail system while keeping a single decryption key. Bob knows Alice's gmail address and wants to send a secured message. He first asks Alice the corresponding public key and then checks the validity of the public key using the gmail address. Finally, Bob encrypts the email message using the public key. Carol, Alice's company colleague, also obtains another public key based on Alice's company email address and then sends an encrypted email message with the public key. Alice can decrypt both emails from Bob and Carol using her single decryption key. Bob and Carol can easily check that those public keys they have received are derived from the email addresses they already know. Therefore, they need not certificates for those public keys. If Alice tries to use an AOL email system newly then she can create a new public key based on the AOL email address at any time if she wants. Alice can still use her decryption key for the new public key.

Organization The rest of the paper is organized as follows. In Section 2, we review the basic notion of a bilinear map and the related cryptographic assumptions which give the basic security of the proposed scheme and then describe the formal definitions of our MU-PKE model and security. The concrete descriptions of the practical MU-PKE schemes are proposed in Section 3, and the unlinkability and security of the proposed schemes against the adaptive chosen ciphertext attack are proven under the random oracle model in Section 4. In Section 5, we suggest a decryption key renewal protocol and then conclude the paper in Section 6.

2 Model and Definition

2.1 System Environments

Before we stress a formal definition of our scheme, we outline the system environment that we assume for the MU-PKE scheme.

Our model consists of **Sender** who makes and sends an encrypted message, **Receiver** who receives and decrypts the encrypted message and a single **Key Generation Center (KGC)**. Once a receiver creates her decryption key and public keys in collaboration with the KGC, then a sender can obtain the receiver's public key and make an encryption for a message with the public key.

By the way, a single receiver can be associated with a lot of different senders. In other word, the receiver's potential senders can be classified into different sender groups according to identities used to identify the receiver for individual sender groups.

Suppose that Alice, a receiver denoted as R_A, possesses t identities used in numerous online applications. Then the potential senders can be logically divided into t sender groups. We denote the total sender groups as $S_A = \{S_1, ..., S_t\}$.

We also denote each base identity Alice uses as her identifier in S_i as $ID_{A,i}$ for $i = 1, ..., t$. We assume that S_i already knows that $ID_{A,i}$ indicates Alice.

For a secure communication between Alice and the senders, Alice can make separate public keys published to those different sender groups for encryption and a single decryption key for decryption. Alice first creates her decryption key DK_A with the KGC. Then, for $i = 1, ..., t$, Alice makes a public key set $PKS_{A,i}$ based on $ID_{A,i}$ for S_i with the KGC at any time Alice needs. S_i can certify $PKS_{A,i}$ with $ID_{A,i}$, and then can make encryptions with both $PKS_{A,i}$ and $ID_{A,i}$. Consequently, Alice can decrypt all ciphertexts with DK_A no matter which public keys of hers are used for making those ciphertexts.

2.2 Cryptographic Assumptions

We describe the background mathematical definitions and assumptions used throughout this paper. Our scheme is built from a modified bilinear map such as Weil pairing on elliptic curves[3]. We review the modified bilinear map briefly and its computational hard problem that provides the basic security for our MU-PKE scheme.

Bilinear Map: Let G_1 be an additive group of prime order q and G_2 be a multiplicative group of the same order. An admissible bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ between these two groups must satisfy the following properties:

1. Bilinear: We say that a map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is bilinear, for all $Q, W, Z \in G_1$, if

   $$\hat{e}(Q, W + Z) = \hat{e}(Q, W)\hat{e}(Q, Z) \text{ and } \hat{e}(Q + W, Z) = \hat{e}(Q, Z)\hat{e}(W, Z).$$

   Consequently, for any $a, b \in Z_q^*$,

   $$\hat{e}(aQ, bW) = \hat{e}(Q, W)^{ab} = \hat{e}(abQ, W).$$

2. Non-degenerate: The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2. Note that since G_1, G_2 are prime order groups, it is implied that if P is a generator of G_1 then $\hat{e}(P, P)$ is a generator of G_2.
3. Computable: There is an efficient algorithm to compute $\hat{e}(Q, W)$ for all $Q, W \in G_1$.

Bilinear Diffie-Hellman (BDH) Problem: Let G_1, G_2 be two groups of prime order q. Let $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be an admissible bilinear map and let P be a generator of G_1. The BDH problem in $(G_1, G_2, \hat{e})$ is as follow: For any $a, b, c \in Z_q^*$, given $\langle P, aP, bP, cP \rangle$, compute $W = \hat{e}(P, P)^{abc} \in G_2$.

BDH Parameter Generator: As described in [3], this randomized algorithm outputs a triple $(G_1, G_2, \hat{e})$ where G_1 and G_2 are of prime order q and $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a pairing. In general, G_1 is a cyclic subgroup of the additive group of points on a supersingular elliptic curve over E/F_p. G_2 is a cyclic subgroup of the multiplicative group associated with a finite extension of $F_{p^2}^*$.

BDH Assumption: Let $\mathcal{IG}$ be a BDH parameter generator. We say that an algorithm $\mathcal{A}$ has advantage $\epsilon(k)$ in solving the BDH problem for $\mathcal{IG}$ if for sufficiently large k:

$$Adv_{\mathcal{IG},\mathcal{A}}(k) = Pr\left[\mathcal{A}(q, G_1, G_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P,P)^{abc} \middle| \begin{array}{c}(q, G_1, G_2, \hat{e}) \leftarrow \mathcal{IG}(1^k), \\ P \leftarrow G_1^*,\ a, b, c \leftarrow Z_q^*\end{array}\right] \geq \epsilon(k)$$

It is said that $\mathcal{IG}$ satisfies the BDH assumption if for any randomized polynomial time algorithm $\mathcal{A}$ and for any polynomial $f \in Z[x]$ we have that $Adv_{\mathcal{IG},\mathcal{A}}(k) < 1/f(k)$ for sufficiently large k. When $\mathcal{IG}$ satisfies the BDH assumption it is said that the BDH problem is hard in groups generated by $\mathcal{IG}$.

2.3 Definitions and Security Model

We define our MU-PKE scheme with formal descriptions. Let k be a security parameter, and let $\mathcal{IG}$ be a BDH parameter generator. We assume that the KGC can make use of a secure communication channel to communicate with its users.

Definition 1. A MU-PKE scheme is specified by five randomized algorithms $(\mathsf{Setup}, \mathsf{Gen}_{\mathsf{DK}}, \mathsf{Gen}_{\mathsf{PK}}, \mathsf{Encryption}, \mathsf{Decryption})$ such that:

1. Setup: The probabilistic system parameter generation algorithm Setup takes as input the security parameter k and $\mathcal{IG}$. It returns the system parameters *params* and a master key s. The system parameters include descriptions of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. Usually, this algorithm is run by the KGC. The system parameters are publicly available, but only the KGC knows the master key.
2. $\mathsf{Gen}_{\mathsf{DK}}$: The deterministic decryption key generation algorithm $\mathsf{Gen}_{\mathsf{DK}}$ consists of three steps which are set-private-key, extract-partial-decryption key and set-decryption key.
   (a) Set-Private-Key: This step takes as inputs *params* and a receiver R_A's chosen identity string and random x_A, and then outputs R_A's private key pair (x_A, P_A) and master identity $MID_A \in \{0,1\}^*$ which includes P_A. This step is performed by individual receivers.
   (b) Extract-Partial-Decryption-Key: This step takes as inputs the receiver'MID_A and the KGC's master key s, and then outputs a partial decryption key PDK_A for the receiver. MID_A and PDK_A are delivered in a secure manner. This step is performed by the KGC.
   (c) Set-Decryption-Key: This step takes as inputs the partial decryption key PDK_A and the private key x_A, and then outputs a final decryption key DK_A. This step is performed by individual receivers. R_A keeps MID_A, x_A and DK_A in a secure manner.
3. $\mathsf{Gen}_{\mathsf{PK}}$: The probabilistic public key generation algorithm $\mathsf{Gen}_{\mathsf{PK}}$ consists of two steps which are extract-partial-public-key and set-public-key.
   (a) Extract-Partial-Public-Key: This step takes as inputs the receiver's single identity $ID_{A,i}$, a proof about the holder of $ID_{A,i}$ and the KGC's master key s, and then outputs a partial public key $PPK_{A,i}$ for the identity

$ID_{A,i}$. This step is performed by the KGC. $PPK_{A,i}$ is delivered to R_A in a secure way.

(b) Set-Public-Key: This step takes as inputs the identity $ID_{A,i}$, the partial public key $PPK_{A,i}$, the decryption key DK_A, the private key x_A and the master identity MID_A, and then outputs a public key set $PKS_{A,i} = \{E1, E2, E3, E4\}$ for $ID_{A,i}$. This step is performed by individual receivers.

4. Encryption: The probabilistic encryption algorithm Encryption takes as inputs the *params*, a message $M \in \mathcal{M}$, a receiver R_A's single identity $ID_{A,i}$ and its public key set $PKS_{A,i}$ which a sender knows. It first checks the validity of $PKS_{A,i}$ with *params* and $ID_{A,i}$, and if $PKS_{A,i}$ is correct then it makes encryption. It returns either a ciphertext $C \in \mathcal{C}$ or the null symbol $\perp$ indicating an encryption failure. This step is performed by individual senders.
5. Decryption: The deterministic decryption algorithm Decryption takes *params*, R_A's decryption key DK_A and the ciphertext $C \in \mathcal{C}$ as input. It returns the message $M \in \mathcal{M}$ or a null symbol $\perp$ indicating a decryption failure. This step is performed by individual receivers. □

Operation of Algorithms: The KGC initializes all system parameters using the *Setup* algorithm. A receiver, Alice, creates her decryption key with the KGC using the Gen_{DK} algorithm. She performs this algorithm just once when she starts using MU-PKE. On the other hand, Alice performs the Gen_{PK} algorithm whenever she needs to create a new public key for an identity which has no corresponding public key. If a public key is set up between Alice and a sender group, the sender can make encryption for a message with the public key using the *Encryption* algorithm, and then Alice can decrypt the ciphertext with her decryption key using the *Decryption* algorithm.

We now define our security model against adaptive chosen-ciphertext attackers (IND-CCA)[3][2][12] which is an acceptable standard security model for public key cryptography. The traditional IND-CCA model allows for an attacker to obtain plaintexts corresponding to any other ciphertexts the attacker chose rather than a particular ciphertext which is challenged by the attacker. This IND-CCA model can be strengthened to an IND-ID-CCA model for identity-based cryptography by allowing that the attacker can also obtain decryption keys associated with identities that the attacker chose but those identities are different from a particular identity which is challenged by the attacker[3]. Since our scheme is a kind of identity-based public key encryption scheme, we borrow the IND-ID-CCA model for our security. However we need to modify and strengthen the IND-ID-CCA model slightly to make it adequate for our scheme. Because a single receiver maintains multiple identities, the attacker should be able to obtain decryption keys associated with any identities the attacker chose other than identities belonging to a particular receiver which is challenged. On the other hand, the attacker might already know some secure information such as partial public keys and partial decryption keys associated with any identities the attacker chose except for the receiver chosen-private key. The system should remain secure under such those attacks. Hence, we define our adaptive

chosen-ciphertext adversary (IND-MUP-CCA) $\mathcal{A}$'s power as follows:

- The adversary can obtain any valid public key sets associated with identities of his choice.
- The adversary can obtain plaintexts of any ciphertexts of his choice (other than the challenging ciphertext C_{ch} encrypted in a particular identity and public key set $\langle ID_{ch,I}, PKS_{ch,I}\rangle$ of the challenger R_{ch} being attacked).
- The adversary can obtain decryption keys associated with any identities of receiver R_i of his choice (other than identities of the challenger R_{ch} being attacked).
- The adversary can obtain any partial public keys associated with identities of his choice which may include a partial public key associated with the particular identity $ID_{ch,I}$ of the challenger R_{ch} being attacked.
- The adversary can obtain any partial decryption keys associated with identities of his choice which may include the partial decryption key of the challenger R_{ch} being attacked.

We specify the above adversary $\mathcal{A}$ into two types of adversaries denoted as $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{A}_{\mathcal{II}}$ according to the permission of an access to the KGC's master secret s. The Type-1 adversary $\mathcal{A}_{\mathcal{I}}$ cannot access s while the Type-2 adversary $\mathcal{A}_{\mathcal{II}}$ that is like a malicious KGC can obtain s and some receivers' mater identities as well. Notice that both of the adversaries do not have an access to the receivers' private keys.

We say that a multiple and unlinkable public key encryption (MU-PKE) scheme is semantically secure against an adaptive chosen ciphertext attack (IND-MUP-CCA) if no polynomially bounded adversary $\mathcal{A}$ (for both $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{A}_{\mathcal{II}}$) has a non-negligible advantages against the Challenger in the following IND-MUP-CCA game:

- **Setup:** The challenger takes a security parameter k and runs the Setup algorithm. It gives $\mathcal{A}$ the resulting system parameters $params$. It keeps the master key s to itself for the Type-1 adversary $\mathcal{A}_{\mathcal{I}}$ but it gives s to the Type-2 adversary $\mathcal{A}_{\mathcal{II}}$. The challenger is supposed to know (or decide) the relationship regarding which identities belong to which receivers.
- **Phase 1:** $\mathcal{A}$ issues queries $q_1, ..., q_m$ where q_i is one of the following queries. For Type-2 adversary $\mathcal{A}_{\mathcal{II}}$, the challenger responds with not only each query answer but also the master identity associated with each identity since the Type-2 adversary $\mathcal{A}_{\mathcal{II}}$ is supposed to know the master identities of receivers.
  - Public Key Set Query on $\langle ID_{i,j}\rangle$: The challenger decides the corresponding receiver R_i to $ID_{i,j}$ and runs algorithm $\mathsf{GEN_{DK}}$ to obtain R_i's decryption key, private key and master identity. Then it runs $\mathsf{GEN_{PK}}$ to generate the public key set $PKS_{i,j}$ corresponding to $ID_{i,j}$. It responds to the adversary with $PKS_{i,j}$.
  - Partial Public Key Query on $\langle ID_{i,j}\rangle$: The challenger decides the corresponding receiver R_i to $ID_{i,j}$ and responds by running algorithm $\mathsf{GEN_{PK}}$ to get the partial public key $PPK_{i,j}$ corresponding to $ID_{i,j}$. It sends $PPK_{i,j}$ to the adversary.

- Partial Decryption Key Query on $\langle ID_{i,j} \rangle$: The challenger decides the corresponding receiver R_i to $ID_{i,j}$ and responds by running algorithm $\mathsf{GEN}_{\mathsf{DK}}$ to get the partial decryption key PDK_i of R_i. It sends PDK_i to the adversary.
- Decryption Key Query on $\langle ID_{i,j} \rangle$: The challenger decides the corresponding receiver R_i to $ID_{i,j}$ and responds by running algorithm $\mathsf{GEN}_{\mathsf{DK}}$ to generate the decryption key DK_i of R_i. It sends DK_i to the adversary.
- Decryption Query on $\langle ID_{i,j}, C_{i,j} \rangle$: The challenger decides the corresponding receiver R_i to $ID_{i,j}$ and responds by running $\mathsf{GEN}_{\mathsf{DK}}$ to generate the decryption key DK_i of R_i. It then runs *Decryption* to decrypt the ciphertext $C_{i,j}$ using the decryption key DK_i. It sends the resulting plaintext to the adversary.

– **Challenge:** Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $M_0, M_1 \in \{\mathcal{M}\}$ and an identity $ID_{ch,I}$ on which it wishes to be challenged. The only constraint is that the corresponding receiver R_{ch} to $ID_{ch,I}$ did not appear in decryption key query in Phase 1. The challenger picks a random bit $b \in \{0, 1\}$ and sets $C_{ch} = Encryption(params, ID_{ch,I}, PKS_{ch,I}, M_b)$. It sends C_{ch} as the challenge to the adversary.

– **Phase 2:** The adversary issues more queries $q_{m+1}, ..., q_n$ where query q_i is one of the following queries. The constraints are that (1) the corresponding receiver R_i of identity $\langle ID_{i,j} \rangle$ queried for requesting decryption key should not be the same as R_{ch}, and that (2) $\langle ID_{i,j}, C_{i,j} \rangle$ queried for requesting decryption should not be the same as $\langle ID_{ch,I}, C_{ch,I} \rangle$. For Type-2 adversary $\mathcal{A}_{\mathcal{II}}$, the challenger responds with not only each query answer but also the master identity associated with each identity.

- Public Key Set Query on $\langle ID_{i,j} \rangle$: The challenger responds as in Phase 1.
- Partial Public Key Query on $\langle ID_{i,j} \rangle$: The challenger responds as in Phase 1.
- Partial Decryption Key Query on $\langle ID_{i,j} \rangle$: The challenger responds as in Phase 1.
- Decryption Key Query on $\langle ID_{i,j} \rangle$ where $R_i \neq R_{ch}$: The challenger responds as in Phase 1.
- Decryption Query on $\langle ID_{i,j}, C_{i,j} \rangle \neq \langle ID_{ch,I}, C_{ch,I} \rangle$: The challenger responds as in Phase 1.

– **Guess:** The adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-MUP-CCA adversary. We define $\mathcal{A}$'s advantage in attacking the scheme as $Adv_{\mathcal{IG},\mathcal{A}}(k) = |Pr[b = b'] - \frac{1}{2}|$.

Definition 2. We define that a MU-PKE scheme is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time IND-MUP-CCA adversary $\mathcal{A}$ the function $Adv_{\mathcal{IG},\mathcal{A}}(k)$ is negligible. We say that the MU-PKE scheme is IND-MUP-CCA secure. □

The proof of the security for our MU-PKE scheme requires to make use of a weaker notion of security known as semantic security against a chosen plaintext attack (IND-ID-CPA)[3][2][12]. This security is similar to IND-ID-CCA except that the adversary cannot make decryption queries. Thus we define a security against the chosen plaintext attack (IND-MUP-CPA) for our MU-PKE scheme using an IND-MUP-CPA game. The game is identical to the IND-MUP-CCA game defined previously except that the adversary cannot make the decryption queries.

We say that a multiple and unlinkable public key encryption scheme is semantically secure against the chosen plaintext attack (IND-MUP-CPA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following IND-MUP-CPA game:

- **Setup:** Identical to **Setup** in the IND-MUP-CCA game.
- **Phase 1:** $\mathcal{A}$ issues queries $q_1, ..., q_m$ where q_i is one of public key set queries, partial public key queries, partial decryption key queries and decryption key queries. For each query the challenger responds to $\mathcal{A}$ as the same way defined in IND-MUP-CCA. For Type-2 adversary $\mathcal{A}_{\mathcal{II}}$, the challenger responds with not only each query answer but also the master identity associated with each identity since the Type-2 adversary $\mathcal{A}_{\mathcal{II}}$ is supposed to know master identities of receivers.
- **Challenge:** Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $M_0, M_1 \in \{\mathcal{M}\}$ and an identity $ID_{ch,I}$ on which it wishes to be challenged. The only constraint is that the corresponding receiver R_{ch} to the challenging identity $ID_{ch,I}$ did not appear in decryption key query in Phase 1. The challenger picks a random bit $b \in \{0, 1\}$ and sets $C_{ch} = Encryption(params, ID_{ch,I}, PKS_{ch,I}, M_b)$. It sends C_{ch} as the challenge to the adversary.
- **Phase 2:** The adversary issues more queries $q_{m+1}, ..., q_n$ where query q_i is one of public key set queries, partial public key queries, partial decryption key queries and decryption key queries. The only constraint is that the receiver R_i of identity ID_i queried for requesting decryption key should not be the same as R_{ch}. The challenger responds to $\mathcal{A}$ as the same way in Phase 1. For Type-2 adversary $\mathcal{A}_{\mathcal{II}}$, the challenger responds with not only each query answer but also the master identity associated with each identity.
- **Guess:** The adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-MUP-CPA adversary. $\mathcal{A}$'s advantage in attacking the scheme is defined as $Adv_{\mathcal{IG},\mathcal{A}}(k) = |Pr[b = b'] - \frac{1}{2}|$.

Definition 3. We define that a MU-PKE scheme is semantically secure against a chosen plaintext attack if for any polynomial time IND-MUP-CPA adversary $\mathcal{A}$ the function $Adv_{\mathcal{IG},\mathcal{A}}(k)$ is negligible. We say that the MU-PKE scheme is IND-MUP-CPA secure. □

Now we define the unlinkability for our MU-PKE scheme. Roughly speaking, the unlinkability means that an adversary who can obtain every public key sets

he wants cannot decide whether two arbitrary public key sets belong to the same receiver or not when we assume that the adversary has no idea about who created those two public key sets. We say that the MU-PKE scheme is unlinkable if no polynomially bounded adversary $\mathcal{A}_{\mathcal{L}}$ takes non-negligible advantage against the Challengers in the following IND-LlNK game:

- **Setup:** There are two challengers denoted as CH_0, CH_1. They take the security parameter k and run Setup algorithm. Then they run $\mathsf{GEN_{DK}}$ to generate their master identities, private keys and decryption keys. They give the system parameter *params* to $\mathcal{A}_{\mathcal{L}}$.
- **Challenge:** $\mathcal{A}_{\mathcal{L}}$ chooses arbitrary two random strings $ID_1, ID_2 \in \{0,1\}^*$. Then $\mathcal{A}_{\mathcal{L}}$ requests the corresponding public key sets to the challengers. The challengers respond to $\mathcal{A}_{\mathcal{L}}$ as follows:
  1. CH_0 selects a random $c_0 \in \{0,1\}$, then CH_1 sets $c_1 = 1 - c_0$. The challenger CH_i with $c_i = 1$ runs $\mathsf{GEN_{PK}}$ on ID_1 and then outputs PKS_1.
  2. CH_0 selects a random $c_0' \in \{0,1\}$ again, then CH_1 sets $c_1' = 1 - c_0'$. The challenger CH_i with $c_i' = 1$ runs $\mathsf{GEN_{PK}}$ on ID_2 and then outputs PKS_2.
  3. They set $c = c_0 \oplus c_0'$.
  4. They respond with PSK_1, PSK_2 to $\mathcal{A}_{\mathcal{L}}$.
- **Guess:**$\mathcal{A}_{\mathcal{L}}$ guesses $c' \in \{0,1\}$. $\mathcal{A}_{\mathcal{L}}$ sets $c' = 0$ if two public key sets are generated from the same challenger. Otherwise, $\mathcal{A}_{\mathcal{L}}$ sets $c' = 1$ (two public key sets are created by two different challengers). If $c = c'$, then $\mathcal{A}_{\mathcal{L}}$ can link at least two public key sets which belong to the same challenger. In this case, $\mathcal{A}_{\mathcal{L}}$ wins.

We refer to such an adversary $\mathcal{A}_{\mathcal{L}}$ as an IND-LINK adversary. We define $\mathcal{A}_{\mathcal{L}}$'s advantage in attacking the scheme as $Adv_{\mathcal{IG},\mathcal{A}_{\mathcal{L}}}(k) = |Pr[c = c'] - \frac{1}{2}|$.

Definition 4. A multiple public keys encryption scheme is unlinkable if for any polynomial time IND-LINK adversary $\mathcal{A}_{\mathcal{L}}$ the function $Adv_{\mathcal{IG},\mathcal{A}_{\mathcal{L}}}(k)$ is negligible. □

3 A Multiple and Unlinkable Public Key Encryption (MU-PKE) Scheme

We propose concrete descriptions to build our MU-PKE using the admissible bilinear map. We describe our MU-PKE scheme in two stages. First, we give a basic multiple and unlinkable public key encryption (BasicMU-PKE) scheme which is IND-MUP-CPA secure against IND-MUP-CPA game under the BDH assumption. And then we suggest a full multiple and unlinkable public keys encryption (FullMU-PKE) scheme which is transformed by Fujisaki-Okamoto transformation so that is IND-MUP-CCA secure against IND-MUP-CCA game. The security proofs will be shown in Section 4.

3.1 BasicMU-PKE

The full descriptions of the five algorithms needed to define BasicMU-PKE are described below. Presenting our protocols, we use the BDH parameter generator $\mathcal{IG}$ which satisfies the BDH assumption. As we assumed, Alice has already generated her different identities used in different sender groups so senders in each group can authenticate Alice with individual identity which is known to the senders.

1. Setup: The KGC sets the system parameters with this algorithm which performs the following steps with inputs $(k, \mathcal{IG})$: The algorithm
   (a) runs $\mathcal{IG}$ on input k to generate two groups G_1, G_2 of a prime order q and a bilinear map $\hat{e} : G_1 \times G_1 \longrightarrow G_2$;
   (b) chooses a generator $P \in G_1$;
   (c) selects a master secret key s uniformly at random from Z_q^* and sets $P_0 = sP \in G_1$; and
   (d) chooses cryptographic hash functions $H_0 : \{0,1\}^* \longrightarrow Z_q^*, H_1 : \{0,1\}^* \longrightarrow G_1^*, H_2 : G_2 \longrightarrow \{0,1\}^n$, where n is the bit-length of plaintexts.

   The system parameters are $params = \langle G_1, G_2, \hat{e}, n, P, P_0, H_0, H_1, H_2 \rangle$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1 \times \{0,1\}^{2n}$. The KGC keeps s in a secure manner but publishes $params$ to public.
2. $\mathsf{Gen}_{\mathsf{DK}}$: This algorithm is just once performed interactively between Alice and the KGC in a secure way when Alice registers her master identity to the KGC. The algorithm performs the following steps:
   (a) Alice creates user information $Info_A \in \{0,1\}^*$;
   (b) Alice runs Set-Private-Key with inputs $(params, Info_A)$. The algorithm chooses a random number $x_A \in Z_q^*$ and sets $P_A = x_A P \in G_1$ and $MID_A = Info_A \parallel P_A \in \{0,1\}^*$. It outputs her master identity MID_A, her hashed-master identity $M_A = H_1(MID_A) \in G_1$ and her private key pair (x_A, P_A).
   (c) The KGC runs Extract-Partial-Decryption-Key with inputs $(params, MID_A, s)$: The algorithm makes and outputs a partial decryption key PDK_A as follows:

   $$M_A = H_1(MID_A) \in G_1, PDK_A = sM_A \in G_1$$

   The KGC keeps MID_A in its secure storage.
   (d) Alice runs Set-Decryption-Key with inputs $(params, PDK_A, x_A)$. The algorithm makes and outputs the final decryption key DK_A as follows:

   $$DK_A = x_A PDK_A = x_A sM_A \in G_1.$$

   Alice keeps her master identity MID_A, private key x_A and decryption key DK_A in a secure manner.

   We suppose that Alice has t identities denoted as $ID_{A,i}$ for $i = 1, ..., t$. Alice can create a public key set on $ID_{A,i}$ by the following algorithm. And she can generate other public keys based on other identities in the same way.

3. $\mathsf{Gen_{PK}}$: This algorithm is performed interactively between Alice and the KGC in a secure way. This algorithm generates a public key set for an identity $ID_{A,i}$ used for S_i. The algorithms performs the following steps:
   (a) Alice creates a proof $PF_{A,i} = x_A H_1(Info_A \parallel ID_{A,i}) \in G_1$ and sends her identity $ID_{A,i} \in \{0,1\}^*$ with her user information $Info_A$ and $PF_{A,i}$ to the KGC.
   (b) The KGC runs Extract-Partial-Public-Key with inputs $(params, ID_{A,i}, Info_A, PF_{A,i}, s)$. The KGC can retrieve P_A corresponding to $Info_A$ and then checks if $\hat{e}(PF_{A,i}, P) = \hat{e}(H_1(Info_A \parallel ID_{A,i}), P_A)$. If the proof is correct, the algorithm computes $Q_{A,i} = H_1(ID_{A,i}) \in G_1$ and then creates a partial public key $PPK_{A,i} = sQ_{A,i} \in G_1$.
   (c) The KGC sends $PPK_{A,i}$ to Alice.
   (d) Alice runs Set-Public-Key with inputs $(params, ID_{A,i}, PPK_{A,i}, MID_A, x_A, DK_A)$. The algorithms performs as follows:
      i. computes $Q_{A,i} = H_1(ID_{A,i}) \in G_1$;
      ii. computes $a_i = H_0(MID_A || ID_{A,i}) \in Z_q^*$;
      iii. sets $E1 = a_i x_A M_A \in G_1$;
      iv. sets $E2 = \frac{1}{a_i} PPK_{A,i} = \frac{1}{a_i} sQ_{A,i} \in G_1$;
      v. sets $E3 = \frac{1}{a_i} Q_{A,i}$;
      vi. computes $QC_{A,i} = H_1(E1||E2||E3||ID_{A,i})$; and
      vii. sets $E4 = \frac{1}{a_i} QC_{A,i}$; and
      viii. sets and outputs $PKS_{A,i} = \langle E1, E2, E3, E4 \rangle$
   (e) Alice publish each public key set $PKS_{A,i}$ to S_i for encryption.
4. Encryption: This algorithm is performed by S_i who wants to encrypt a message M with Alice's identity $ID_{A,i}$ and public key set $PKS_{A,i}$. The algorithms performs the following steps with inputs $(params, ID_{A,i}, PKS_{A,i}, M)$:
   (a) computes $Q = H_1(ID_{A,i})$;
   (b) computes $QC = H_1(E1||E2||E3||ID_{A,i})$;
   (c) checks if $\hat{e}(E4, Q) = \hat{e}(QC, E3)$, otherwise, rejects the public key set;
   (d) checks if $\hat{e}(E2, P) = \hat{e}(E3, P_0)$, otherwise, rejects the public key set;
   (e) sets
   $$\begin{aligned} g &= \hat{e}(E1, E2) \\ &= \hat{e}(a_i x_A M_A, \tfrac{1}{a_i} sQ_{A,i}) \\ &= \hat{e}(a_i x_A M_A, \tfrac{1}{a_i} sQ) \\ &= \hat{e}(x_A s M_A, Q) \end{aligned}$$
   (f) chooses a random $r \in Z_q^*$ and computes a ciphertext:
   $$C = \langle U, V \rangle = \langle rQ, M \oplus H_2(g^r) \rangle$$
   We note that steps (a) to (e)are just carried out once when S_i gets the public key, and do not have to be carried out in every encryptions.
5. Decryption: Alice can decrypt $C = \langle U, V \rangle$ using the decryption key DK_A with this algorithm. The algorithm performs the following steps with inputs $(params, DK_A, C)$:
   (a) computes
   $$V \oplus H_2(\hat{e}(DK_A, U)) = M$$

Correctness: We show that each ciphertext can be decrypted to the message correctly using the decryption key.

$$\begin{aligned} H_2(g^r) &= H_2(\hat{e}(x_A s M_A, Q)^r) \\ &= H_2(\hat{e}(DK_A, rQ)) \\ &= H_2(\hat{e}(DK_A, U)) \end{aligned}$$

3.2 FullMU-PKE

In this section, we suggest the full descriptions of FullMU-PKE with Fujisaki-Okamoto transformation.

1. Setup: Identical to Setup in the previous BasicMU-PKE except that the KGC chooses two more cryptographic hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \longrightarrow Z_q^*, H_4 : \{0,1\}^n \longrightarrow \{0,1\}^n$. The system parameters are $params = \langle G_1, G_2, \hat{e}, n, P, P_0, H_0, H_1, H_2, H_3, H_4 \rangle$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1 \times \{0,1\}^{2n}$.
2. $\mathsf{Gen}_{\mathsf{DK}}$: Identical to $\mathsf{Gen}_{\mathsf{DK}}$ in BasicMU-PKE.
3. $\mathsf{Gen}_{\mathsf{PK}}$: Identical to $\mathsf{Gen}_{\mathsf{PK}}$ in BasicMU-PKE.
4. Encryption: Identical to step (a) to step (e) in BasicMU-PKE. And the algorithm performs the followings:
   (a) chooses a random $\sigma \in \{0,1\}^n$ and sets $r = H_3(\sigma, M) \in Z_q^*$; and
   (b) computes a ciphertext:

$$C = \langle U, V, T \rangle = \langle rQ, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma) \rangle$$

5. Decryption: The algorithm performs the following steps with inputs $(params, DK_A, C)$ where $C = \langle U, V, T \rangle$:
   (a) computes

$$V \oplus H_2(\hat{e}(DK_A, U)) = \sigma'$$

   (b) computes $T \oplus H_4(\sigma') = M'$
   (c) sets $r' = H_3(\sigma', M')$ and tests if $U = r'Q_{A,i}$. if not, rejects the ciphertext; and
   (d) outputs M' as a plaintext.

4 Security Analysis

We show the securities of our proposed schemes along with the security proofs of Boneh and Franklin's IBE schemes which show foundations to prove the security for identity-based cryptography.

4.1 IND-MUP-CPA Security of BasicMU-PKE

First we show that our BasicMU-PKE is IND-MUP-CPA secure assuming that BDHP is hard in groups generated by $\mathcal{IG}$. The following theorem shows that another adversary $\mathcal{B}$ can solve the BDH problem using the IND-MUP-CPA adversary $\mathcal{A}$ if $\mathcal{A}$ has non-negligible advantage in the IND-MUP-CPA game.

Theorem 1. Suppose that two hash functions H_1 and H_2 are random oracles, and that there are two types of IND-MUP-CPA adversary $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{A}_{\mathcal{II}}$ each of which has advantage $\epsilon(k)$ against BasicMU-PKE. Suppose that $\mathcal{A}_{\mathcal{I}}$ makes at most $q_{PK} > 0$ public key set queries, $q_{PPK} > 0$ partial public key queries, $q_{PD} > 0$ partial decryption key queries, $q_{DK} > 0$ decryption key queries and $q_{H_2} > 0$ hash queries to H_2, and that $\mathcal{A}_{\mathcal{II}}$ makes at most $q_{PK} > 0$ public key set queries, $q_{PD} > 0$ partial decryption key queries, $q_{DK} > 0$ decryption key queries and $q_{H_2} > 0$ hash queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem in groups generated by $\mathcal{IG}$ with advantages at least:

$$Adv^{\mathcal{A}_{\mathcal{I}}}_{\mathcal{IG},\mathcal{B}}(k) \geq \frac{2\epsilon(k)}{e(1 + q_{PPK} + q_{PD} + q_{DK})q_{H_2}},$$

$$Adv^{\mathcal{A}_{\mathcal{II}}}_{\mathcal{IG},\mathcal{B}}(k) \geq \frac{2\epsilon(k)}{e(1 + q_{DK})q_{H_2}}$$

$e \approx 2.71$ is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.
□

In order to prove the above theorem we first define BasicMultiPub which is a related multiple public key encryption scheme with t public keys for a single user but those public keys are not based on identities. BasicMultiPub is specified with the following three algorithms: Setup, Encryption and Decryption. Let k be the security parameter given to the setup algorithm, let $\mathcal{IG}$ be a BDH parameter generator.

- Setup: The algorithm performs as follows:
  - runs $\mathcal{IG}$ to generate $\langle G_1, G_2, \hat{e}\rangle$ with the security parameter k
  - chooses a generator $P \in G_1^*$
  - chooses a random secret $M \in G_1^*$
  - chooses secret values $s, x \in Z_q^*$ and then sets $P_0 = sP$ and $D = xsM \in G_1^*$
  - chooses t random $Q_1, Q_2, ..., Q_t \in G_1^*$ and t random secret integers $a_1, a_2, ..., a_t \in Z_q^*$
  - makes t public key sets $PKS_1, PKS_2, ..., PKS_t$ such like $PKS_i = \langle Q_i, a_i xM, \frac{1}{a_i}sQ_i, \frac{1}{a_i}Q_i, \frac{1}{a_i}T_i\rangle$ where $T_i = Q_i + a_i xM + \frac{1}{a_i}sQ_i + \frac{1}{a_i}Q_i$ for $i = 1, .., t$
  - sets $PKS = \{PKS_1, PKS_2, ..., PKS_t\}$
  - chooses a cryptographic hash function $H_2 : G_2 \rightarrow \{0,1\}^n$.

  The public key is $K_{pub} = \langle G_1, G_2, \hat{e}, n, P, P_0, PKS, H_2\rangle$. The decryption key is $D = xsM$, the message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1 \times \{0,1\}^{2n}$.

– Encryption: To encrypt a message $M \in \mathcal{M}$ with a single public key set $PKS_i = \langle E0, E1, E2, E3, E4\rangle = \langle Q_i, a_i xM, \frac{1}{a_i} sQ_i, \frac{1}{a_i} Q_i, \frac{1}{a_i} T_i\rangle$, the algorithm performs the following steps:
  - computes $T = E0 + E1 + E2 + E3 \in G_1^*$
  - checks if $\hat{e}(E4, E0) = \hat{e}(T, E3)$, otherwise, rejects the public key set.
  - checks $\hat{e}(E2, P) = \hat{e}(E3, P_0)$, otherwise, rejects the public key set.
  - computes $g = \hat{e}(E1, E2) = \hat{e}(xM, sQ_i)$
  - chooses a random $r \in Z_q^*$ and set a ciphertext $C = \langle U, V\rangle = \langle rE0, M \oplus H_2(g^r)\rangle$
– Decryption: To decrypt a ciphertext $C = \langle U, V\rangle$ created by the public key, the algorithm performs the following steps:
  - computes $M = V \oplus H_2(\hat{e}(D, U))$

  M is the plaintext.

This completes the description of BasicMultiPub. In order to prove Theorem 1 we first show that an IND-MUP-CPA attack on BasicMU-PKE can be converted to an IND-CPA attack on BasicMultiPub. This proof shows that partial public key queries, partial decryption key queries and decryption key queries do not give advantages to the adversary. Then we show that BasicMultiPub is IND-CPA secure if the BDH assumption holds.

Lemma 1.1. Let H_1 be a random oracle from $\{0,1\}^*$ to G_1^*. Let $\mathcal{A}_\mathcal{I}$ be a Type-1 IND-MUP-CPA adversary that has advantage $\epsilon(k)$ against BasicMU-PKE. Suppose that $\mathcal{A}_\mathcal{I}$ makes at most $q_{PK} > 0$ public key set queries, $q_{PPK} > 0$ partial public key queries, $q_{PD} > 0$ partial decryption key queries and $q_{DK} > 0$ decryption key queries. Then there is a IND-CPA adversary $\mathcal{B}$ that has advantage at least $\frac{\epsilon(k)}{e(1+q_{PPK}+q_{PD}+q_{DK})}$ against BasicMultiPub. Its running time is $O(time(A_I))$. □

Proof. We show how to construct a IND-CPA adversary $\mathcal{B}$ that uses $\mathcal{A}_\mathcal{I}$ to gain the advantage $\epsilon(k)/e(1 + q_{PPK} + q_{PD} + q_{DK})$ against BasicMultiPub. $\mathcal{B}$ makes use of $\mathcal{A}_\mathcal{I}$ by simulating the Challenger in the IND-MUP-CPA game. $\mathcal{B}$ works as follows:

Setup: First, the challenger CH carries out the Setup algorithm of BasicMultiPub and gives K_{pub} to an adversary $\mathcal{B}$ who simulates BasicMU-PKE against an IND-MUP-CPA adversary $\mathcal{A}_\mathcal{I}$. $\mathcal{B}$ supplies $\mathcal{A}_\mathcal{I}$ with $params = \{G_1, G_2, \hat{e}, n, P, P_0, H_1, H_2\}$ where $\mathcal{B}$ picked $params$ from K_{pub}. But H_1 is a random oracle that will be controlled by $\mathcal{B}$. $\mathcal{A}_\mathcal{I}$ may make queries of hash functions H_1 and H_2 at any time during its attack. These hash functions are handled as follows:

H_1-queries: $\mathcal{A}_\mathcal{I}$ may request two types of H_1-queries. Type 1 hash query is on input $\langle ID\rangle \in \{0,1\}^*$ (to compute Q in BasicMU-PKE) and Type 2 hash query is on input $\langle ID, PKS\rangle$ (to compute QC in BasicMU-PKE).

$\mathcal{B}$ first generates a random pool $MP = \{m_1, ..., m_l\}$ with l integers, where $l < [q/t]$, every integer of which is randomly chosen in Z_q^*. The random pool indicates l different users (receivers) who can possess at least t identities. $\mathcal{B}$ maintains a list of tuples $\langle ID_j, Q_j, T_j, c_j, b_j, x_j, m_j, PKS_j\rangle$ called H_1 list. The list is initially empty and when $\mathcal{A}_\mathcal{I}$ queries H_1 on either input $\langle ID_i\rangle$ or $\langle ID_i, PKS_i\rangle$, $\mathcal{B}$ responds as follows:

1. If ID_i already appears on the H_1 list in a tuple $\langle ID_j, Q_j, T_j, c_j, b_j, x_j, m_j, PKS_j\rangle$, then $\mathcal{B}$ responds with $H_1(ID_i) = Q_j \in G_1$ for Type 1 hash query or $H_1(ID_i, PKS_i) = T_j \in G_1$ for Type 2 hash query.
2. Otherwise, $\mathcal{B}$ generate a random coin $c_i \in \{0, 1\}$ so that $Pr[c_i = 0] = \delta$ for some δ that will be determined later.
3. If $c_i = 0$, then $\mathcal{B}$ randomly picks $b_i, x_i, a_i \in Z_q^*$, and a random value $m_i \in MP$. $\mathcal{B}$ computes $Q_i = b_i P, E1 = a_i x_i m_i P, E2 = \frac{1}{a_i} b_i P_0, E3 = \frac{1}{a_i} b_i P, T_i = Q_i + E1 + E2 + E3$ and $E4 = \frac{1}{a_i} T_i$.
4. Else if $c_i = 1$, $\mathcal{B}$ picks a random $m_i \in MP$ and sets $b_i = m_i$ and $x_i = \perp$. $\mathcal{B}$ chooses a random $PKS_I = \langle Q_I, a_I x M, \frac{1}{a_I} s Q_I, \frac{1}{a_I} Q_I, \frac{1}{a_I} T_I\rangle$ in PKS. $\mathcal{B}$ computes $Q_i = b_i Q_I, E1 = b_i a_I x M, E2 = b_i \frac{1}{a_I} s Q_I, E3 = b_i \frac{1}{a_I} Q_I, E4 = b_i \frac{1}{a_I} T_I$ and $T_i = Q_i + E1 + E2 + E3$.
5. $\mathcal{B}$ sets $PKS_i = \langle E1, E2, E3, E4\rangle$ and adds the tuple $\langle ID_i, Q_i, T_i, c_i, b_i, x_i, m_i, PKS_i\rangle$ to the H_1 list.
6. $\mathcal{B}$ responds to $\mathcal{A}_{\mathcal{I}}$ with $H_1(ID_i) = Q_i$ for Type 1 hash query or $H_1(ID_i, PSK_i) = T_i$ for Type 2 hash query.

H_2 **queries:** Any H_2 queries made by $\mathcal{A}_{\mathcal{I}}$ are passed to the challenger CH to answer. H_2 does not have to be assumed as a random oracle controlled by $\mathcal{B}$ at this attack.

Phase 1: After receiving *params* from $\mathcal{B}$, $\mathcal{A}_{\mathcal{I}}$ launches Phase 1 of its attack, by making a series of requests, each of which is either a public key set query, a partial public key query, a partial decryption key query or a decryption key query.

- **Public Key Set Query:** Suppose that the request is on $\langle ID_i\rangle$. $\mathcal{B}$ runs H_1 query on $\langle ID_i\rangle$ to obtain Q_i and its corresponding tuple in the H_1 list. Then $\mathcal{B}$ returns $\langle PKS_i\rangle$ in the tuple as a public key set.
- **Partial Public Key Query:** Suppose that the request is on $\langle ID_i\rangle$. $\mathcal{B}$ runs H_1 query on $\langle ID_i\rangle$ to obtain Q_i and its corresponding tuple in the H_1 list. There are two cases: (1) If $c_i = 1$, then $\mathcal{B}$ aborts. (2) Otherwise, $\mathcal{B}$ replies with $b_i P_0$.
- **Partial Decryption Key Query:** Suppose that the request is on $\langle ID_i\rangle$. $\mathcal{B}$ runs H_1 on $\langle ID_i\rangle$ query to obtain Q_i and its corresponding tuple in the H_1 list. There are two cases: (1) If $c_i = 1$, then $\mathcal{B}$ aborts. (2) Otherwise, $\mathcal{B}$ replies with $m_i P_0$.
- **Decryption Key Extraction:** Suppose that the request is on $\langle ID_i\rangle$. $\mathcal{B}$ runs H_1 on $\langle ID_i\rangle$ query to obtain Q_i and its corresponding tuple in the H_1 list. There are two cases: (1) If $c_i = 1$, then $\mathcal{B}$ aborts. (2) Otherwise, $\mathcal{B}$ replies with $x_i m_i P_0$.

Challenge: Once $\mathcal{A}_{\mathcal{I}}$ decides that Phase 1 is over, $\mathcal{A}_{\mathcal{I}}$ picks ID_{ch} and two messages M_0, M_1 on which it wishes to be challenged.

$\mathcal{B}$ responds as follows:

1. $\mathcal{B}$ runs H_1 query on ID_{ch} to obtain Q_{ch} and its corresponding tuple. Let $\langle ID_{ch}, Q_{ch}, T_{ch}, c, b, x, m, PKS_{ch}\rangle$ be the corresponding tuple in the H_1 list.

2. If $c = 0$, the $\mathcal{B}$ aborts.
3. Else if $c = 1$, then $\mathcal{B}$ gives the pair $\langle M_0, M_1\rangle$ and $Q_I = b^{-1}Q_{ch}$ to the challenger CH as the messages and public key to be challenged. CH responds with the challenge ciphertext $C_{ch} = \langle U', V'\rangle$ such that C_{ch} is the BasicMultiPub encryption of M_g under PKS_I and K_{pub} for a random $g \in \{0, 1\}$. Then $\mathcal{B}$ sets $C^* = \langle b^{-1}U', V'\rangle$ and delivers C^* to $\mathcal{A}_{\mathcal{I}}$. It is easy to see that C^* is the BasicMU-PKE encryption of M_g for the identity ID_{ch} under public key set PKS_{ch}. Since $Q_{ch} = bQ_I$ and $PKS_{ch} = \langle Q_{ch}, ba_IxM, \frac{1}{a_I}sbQ_I, \frac{1}{a_I}bQ_I, \frac{1}{a_I}bT_I\rangle$, ID_{ch}'s decryption key in the BasicMU-PKE scheme is $D_{ch} = bsxM$. So, observe that
$$\hat{e}(b^{-1}U', D_{ch}) = \hat{e}(b^{-1}U', bD) = \hat{e}(U', D)$$
where D is the challenger's decryption key. Thus the BasicMU-PKE decryption of C^* using D_{ch} is the same as the BasicMultiPub decryption of C_{ch} using D.

Phase 2: $\mathcal{B}$ continues to respond to $\mathcal{A}_{\mathcal{I}}$'s requests as in Phase 1.

Guess: $\mathcal{A}_{\mathcal{I}}$ should make guess g' for g. Then $\mathcal{B}$ outputs g' as its guess for g.

Claim: If $\mathcal{B}$ does not abort during the simulation, then $\mathcal{A}_{\mathcal{I}}$'s view is identical to its view in the real attack. Also, if $\mathcal{B}$ does not abort then $|Pr[g = g'] - \frac{1}{2}| \geq \epsilon(k)$. The probability is over the random bits used by $\mathcal{A}_{\mathcal{I}}$, $\mathcal{B}$ and the challenger CH.

Proof of Claim: $\mathcal{B}$'s responses to all H_1 queries are uniformly and independently distributed as in the real attack. All responses to $\mathcal{A}_{\mathcal{I}}$'s requests including decryption key queries are valid. Furthermore, the challenge ciphertext C^* given to $\mathcal{A}_{\mathcal{I}}$ is the BasicMU-PKE encryption of M_g under the current public key set PKS_{ch} for the identity ID_{ch}, where $g \in \{0, 1\}$ is random. Therefore, by the definition of algorithm $\mathcal{A}_{\mathcal{I}}$, we have $g = g'$ with probability at least $\epsilon(k) + \frac{1}{2}$.

Probability: To complete the proof of Lemma 1.1 we should calculate the probability that $\mathcal{B}$ does not abort during the simulation. Since $\mathcal{B}$ always does not abort for the public key set queries, we count partial public key queries, partial decryption key queries and decryption key queries about abortion. Suppose that $\mathcal{A}$ makes at most q_{PK} public key set queries, q_{PPK} partial public key queries, q_{PD} partial decryption key queries and q_{DK} decryption key queries in total.

The probability $\mathcal{B}$ cannot abort in Phase 1 and Phase 2 is $\delta^{q_{PPK}+q_{PD}+q_{DK}}$. And the probability $\mathcal{B}$ cannot abort in the challenge queries is $1 - \delta$. Therefore, the probability $\mathcal{B}$ cannot abort during the entire simulation is $(1-\delta)\delta^{q_{PPK}+q_{PD}+q_{DK}}$. This value can be maximized at $\delta_O = 1 - \frac{1}{q_{PPK}+q_{PD}+q_{DK}+1}$. Applying δ_O to the above probability, the probability that $\mathcal{B}$ does not abort is at least $1/e(1 + q_{PPK} + q_{PD} + q_{DK})$. □

Lemma 1.2. Let H_1 be a random oracle from $\{0, 1\}^*$ to G_1^*. Let $\mathcal{A}_{\mathcal{II}}$ be a Type-2 IND-MUP-CPA adversary that knows the master secret key s and master identities. Suppose that $\mathcal{A}_{\mathcal{II}}$ has advantage $\epsilon(k)$ against BasicMU-PKE and makes at most $q_{PK} > 0$ public key set queries, $q_{PD} > 0$ partial decryption key queries and $q_{DK} > 0$ decryption key queries. Then there is a IND-CPA adversary $\mathcal{B}$ that has advantage at least $\epsilon(k)/e(1 + q_{DK})$ against BasicMultiPub. Its running time is $O(time(A_{II}))$. □

Proof. We show how to construct an IND-CPA adversary $\mathcal{B}$ that uses $\mathcal{A}_{\mathcal{II}}$ to gain the advantage $\epsilon(k)/e(1+q_{DK})$ against BasicMultiPub. The game between $\mathcal{B}$ and $\mathcal{A}_{\mathcal{II}}$ is similar to the game used in the proof of Lemma 1.1. $\mathcal{B}$ makes use of $\mathcal{A}_{\mathcal{II}}$ by simulating the Challenger.

Setup: Identical to Setup in the proof of Lemma 1.1 except that CH gives the secret s and M to $\mathcal{B}$ so that $\mathcal{B}$ supplies $\mathcal{A}_{\mathcal{II}}$ with $params$ and s (CH still keeps x in secret). Since $\mathcal{A}_{\mathcal{II}}$ is supposed to know some master identities, $\mathcal{A}_{\mathcal{II}}$ can make additional H_1 hash query on those master identities at any time during its attack. The H_1-queries in the proof of Lemma 1.1 should be slightly modified to respond with it to $\mathcal{A}_{\mathcal{II}}$.

H_1-queries: Identical to H_1-queries in the proof of Lemma 1.1 for Type 1 and Type 2 hash queries. $\mathcal{A}_{\mathcal{II}}$ may request a Type 3 hash query on input $\langle MID\rangle \in \{0,1\}^*$ (to compute M in BasicMU-PKE). $\mathcal{B}$ maintains an additional T_3H_1 list with tuples of $\langle MID_i, c_i, m_i, M_i\rangle$. The list is initially empty.

1. If MID_i already appears on the T_3H_1 list in a tuple $\langle MID_j, c_j, m_j, M_j\rangle$, then $\mathcal{B}$ responds with $H_1(MID_i) = M_j$ for Type 3 hash query.
2. Otherwise, $\mathcal{B}$ generates a random coin $c_i \in \{0,1\}$ so that $Pr[c_i = 0] = \delta$.
3. $\mathcal{B}$ picks a random $m_i \in MP$.
4. If $c_i = 0$, then $\mathcal{B}$ sets $M_i = m_iP$.
5. Else if $c_i = 1$, $\mathcal{B}$ sets $M_i = m_iM$
6. $\mathcal{B}$ adds the tuple $\langle MID_i, c_i, m_i, M_i\rangle$ to the T_3H_1 list.
7. $\mathcal{B}$ responds to $\mathcal{A}_{\mathcal{II}}$ with $H_1(MID_i) = M_i$.

Phase 1: After receiving $params$ from $\mathcal{B}$, $\mathcal{A}_{\mathcal{II}}$ launches Phase 1 of its attack, by making a series of requests, each of which is either public key set query, partial decryption key query or decryption key query. Notice that $\mathcal{A}_{\mathcal{II}}$ does not need to make partial public key queries because $\mathcal{A}_{\mathcal{II}}$ can compute a partial public key sQ_i for any Q_i.

- **Public Key Set Query:** Identical to Public Key Set Query in Lemma 1.1.
- **Partial Decryption Key Query:** Identical to Public Key Set Query in Lemma 1.1 except that $\mathcal{B}$ replies with sb_iM if $c_i = 1$.
- **Decryption Key Extraction:** Identical to Decryption Key Extraction in Lemma 1.1.

Challenge: Identical to Challenge in Lemma 1.1.

Phase 2: $\mathcal{B}$ continues to respond to $\mathcal{A}_{\mathcal{II}}$'s requests as in Phase 1.

Guess: $\mathcal{A}_{\mathcal{II}}$ should make guess g' for g. Then $\mathcal{B}$ outputs g' as its guess for g.

As mentioned in Lemma 1.1, if $\mathcal{B}$ does not abort during the simulation, then $\mathcal{A}_{\mathcal{II}}$'s view is identical to its view in the real attack.

Probability: To complete the proof of Lemma 1.2 we should calculate the probability that $\mathcal{B}$ does not abort during the simulation. Since $\mathcal{B}$ always does not abort for the public key set queries and the partial decryption key queries, we just count the decryption key queries about abortion. Suppose that $\mathcal{A}$ makes q_{DK} decryption key queries. The probability $\mathcal{B}$ cannot abort in Phase 1 and

Phase 2 is $\delta^{q_{DK}}$. And the probability $\mathcal{B}$ cannot abort in the challenge queries is $1-\delta$. Therefore, the probability $\mathcal{B}$ cannot abort during the entire simulation is $(1-\delta)\delta^{q_{DK}}$. This value can be maximized at $\delta_O = 1-\frac{1}{q_{DK}+1}$. Applying δ_O to the above probability, the probability that $\mathcal{B}$ does not abort is at least $1/e(1+q_{DK})$ as required. □

Next we show that BasicMultiPub is a semantically secure public key system if the BDH assumption holds.

Lemma 1.3 Let H_2 be a random oracle from G_2 to $\{0,1\}^n$. Let $\mathcal{A}$ be an IND-CPA adversary that has advantage $\epsilon(k)$ against BasicMultiPub. Suppose $\mathcal{A}$ makes at most $q_{H_2} > 0$ queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem for $\mathcal{IG}$ with advantage at least $2\epsilon(k)/q_{H_2}$ and a running time $O(time(\mathcal{A}))$. □

Proof. We assume that $\mathcal{B}$ is given as inputs the BDH parameters $\langle G_1, G_2, \hat{e}\rangle$ produced by $\mathcal{IG}$ and a random instance $\langle P, aP, bP, cP\rangle = \langle P, P_1, P_2, P_3\rangle$ of the BDH problem for these parameters where P is a random generator in G_1 and a, b, c are random integers in Z_q^*. Consequently, $D = \hat{e}(P,P)^{abc} \in G_2$ is the solution of this BDH problem. We show how $\mathcal{B}$ interacts with $\mathcal{A}$ to find D. $\mathcal{B}$ works as follows:

Setup: $\mathcal{B}$ creates BasicMultiPub public key $K_{pub} = \langle G_1, G_2, \hat{e}, n, P, P_0, PKS, H_2\rangle$ such that $P_0 = P_1$ and each $PKS_I = \langle E0, E1, E2, E3, E4\rangle = \langle b_I P, a_I P_2, \frac{1}{a_I} b_I P_0, \frac{1}{a_I} b_I P, \frac{1}{a_I} T_I\rangle$ in PKS where a_I and b_I are random integers in Z_q^* for $I = 1, ..., t$. Here H_2 is a random oracle controlled by $\mathcal{B}$ as described below. $\mathcal{B}$ gives K_{pub} to $\mathcal{A}$. Observe that the (unknown) decryption key associated to K_{pub} is $DK = aP_2 = abP$.

$\mathcal{A}$ may make queries of the hash function H_2 at any time. And these hash queries are handled as follows:

H_2-queries: $\mathcal{B}$ maintains a list of tuples $\langle X_i, H_i\rangle$ which we call H_2 list. This list is initially empty. When $\mathcal{A}$ queries H_2 on an input X_i $\mathcal{B}$ responds as follows:

1. If X_i already appears on H_2 list in a tuple $\langle X_j, H_j\rangle$ then $\mathcal{B}$ responds with $H_2(X_i) = H_j$.
2. Otherwise, $\mathcal{B}$ just picks a random string $H_i \in \{0,1\}^n$ and adds the tuple $\langle X_i, H_i\rangle$ to the H_2 list. $\mathcal{B}$ responds with $H_2(X_i) = H_i$.

Challenge: $\mathcal{A}$ chooses a single PKS_I and outputs two messages M_0, M_1 on which it wishes to be challenged. $\mathcal{B}$ picks a random string $R \in \{0,1\}^n$ and defines C to be the ciphertext $C = \langle b_I P_3, R\rangle$. $\mathcal{B}$ gives C as the challenge to $\mathcal{A}$. Observe that the decryption of C is $R \oplus H_2(\hat{e}(DK, b_I P_3)) = R \oplus H_2(D^{b_I})$

Guess: $\mathcal{A}$ outputs its guess $c' \in \{0,1\}$. At this point $\mathcal{B}$ picks a random tuple $\langle X_j, H_j\rangle$ from the H_2 list and then outputs $D' = X_j^{b_I^{-1}}$ as the solution to the given BDH instances.

$\mathcal{B}$ is simulating the Challenger and the random oracle for H_2. And $\mathcal{A}$'s view is identical to its view in the real attack environment. The probability that $\mathcal{B}$ outputs the correct answer D is at least $2\epsilon(k)/q_{H_2}$ as required if $\mathcal{A}$ is supposed to make at most q_{H_2} H_2 queries. The probability analysis is the same as the proof of Lemma 4.3 in [3] so we omit it. □

Proof Of Theorem 1. The theorem follows directly from Lemma 1.1, Lemma 1.2 and Lemma 1.3. Combining all reductions shows that a Type-1 IND-MUP-CPA adversary on BasicMU-PKE with advantage $\epsilon(k)$ gives a BDH problem for $\mathcal{IG}$ with advantage at least $2\epsilon(k)/e(1+q_{PPK}+q_{PD}+q_{DK})q_{H_2}$, and that a Type-2 IND-MUP-CPA adversary on BasicMU-PKE gives a BDH problem with advantage at least $2\epsilon(k)/e(1+q_{DK})q_{H_2}$, as required. □

4.2 IND-MUP-CCA Security of FullMU-PKE

Next we show that our FullMU-PKE is a chosen ciphertext secure against the IND-MUP-CCA adversary under the assumption that BDH is hard in groups generated by $\mathcal{IG}$.

FullMU-PKE results from applying the Fujisaki-Okamoto transformation[5] to BasicMU-PKE to gain the adaptive chosen ciphertext security. Fujisaki-Okamoto showed that their transformation gives the IND-CCA security to the IBE scheme[3]. Their theorem described below shows that there is an IND-CPA adversary $\mathcal{B}$ that takes advantage $\epsilon_1(k)$ against BasicMultiPub if there is an IND-CCA adversary $\mathcal{A}$ with advantage $\epsilon(k)$ against FullMultiPub which is the result of applying the Fujisaki-Okamoto transformation to BasicMultiPub.

Theorem 2 (Fujisaki-Okamoto). Suppose $\mathcal{A}$ is an IND-CCA adversary that achieves advantage $\epsilon(k)$ when attacking FullMultiPub. Suppose $\mathcal{A}$ has running time $t(k)$, makes at most q_D decryption queries, and makes at most q_{H_3}, q_{H_4} queries to the hash functions H_3, H_4 respectively. Then there is an IND-CPA adversary $\mathcal{B}$ against BasicMultiPub with running time $t_1(k)$ and advantage $\epsilon_1(k)$ where

$$\epsilon_1(k) \geq FO_{adv}(\epsilon(k), q_{H_4}, q_{H_3}, q_D) = \frac{1}{2(q_{H_4}+q_{H_3})}[(\epsilon(k)+1)(1-2/q)^{q_D}-1]$$

$$t_1(k) \leq FO_{time}(t(k), q_{H_4}, q_{H_3}) = t(k) + O((q_{H_4}+q_{H_3})\cdot n).$$

Here q is the size of the groups G_1, G_2 and n is the length of σ. □

Using the Fujisaki-okamoto theorem, the following theorem shows that there is an adversary $\mathcal{B}$ that can solve the BDH problem using the IND-MUP-CCA adversary $\mathcal{A}$ if $\mathcal{A}$ has non-negligible advantage in the IND-MUP-CCA game.

Theorem 3. Suppose that four hash functions H_1, H_2, H_3, H_4 are random oracles, and that there are two types of IND-MUP-CCA adversary $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{A}_{\mathcal{II}}$ each of which has advantage $\epsilon(k)$ against FullMU-PKE and each runs in time at most $t(k)$. Suppose that $\mathcal{A}_{\mathcal{I}}$ makes at most $q_{PK} > 0$ public key set queries, $q_{PPK} > 0$ partial public key queries, $q_{PD} > 0$ partial decryption key queries, $q_{DK} > 0$ decryption key queries and $q_D > 0$ decryption queries, and that $\mathcal{A}_{\mathcal{II}}$ makes at most $q_{PK} > 0$ public key set queries, $q_{PD} > 0$ partial decryption key queries, $q_{DK} > 0$ decryption key queries and $q_D > 0$ decryption queries. Also suppose that each adversary makes at most $q_{H_2}, q_{H_3}, q_{H_4}$ queries to the hash functions H_2, H_3, H_4 respectively. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem in groups generated by $\mathcal{IG}$ with advantages at least:

$$Adv^{\mathcal{A}_{\mathcal{I}}}_{\mathcal{IG},\mathcal{B}}(k) \geq 2FO_{adv}(\frac{\epsilon(k)}{e(1+q_{PPK}+q_{PD}+q_{DK}+q_D)}, q_{H_4}, q_{H_3}, q_D)/q_{H_2},$$

$$Adv^{\mathcal{A}_{\mathcal{IT}}}_{\mathcal{IG},\mathcal{B}}(k) \geq 2FO_{adv}(\frac{\epsilon(k)}{e(1+q_{DK}+q_D)}, q_{H_4}, q_{H_3}, q_D)/q_{H_2}$$

The running time of $\mathcal{B}$ is:

$$t_B(k) \leq FO_{time}(t(k), q_{H_4}, q_{H_3})$$

where FO_{adv} and FO_{time} are Fujisaki-Okamoto functions which are defined in Theorem 2. □

In order to prove the above theorem we first show that an IND-MUP-CCA attack on FullMU-PKE can be converted to an IND-CCA attack on FullMultiPub. Then we show that FullMultiPub is IND-CCA secure if the BDH assumptions holds.

The specification of FullMultiPub is almost same as BasicMultiPub except two more hash functions H_3, H_4 are chosen and used in encryption and decryption. FullMultiPub chooses two cryptographic hash functions H_3, H_4 such like $H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow Z_q^*, H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$. Those hash functions are used for making encryption and decryption in FullMultiPub as the same way used in FullMU-PKE.

Lemma 2.1. Let H_1 be a random oracle. Let $\mathcal{A}_{\mathcal{I}}$ be a Type-1 IND-MUP-CCA adversary that has advantage $\epsilon(k)$ against FullMU-PKE. Suppose that $\mathcal{A}_{\mathcal{I}}$ makes at most $q_{PK} > 0$ public key set queries, $q_{PPK} > 0$ partial public key queries, $q_{PD} > 0$ partial decryption key queries, $q_{DK} > 0$ decryption key queries and $q_D > 0$ decryption queries. Then there is an IND-CCA adversary $\mathcal{B}$ that has advantage at least $\frac{\epsilon(k)}{e(1+q_{PPK}+q_{PD}+q_{DK}+q_D)}$ against FullMultiPub. Its running time is $O(time(A_I))$. □

Proof. We construct an IND-CCA adversary $\mathcal{B}$ that uses $\mathcal{A}_{\mathcal{I}}$ to gain the advantage $\frac{\epsilon(k)}{e(1+q_{PPK}+q_{PD}+q_{DK}+q_D)}$ against FullMultiPub. $\mathcal{B}$ makes use of $\mathcal{A}_{\mathcal{I}}$ by simulating the Challenger in the IND-MUP-CCA Game. $\mathcal{B}$ works as follows:

Setup: Identical to Setup in Lemma 1.1 except that H_3, H_4 are included in K_{pub} and $params$.

H_1-queries: Identical to H_1-queries in Lemma 1.1.

Phase 1: Public key set query, partial public key query, partial decryption key query and decryption key query are handled as identical as those in Lemma 1.1.

- **Decryption Query:** Suppose that the request is on $\langle ID_i, C_i \rangle$. Let $C_i = \langle U_i, V_i, W_i \rangle$ be a ciphertext encrypted with a valid public key set PKS_i associated with ID_i. $\mathcal{B}$ responds to this query as follows:
  1. $\mathcal{B}$ runs H_1 query on ID_i to obtain Q_i and its corresponding tuple. Let $\langle ID_i, Q_i, T_i, c_i, b_i, x_i, m_i, PKS_i \rangle$ be the corresponding tuple in the H_1 list.
  2. if $c_i = 0$, then $\mathcal{B}$ runs decryption key query to obtain the decryption key $D_i = x_i m_i P_0$ for the public key $\langle ID_i, PKS_i \rangle$, and uses it to decrypt C_i.
  3. If $c_i = 1$, then $Q_i = b_i Q_I$ and $PKS_i = \langle Q_i, b_i a_I xM, \frac{1}{a_I} s b_i Q_I, \frac{1}{a_I} b_i Q_I, \frac{1}{a_I} b_i T_I \rangle$. And the corresponding decryption key is $D_i = b_i sxM$. $\mathcal{B}$ sets $C_i' = \langle b_i U, V, W \rangle$. $\mathcal{B}$ relays C' to the challenger CH. CH's decryption key is

$D = xsM$, thus $\hat{e}(D, b_i U) = \hat{e}(xsM, b_i U) = \hat{e}(b_i xsM, U) = \hat{e}(D_i, U)$. The FullMU-PKE decryption of C_i using D_i is the same as the Full-MultiPub decryption of C_i' using D. Therefore CH provides the correct decryption of C'. $\mathcal{B}$ relays the challenger's response back to $\mathcal{A}_{\mathcal{I}}$.

Challenge: Once $\mathcal{A}_{\mathcal{I}}$ decides that Phase 1 is over, $\mathcal{A}_{\mathcal{I}}$ picks ID_{ch} and two messages M_0, M_1 on which it wishes to be challenged.

$\mathcal{B}$ responds as follows:

1. $\mathcal{B}$ runs H_1 query on ID_{ch} to obtain Q_{ch} and its corresponding tuple. Let $\langle ID_{ch}, Q_{ch}, T_{ch}, c, b, x, m, PKS_{ch} \rangle$ be the corresponding tuple in the H_1 list.
2. If $c = 0$, the $\mathcal{B}$ aborts.
3. Else if $c = 1$, then $\mathcal{B}$ gives the pair $\langle M_0, M_1 \rangle$ and $Q_I = b^{-1} Q_{ch}$ to the challenger CH as the messages and public key to be challenged. CH responds with the challenge ciphertext $C_{ch} = \langle U', V', W' \rangle$ such that C_{ch} is the Full-MultiPub encryption of M_g under PKS_I and K_{pub} for a random $g \in \{0, 1\}$. Then $\mathcal{B}$ sets $C^* = \langle b^{-1}U', V', W' \rangle$ and delivers C^* to $\mathcal{A}_{\mathcal{I}}$. As described in Lemma 1.1., C^* is the FullMU-PKE encryption of M_g for the identity ID_{ch} under public key set PKS_{ch}.

Phase 2: $\mathcal{B}$ responds to public key set query, partial public key query, partial decryption key query and decryption key query in the same way described in phase 1.

- **Decryption Query:** $\mathcal{B}$ responds to decryption query in the same way described in phase 1. However if the ciphertext relayed to the Challenger is equal to the Challenger's ciphertext C_{ch}, then $\mathcal{B}$ aborts.

Guess: $\mathcal{A}_{\mathcal{I}}$ should make a guess g' for g. Then $\mathcal{B}$ outputs g' as its guess for g.

As mentioned in Lemma 1.1, if $\mathcal{B}$ does not abort during the simulation then $\mathcal{A}_{\mathcal{I}}$'s view is identical to its view in the real attack.

Probability: Now we analyze the probability $\mathcal{B}$ aborts. $\mathcal{B}$ could abort when one of the following events happens.

1. $E1$ is the event that $\mathcal{A}_{\mathcal{I}}$ issues a partial public key query on ID_i in Phase 1 and Phase 2 that causes $\mathcal{B}$ to abort. If $c_i = 1$ on ID_i then $\mathcal{B}$ aborts.
2. $E2$ is the event that $\mathcal{A}_{\mathcal{I}}$ issues a partial decryption key query on ID_i in Phase 1 and Phase 2 that causes $\mathcal{B}$ to abort. If $c_i = 1$ on ID_i then $\mathcal{B}$ aborts.
3. $E3$ is the event that $\mathcal{A}_{\mathcal{I}}$ issues a decryption key query on ID_i in Phase 1 and Phase 2 that causes $\mathcal{B}$ to abort. If $c_i = 1$ on ID_i then $\mathcal{B}$ aborts.
4. $E4$ is the event that $\mathcal{A}_{\mathcal{I}}$ chooses an identity ID_{ch} to be challenged on that causes $\mathcal{B}$ to abort. If $c_{ch} = 0$ on ID_{ch} then $\mathcal{B}$ aborts.
5. $E5$ is the event that $\mathcal{A}_{\mathcal{I}}$ issues a decryption query on $\langle ID_i, C_i \rangle$ in Phase 2 that causes $\mathcal{B}$ to abort. If C_i' relayed to the FullMultiPub challenger is equal to C_{ch}, then $\mathcal{B}$ aborts.

Claim: $Pr[-E1 \wedge -E2 \wedge -E3 \wedge -E4 \wedge -E5] \geq \delta^{q_{PPK}+q_{PD}+q_{DK}+q_D}(1-\delta)$

Proof of Claim. We prove the claim in a very similar way used to calculate the probability the $\mathsf{BasicPub}^{\mathsf{hy}}$ adversary aborts in [3].

We know that

$$\begin{aligned} &Pr[-E1 \wedge -E2 \wedge -E3 \wedge -E4 \wedge -E5] \\ &= Pr[-E3 \wedge -E5| - E1 \wedge -E2 \wedge -E4]Pr[-E1 \wedge -E2 \wedge -E4]. \end{aligned}$$

As we proved in Lemma 1.1, since $Pr[c = 0] = \delta$, we can obtain easily that $Pr[-E1 \wedge -E2 \wedge -E4] \geq \delta^{q_{PPK}+q_{PD}}(1-\delta)$. Thus

$$\begin{aligned} &Pr[-E1 \wedge -E2 \wedge -E3 \wedge -E4 \wedge -E5] \\ &\geq Pr[-E3 \wedge -E5| - E1 \wedge -E2 \wedge -E4]\delta^{q_{PPK}+q_{PD}}(1-\delta). \end{aligned}$$

For simplifying the notation let E_{124} be the event $E_1 \vee E_2 \vee E_4$.

Now we prove that $Pr[-E3 \wedge -E5| - E_{124}] \geq \delta^{q_{DK}+q_D}$ by induction on the maximum number of queries $q_{DK} + q_D$ for both decryption key query and decryption query made by $\mathcal{A}_{\mathcal{I}}$. Let $i = q_{DK} + q_D$ and let $E^{0\ldots i}$ be the event that $E_3 \vee E_5$ happens after $\mathcal{A}_{\mathcal{I}}$ issues at most i queries. Also, let E^i be the event that $E_3 \vee E_5$ happens for the first time when $\mathcal{A}_{\mathcal{I}}$ issues the i-th query. We prove it by induction on i that $Pr[-E^{0\ldots i}| - E_{124}] \geq \delta^i$. For $i = 0$, $Pr[-E^{0\ldots 0}] = 1$ by definition. Now suppose that the claim holds for $i-1$. Then

$$\begin{aligned} Pr[-E^{0\ldots i}| - E_{124}] &= Pr[-E^{0\ldots i}| - E^{0\ldots i-1} \wedge -E_{124}]Pr[-E^{0\ldots i-1} \wedge -E_{124}] \\ &= Pr[-E^i| - E^{0\ldots i-1} \wedge -E_{124}]Pr[-E^{0\ldots i-1} \wedge -E_{124}] \\ &\geq Pr[-E^i| - E^{0\ldots i-1} \wedge -E_{124}]\delta^{i-1} \end{aligned}$$

Thus the induction suffices if we show that $Pr[-E^i| - E^{0\ldots i-1} \wedge -E_{124}] \geq \delta$. Consider the i-th query $\mathcal{A}_{\mathcal{I}}$ can make. The query would be either decryption key query on ID_i or decryption query on $\langle ID_i, C_i \rangle$ $\mathcal{A}_{\mathcal{I}}$ chose. There are three cases to consider for the i-th identity. ID_i can be either a new identity, an existing identity queried before or $ID_i = ID_{ch}$.

Let $\langle ID_i, Q_i, T_i, c_i, b_i, x_i, m_i, PKS_i \rangle$ be the corresponding tuple of the H_1 list to ID_i. We use a fact that if $c_i = 0$ then E_3 does not happen and E_5 as well since the decryption query is not relayed to the $\mathsf{FullMultiPub}$ challenger. Let P_i be the lowest probability that $E_3 \vee E_5$ does not occur for the i-th query when $E^{0\ldots i-1} \vee E_{124}$ did not happen.

- Case 1: ID_i is a new identity. If the i-th query is decryption key query, then E_3 does not happen only if $c_i = 0$. Thus $P_i = \delta$. Else if the i-th query is decryption query, then E_5 does not happen regardless of the value of c_i. If $c_i = 0$, $\mathcal{B}$ decrypts the query with the corresponding decryption key. Else if $c_i = 1$, $\mathcal{B}$ relays the modified ciphertext to the $\mathsf{FullMultiPub}$ challenger. Thus $P_i = 1$
- Case 2: ID_i has been once queried during the past $i-1$ queries (for decryption key and decryption queries), public key set queries, partial public key queries or partial decryption key queries. We consider each of cases.

- ID_i appeared in the past $i-1$ queries: Since $E_3 \vee E_5$ did not happen during the $i-1$ rounds by definition, if ID_i was queried for decryption key then $c_i = 0$ so $P_i = 1$. Else if ID_i was queried for decryption then c_i could be 0 or 1. If the i-th query on ID_i is decryption query, then $P_i = 1$. Else if the i-th query on ID_i is decryption key query, E_3 does not happen only if $c_i = 0$. So $P_i = \delta$.
- ID_i appeared in public key set query: The value of c_i could be 0 or 1. If the i-th query on ID_i is decryption query, then $P_i = 1$. Else if the i-th query on ID_i is decryption key query, E_3 does not happen only if $c_i = 0$. So $P_i = \delta$.
- ID_i appeared in either partial public key query or partial decryption key query: Since $E_1 \vee E_2$ did not happened, $c_i = 0$. Thus $P_i = 1$ for both decryption key query and decryption query.

– Case 3: $ID_i = ID_{ch}$. By the definition of the IND-MUP-CCA game, the decryption key query on the challenger identity is not allowed. So the i-th query would be for decryption on $\langle ID_i, C_i \rangle$ and $c_i = 1$ as well. So $\mathcal{B}$ would relay a modified ciphertext $C_i' = \langle b_i U_i, V_i, T_i \rangle$ to the challenger. By the definition of the game, C_i is different from $C^* = \langle U, V, T \rangle$ which is the challenging ciphertext sent to $\mathcal{A}_{\mathcal{I}}$. Recall that the challenger's ciphertext is $C_{ch} = \langle U', V, T \rangle = \langle b^{-1}U, V, T \rangle$. Since $C_i \neq C^*$, $C_i' \neq C_{ch}$. Thus $P_i = 1$.

Consequently, we have $P_i = Pr[-E^i| - E^{0 \ldots i-1} \wedge -E_{124}] \geq \delta$ whatever the i-th query is. Hence we say that $Pr[-E1 \wedge -E2 \wedge -E3 \wedge -E4 \wedge -E5] \geq \delta^{q_{PPK}+q_{PD}+q_{DK}+q_D}(1-\delta)$ as required.

The rest of probability analysis to get $\frac{\epsilon(k)}{e(1+q_{PPK}+q_{PD}+q_{DK}+q_D)}$ is identical to the descriptions specified in the proof of Lemma 1.1 so we omit it. □

Lemma 2.2. Let H_1 be a random oracle. Let $\mathcal{A}_{\mathcal{II}}$ be a Type-2 IND-MUP-CCA adversary that has advantage $\epsilon(k)$ against FullMU-PKE. Suppose that $\mathcal{A}_{\mathcal{II}}$ makes at most $q_{PK} > 0$ public key set queries, $q_{PD} > 0$ partial decryption key queries, $q_{DK} > 0$ decryption key queries and $q_D > 0$ decryption queries. Then there is an IND-CCA adversary $\mathcal{B}$ that has advantage at least $\frac{\epsilon(k)}{e(1+q_{DK}+q_D)}$ against FullMultiPub. Its running time is $O(time(A_I I))$. □

Proof. The proof of Lemma 2.2 is almost same as the proof of Lemma 2.1 except that $\mathcal{A}_{\mathcal{II}}$ does not make partial public key queries and that $\mathcal{A}_{\mathcal{II}}$ does not abort on partial decryption key queries. So the events $\mathcal{B}$ aborts are bounded to E_3, E_4 and E_5. Thus we show that $Pr[-E_3 \wedge -E_4 \wedge -E_5] = Pr[-E_3 \wedge -E_5| - E_4]Pr[-E_4] \geq \delta^{q_{DK}+q_D}(1-\delta)$. The probability analysis for that is identical to the analysis in Lemma 2.1 so we omit it. □

Proof of Theorem 3. The above two lemmas show that there is an IND-CCA adversary who has advantage $\epsilon_1(k)$ against FullMultiPub if there is an IND-MUP-CCA adversary who has advantage $\epsilon(k)$ against FullMU-PKE.

Combining Theorem 2 and Lemma 1.3 we obtain easily the fact that there is an adversary $\mathcal{B}$ that can solve the BDH problem with advantage at least $2FO_{adv}(\epsilon_1(k), q_{H_4}, q_{H_3}, q_D)/q_{H_2}$ if there is an IND-CCA adversary $\mathcal{A}$ with advantage $\epsilon_1(k)$ against FullMultiPub.

By combining all together, we say that (1) if there is a Type-1 IND-MUP-CCA adversary on FullMU-PKE with advantage $\epsilon(k)$, then there is another algorithm $\mathcal{B}$ that solves the BDH problem with advantage at least $Adv^{\mathcal{A}_{\mathcal{I}}}_{\mathcal{IG},\mathcal{B}}(k) \geq 2FO_{adv}(\frac{\epsilon(k)}{e(1+q_{PPK}+q_{PD}+q_{DK}+q_D)}, q_{H_4}, q_{H_3}, q_D)/q_{H_2}$, and that (2) if there is a Type-2 IND-MUP-CCA adversary on FullMU-PKE with advantage $\epsilon(k)$, then there is another algorithm $\mathcal{B}$ that solves the BDH problem with advantage at least $Adv^{\mathcal{A}_{\mathcal{II}}}_{\mathcal{IG},\mathcal{B}}(k) \geq 2FO_{adv}(\frac{\epsilon(k)}{e(1+q_{DK}+q_D)}, q_{H_4}, q_{H_3}, q_D)/q_{H_2}$, as required. □

4.3 Proof of Unlinkability

Lastly we show the unlinkability of public keys.

Theorem 4. Suppose that the hash function H_1 is a random oracle. Then public keys in the MU-PKE scheme are unlinkable. □

Proof of Theorem 4. In order to prove the above theorem, we define a related (multiple and unlinkable) pubic key set generation algorithm denoted as UnLinkPub. UnLinkPub is specified by Setup and G_{PKS} as described below.

Setup: The algorithm runs $\mathcal{IG}$ to generates BDH parameters $(\hat{e}, G_1, G_2)$ with a given security parameter k. It chooses random integers $s, b_0, b_1, d_1, d_2, q_1, q_2 \in Z_q^*$ and random $P, P_1, T \in G_1$ and then sets $P_0 = sP$. It sets the system parameter $sparams = \langle \hat{e}, G_1, G_2, P, P_0, P_1, T, b_0, b_1, d_1, d_2, q_1, q_2, s\rangle$. It creates two decryption keys $DK_0 = sb_0P_1, DK_1 = sb_1P_1$. It chooses cryptographic hash functions H_2, H_3, H_4 as defined in FullMU-PKE. It publishes a public parameter set $K_{pub} = \langle \hat{e}, G_1, G_2, n, P, P_0, H_2, H_3, H_4\rangle$.

G_{PKS} **on** $\langle ID_j, CH_i, sparams\rangle$**:** This algorithm takes ID_j, a challenger CH_i's information and $sparams$ as inputs and creates a public key set as follows: It

- sets $H_1(ID_j) = Q_j = q_j\bar{b_i}P$ and $H_1(ID_j, PSK_j) = T_j = \bar{b_i}T$ where $\bar{b_i}$ means b_{1-i};
- sets $a_j = d_j\bar{b_i} \in Z_q^*$; and
- creates $PKS_j = \langle a_jb_iP_1, \frac{1}{a_j}sQ_j, \frac{1}{a_j}Q_j, \frac{1}{a_j}T_j\rangle$

Now we show that if an IND-LINK adversary $\mathcal{A}_{\mathcal{L}}$ has an advantage $\epsilon(k)$ in the IND-LINK game, then $\mathcal{A}_{\mathcal{L}}$ gets at least the same advantage against UnLinkPub.

Setup: First, two challengers CH_0, CH_1 carry out the Setup algorithm of UnLinkPub. Suppose that DK_0 is CH_0's decryption key and DK_1 is CH_1's. They share $sparams$. The challengers give $params = \langle G_1, G_2, \hat{e}, n, P, P_0, H_1, H_2, H_3, H_4\rangle$ to $\mathcal{A}_{\mathcal{L}}$. Here H_1 is a random oracle controlled by the challengers.

Challenge: $\mathcal{A}_{\mathcal{L}}$ generates two random identities $ID_1, ID_2 \in \{0,1\}^*$ and then makes a public key set query on each of them to the challenger. The challengers respond to $\mathcal{A}_{\mathcal{L}}$ as follows: The challengers maintain a list of tuples $\langle ID_j, CH_i, Q_j, T_j, d_j, q_j\rangle$ called PKS list as well.

1. CH_0 selects a random coin $c_0 \in \{0,1\}$ and then CH_1's coin is determined as $c_1 = 1 - c_0$.

2. The challenger CH_i with $c_i = 1$ runs G_{PKS} with inputs $\langle ID_1, CH_i, sparams\rangle$ and then outputs PKS_1.
3. CH_i adds a tuple $\langle ID_1, CH_i, Q_1, T_1, d_1, q_1\rangle$ to the PKS list.
4. CH_0 selects a random coin $c'_0 \in \{0,1\}$ again and CH_1's coin is set as $c'_1 = 1 - c'_0$.
5. The challenger CH_i with $c'_i = 1$ runs G_{PKS} with inputs $\langle ID_2, CH_i, sparams\rangle$ and then outputs PKS_2.
6. CH_i adds a tuple $\langle ID_2, CH_i, Q_2, T_2, d_2, q_2\rangle$ to the PKS list.
7. The challengers set $c = c_0 \oplus c'_0$
8. The challengers respond with PKS_1, PKS_2 to $\mathcal{A}_{\mathcal{L}}$.

Notice that $\mathcal{A}_{\mathcal{L}}$ can make H_1 hash query on $\langle ID_j\rangle$ or $\langle ID_j, PKS_j\rangle$ to the challengers. They look up the PKS list and respond with $H_1(ID_j) = Q_j$ and $H_1(ID_j, PKS_j) = T_j$ to $\mathcal{A}_{\mathcal{L}}$.

Guess: $\mathcal{A}_{\mathcal{L}}$ should make a guess c' for c.

Claim: $\mathcal{A}_{\mathcal{L}}$'s view is identical to its view in the real attack. Also $|Pr[c = c'] - \frac{1}{2}| \geq \epsilon(k)$.

Proof of Claim: The challengers choose b_i, d_j, q_j, for $i = \{0,1\}$ and $j = \{1,2\}$, in uniformly and independently random distribution so that $Q_j = H_1(ID_j), T_j = H_1(ID_j, PKS_j)$ and a_j are computed randomly and independently as well. And the public key sets are valid. Therefore, $\mathcal{A}_{\mathcal{L}}$ makes a guess $c' = c$ with a probability at least $\epsilon(k)$. □

Finally, we show that $\mathcal{A}_{\mathcal{L}}$ cannot obtain non-negligible advantage against UnLinkPub.

There are four combinations those two challengers can make two public key sets for ID_1, ID_2 in the above challenge step. Each two public key sets given to $\mathcal{A}_{\mathcal{L}}$ for each case can be shown as follows:

- Case 1 (CH_0, CH_0): CH_0 generated both PKS_1 and PKS_2. $\mathcal{A}_{\mathcal{L}}$ receives $PKS_1 = \langle d_1 b_1 b_0 P_1, \frac{1}{d_1 b_1} s q_1 b_1 P, \frac{1}{d_1 b_1} q_1 b_1 P, \frac{1}{d_1 b_1} b_1 T\rangle$ and $PKS_2 = \langle d_2 b_1 b_0 P_1, \frac{1}{d_2 b_1} s q_2 b_1 P, \frac{1}{d_2 b_1} q_2 b_1 P, \frac{1}{d_2 b_1} b_1 T\rangle$. $\mathcal{A}_{\mathcal{L}}$ computes $g_1 = \hat{e}(b_0 P_1, s q_1 b_1 P)$ and $g_2 = \hat{e}(b_0 P_1, s q_2 b_1 P)$.
- Case 2 (CH_0, CH_1): CH_0 generated PKS_1 and CH_1 generated PKS_2. $\mathcal{A}_{\mathcal{L}}$ receives $PKS_1 = \langle d_1 b_1 b_0 P_1, \frac{1}{d_1 b_1} s q_1 b_1 P, \frac{1}{d_1 b_1} q_1 b_1 P, \frac{1}{d_1 b_1} b_1 T\rangle$ and $PKS_2 = \langle d_2 b_0 b_1 P_1, \frac{1}{d_2 b_0} s q_2 b_0 P, \frac{1}{d_2 b_0} q_2 b_0 P, \frac{1}{d_2 b_0} b_0 T\rangle$. $\mathcal{A}_{\mathcal{L}}$ computes $g_1 = \hat{e}(b_0 P_1, s q_1 b_1 P)$ and $g_2 = \hat{e}(b_1 P_1, s q_2 b_0 P)$.
- Case 3 (CH_1, CH_0): CH_1 generated PKS_1 and CH_0 generated PKS_2. $\mathcal{A}_{\mathcal{L}}$ receives $PKS_1 = \langle d_1 b_0 b_1 P_1, \frac{1}{d_1 b_0} s q_1 b_0 P, \frac{1}{d_1 b_0} q_1 b_0 P, \frac{1}{d_1 b_0} b_0 T\rangle$ and $PKS_2 = \langle d_2 b_1 b_0 P_1, \frac{1}{d_2 b_1} s q_2 b_1 P, \frac{1}{d_2 b_1} q_2 b_1 P, \frac{1}{d_2 b_1} b_1 T\rangle$. $\mathcal{A}_{\mathcal{L}}$ computes $g_1 = \hat{e}(b_1 P_1, s q_1 b_0 P)$ and $g_2 = \hat{e}(b_0 P_1, s q_2 b_1 P)$.
- Case 4 (CH_1, CH_1): CH_1 generated both PKS_1 and PKS_2. $\mathcal{A}_{\mathcal{L}}$ receives $PKS_1 = \langle d_1 b_0 b_1 P_1, \frac{1}{d_1 b_0} s q_1 b_0 P, \frac{1}{d_1 b_0} q_1 b_0 P, \frac{1}{d_1 b_0} b_0 T\rangle$ and $PKS_2 = \langle d_2 b_0 b_1 P_1, \frac{1}{d_2 b_0} s q_2 b_0 P, \frac{1}{d_2 b_0} q_2 b_0 P, \frac{1}{d_2 b_0} b_0 T\rangle$. $\mathcal{A}_{\mathcal{L}}$ computes $g_1 = \hat{e}(b_1 P_1, s q_1 b_0 P)$ and $g_2 = \hat{e}(b_1 P_1, s q_2 b_0 P)$.

We observe that the same public key set pair $\langle PKS_1, PKS_2\rangle$ is always created in every case. In other words, the public information $\mathcal{A}_{\mathcal{L}}$ receives is always

the same regardless of the challengers. Therefore, it is impossible $\mathcal{A}_{\mathcal{L}}$ can distinguish whether those public key sets are generated from the same challengers or not. So, $\mathcal{A}_{\mathcal{L}}$ cannot link two public keys which belong to the same challenger with non-negligible advantage. Consequently, $\mathcal{A}_{\mathcal{L}}$ cannot take non-negligible advantage in IND-LINK game. □

5 Decryption Key Renewal

In this section we suggest an extended MU-PKE (ExMU-PKE) which provides decryption key renewal. Since a single decryption key is used to decrypt ciphertexts encrypted with various public key sets of a single user, if the decryption key is stolen or exposed to an attacker then the whole groups and applications which use the associated public keys are in danger. Thus we need to refresh the decryption key.

We approach this problem by combining our proposed MU-PKE scheme with the certificate-based encryption scheme[6] which solved the certificate revocation problem by adding periodic certificates to a decryption key. Likewise, in the extended ExMU-PKE protocol, we use time intervals to refresh users' decryption keys periodically. Hence, each user maintains two types of decryption keys. One is a long-term decryption key which is identical to DK in FullMU-PKE and the other is a short-term decryption key SDK which is newly defined for the ExMU-PKE scheme and should be updated periodically. SDK_j means a short-term decryption which is only available for decrypting in a time period j.

At the beginning of every single time period the KGC creates its new temporary secret-public key pair and publishes the public key. The KGC creates periodic partial decryption keys of its valid users which are computed by the KGC's periodic secret key so that users can renew their new periodic decryption keys. The modified specifications of ExMU-PKE are described as follows:

1. Setup: Identical to Setup in FullMU-PKE except that it returns t time intervals and creates a new random secret $s_i \in Z_q^*$ at the beginning of every time interval i for $i = \{0, ..., t-1\}$. Therefore the KGC maintains two types of public keys. One is a long-term public key $P_0 = sP$ and the other is a periodic temporary public key $P_i = s_i P$. The KGC publishes the temporary public key periodically.
2. $\mathsf{Gen_{DK}}$: Identical to Gen_{DK} in BasicMU-PKE except that a receiver A keeps another periodic decryption key $SDK_{A,i}$ in secret. At the beginning of time period $i \geq 0$ the KGC runs Extract-Partial-Decryption-Key with inputs $(params, MID_A, s_i)$ and outputs the periodic partial decryption key $PSDK_{A,i} = s_i M_A$. The receiver runs Set-Decryption-Key with inputs $(params, PSDK_{A,i}, x_A)$ and gets $SDK_{A,i} = x_A PSDK_{A,i} = x_A s_i M_A$.
3. $\mathsf{Gen_{PK}}$: Identical to Gen_{PK} in BasicMU-PKE.
4. Encryption: In the time period $i \geq 0$ a sender S_j gets the KGC's periodic public key P_i of the time period i. S_j checks the validity of the receiver's identity $ID_{A,j}$ and public key set $PKS_{A,j} = \langle E1, E2, E3, E4 \rangle$ as the same

way as described in FullMU-PKE. If the public key set is correct, S_j generates g as follows:

$$\begin{aligned} g &= \hat{e}(E1, E2)\hat{e}(E1, P_i) \\ &= \hat{e}(x_A s M_A, Q)\hat{e}(a_j x_A s_i M_A, P) \end{aligned}$$

S_j chooses a random $\sigma \in \{0,1\}^n$ and sets $r = H_3(\sigma, M)$, then computes a ciphertext $C = \langle U, W, V, T\rangle = \langle rQ, rP, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma)\rangle$.

5. Decryption: The algorithm performs the following steps with inputs $(params, DK_A, SDK_i, C)$ where $C = \langle U, W, V, T\rangle$:
   (a) computes $a = H_0(MID_A||ID_{A,j})$ and $V' = aW$
   (b) computes
   $$V \oplus H_2(\hat{e}(DK_A, U)\hat{e}(SDK_{A,i}, V')) = \sigma'$$
   (c) computes $T \oplus H_4(\sigma') = M'$
   (d) sets $r' = H_3(\sigma', M')$ and tests if $W = r'P$. if not, rejects the ciphertext; and
   (e) outputs M' as a plaintext.

Since both the long-term decryption key and the short-term decryption key are used for decrypting a ciphertext, a receiver or an attacker who succeeded to get a long-term decryption key but did not get a valid current periodic partial decryption key from the KGC cannot decrypt a ciphertext correctly. Since the current short-term decryption key is created randomly, an attacker who has got valid past short-term decryption key cannot obtain any idea about the current short-term decryption key and cannot decrypt ciphertexts generated in the current time period as well.

6 Conclusion

We proposed a new multiple and unlinkable public key encryption (MU-PKE) scheme which allows the use of multiple identity-based public keys in different groups or applications while keeping a single decryption key. The proposed MU-PKE takes all advantages of the identity-based encryption scheme and traditional public key encryption scheme. Since each public key is created based on an associated identity that is already recognized by each group or application, it is easy to check if the public key came from the identity. The public keys and associated decryption key are created collaboratively with user and the KGC. It provides certification for public keys so it removed the use of certificates the traditional public key encryption scheme has. In addition, it resolved the key escrow problem the traditional identity-based encryption scheme has inherently because a user own selected private key that the KGC does not know is used to make the decryption key. And we showed that our FullMU-PKE is IND-MUP-CCA secure under the random oracle model.

We minimized the size and numbers of secret values by using a single decryption key regardless of the number of public keys. It provided user convenience in storing and maintaining the secret values. Further more, we suggested an

advanced ExMU-PKE scheme which allows the decryption key to be refreshed periodically in order to mitigate some vulnerabilities which may come from using a single decryption key. It can also cope with the identity revocation problem that would result from the corresponding decryption key collapse by updating the decryption key periodically.

The created public keys are unlinkable. Unless an attacker or malicious user knows information about identities and their holders in advance, he cannot link identities and public keys which belong to the same user from the given identities and public keys. So it provides personal privacy in pervasive computing environment by protecting the attacker from collecting and tracing user information. We also proved the unlinkability of public keys under the random oracle model.

While this paper describes the fundamental concepts of the proposed MU-PKE scheme, following researches are going on to further improve its efficiency especially with regard to key renewal problem. The use of a single decryption key for various public keys makes the key renewal problem relatively more important compared with traditional public key cryptosystems. To this end, we need more efficient decryption key renewal protocols which may not need the KGC's periodic temporary key pair or which may allow revoking a specific public key set among the whole bunch of public key sets of each user. The KGC may want to have each user's public key sets under control so that it wants to create each user's periodic decryption key which can be associated with only current valid public key sets in the time period i. Currently, we are researching the decryption key renewal protocol adequate for the MU-PKE environment.

References

1. S. Al-Riyami and K. Paterson, Certificateless Public Key Cryptography, Advances in Cryptology - ASIACRYPT'03, LNCS 2894, 2003, pp. 452-473.
2. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, Relations among Notions of Security for Public-Key Encryption Schemes, Advances in Cryptology - Crypto'98, LNCS 1462, 1998, pp. 26-45.
3. D. Boneh and M. Flanklin, Identity-Based Encrytion from the Weil pairing, SIAM J. of Computing, vol. 32, no. 3, 2003, pp. 586-615.
4. W. Diffie and M. E. Hellman, New Directions in Cryptography, IEEE Trans. on Information Theory, vol. IT-22, no. 6, 1976, pp. 644-654.
5. E. Fujisaki and T. Okamoto, Secure Integration of Asymmetric and Symmetric Encryption Schemes, Advances in Cryptology - Crypto 99, LNCS 1666, 1999, pp. 537-554.
6. C. Gentry, Certificate-Based Encryption and the Certificate Revocation Problem, Advances in Cryptology - Eurocrypt'03, LNCS 2656, 2003, pp. 272-293.
7. Kohnfelder, Toward a Practical Public Key Cryptosystems, Bachelor's thesis, MIT Department of Electronic Engineering, 1978.
8. A. Lysyanskaya, R. Rivest, A. Sahai and S. Wolf, Pseudonym Systems, Selected Areas in Cryptography, vol. 1758, 1999.
9. R. C. Merkle, Secure Communication Over Insecure Channels, Communications of the ACM, vol. 21, no. 4, 1978, pp. 294-299.

10. P. Persiano and I. Visconti, An Anonymous Credential System and a Privacy-Aware PKI, Proc. of ACISP 03, LNCS 2727, 2003, pp. 27-38.
11. P. Persiano and I. Visconti, An Efficient and Usable Multi-show Non-transferable Anonymous Credential System, Proc. of FC 04, LNCS 3110, 2004, pp. 196-211.
12. C. Rackoff and D. Simon, Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attacks, Advances in Cryptology - Crypto'91, LNCS 576, 1991, pp. 433-444.
13. Y. Tamura and A. Miyaji, Anonymity-enhanced Pseudonym System, Proc. of ACNS 03, LNCS 2846, 2003, pp. 33-47.
14. E. R. Verheul, Self-Blindable Credential Certificates from the Weil Pairing, Advances in Cryptology - Asiacrypt 01, LNCS 2248, 2001, pp. 533-551.
15. P. Zimmermann, The official PGP users guide, MIT Press, Cambridge, Mass., 1995.

This article was processed using the LaTeX macro package with LLNCS style