Is it possible to have CBE from CL-PKE?

Bo Gyeong Kang*[1] and Je Hong Park[2]

[1] Department of Mathematics, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea
snubogus@kaist.ac.kr
[2] National Security Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
jhpark@etri.re.kr

Abstract. Recently, Al-Riyami and Paterson proposed a generic conversion from CL-PKE (Certificateless Public Key Encryption) to CBE (Certificate Based Encryption) and claimed that the derived CBE scheme is secure and even more efficient than the original scheme of Gentry. In this paper, we show that their conversion is wrong due to the flaw of the security proof. It leads the new concrete CBE scheme by Al-Riyami and Paterson to be invalidated. In addition, our result supports the impossibility to relate both notions in any directions.

Keywords: Cryptography; Security analysis; Certificateless Public Key Encryption;

1 Introduction

The notions of Certificate Based Encryption (CBE) and Certificateless Public Key Encryption (CL-PKE) are proposed as alternative approaches to overcome several drawbacks of conventional PKIs and Identity Based Encryption (IBE). CBE proposed by Gentry [4] provides an implicit certification mechanism for a conventional PKI and allows a periodical update of certificate status. As conventional PKIs, each user in CBE generates his own public/private key pair and request a long-lived certificate from the CA. But, CA uses identity based cryptography to generate the long-lived certificate as well as short-lived certificates (i.e., certificate status). A short-lived certificate can be pushed only to the owner of the public/private key pair and acts as a partial decryption key. So CBE provides implicit certification, while it is not subjected to the private key escrow problem inherent in IBE.

On the other hand, CL-PKE is designed to overcome the key escrow limitation of IBE. As IBE, each user has a unique identifier and the (partial) private

* This work was done while the first author was studying in the University of Maryland, USA.

key associated with that identifier is computed by a Key Generation Center (KGC), who knows some special master secret, and distributed to the user with that identifier. However, unlike a traditional IBE scheme, the user also publishes a public key, based on a secret value which the user alone knows. This user secret value is also contained in the user's private key. So the KGC does not know the user's private key that implies the escrow freeness. Note that the user's public key need not to be certified as in conventional PKIs.

Although CBE and CL-PKE were developed independently, both of them were motivated to provide alternative scheme with merits of PKI and IBE at the same time. So a natural question to establish the connection of two concepts arose. After it was briefly recognized in [1], Yum and Lee gave a generic construction for CL-PKE from IBE [6] and explored the relationships between IBE, CBE and CL-PKE [7]. However their construction is lack of consideration about the full security model in [1] by placing a certain extra limitations on the adversaries [2]. Recently, Al-Riyami and Paterson in [2] presented a generic conversion of a secure CBE scheme from a secure CL-PKE scheme, also explained why it is unlikely to be forthcoming in the opposite direction.

In this note, we point out that the claim of Al-Riyami and Paterson about relationship between CBE and CL-PKE is wrong. More precisely, their conversion from CL-PKE to CBE has a critical flaw in the security proof, which finally brings the evidence that each concept has its own unique advantage even with many aspects in common. Recent result of Zhang and Feng which showed the insecurity of the CL-PKE scheme in [2] is independent from our work. What they considered is not the possible connectivity of CBE and CL-PKE, but the security of the concrete CL-PKE scheme itself.

This paper is organized as follows. Section 2 and 3 briefly reviews the definition and security model for CL-PKE and CBE, respectively. Much of the detail such as a concrete scheme will be omitted, stating only what is needed for our discussion. Section 4 points out some problems for the generic construction of PKC 2005 and Section 5 draws conclusions.

2 Certificateless Public Key Encryption

In this section, we briefly review the definition and security model for CL-PKE from [2]. As a note, the CL-PKE scheme in [1] does not fit in this definition.

Definition 1. A certificateless public key encryption scheme has the following components:

- $\mathsf{CL.Setup}$ is a probabilistic algorithm that takes security parameter k as input and returns the system parameters params and $\mathsf{master\text{-}key}$.

- CL.PartialPrivateKey is a deterministic algorithm that takes params, master-key and an identifier for entity A, $\mathsf{ID}_A \in \{0,1\}^*$ as inputs and returns a partial private key D_A.
- CL.SetSecret is a probabilistic algorithm that takes params as input and returns a secret value x_A.
- CL.SetPrivate is a deterministic algorithm that takes params, D_A, and x_A as input and returns a private key S_A.
- CL.SetPublic is a deterministic algorithm that takes params, and x_A as input and returns a public key P_A.
- CL.Enc is a probabilistic algorithm that takes params, a message M, P_A, and ID_A as inputs and returns either a ciphertext C or the null symbol $\perp$ indicating an encryption failure.
- CL.Dec is a deterministic algorithm that takes params, C, and S_A as inputs and returns a message M or a message $\perp$ indicating a decryption failure.

The IND-CCA security model distinguishes two types of adversary: A Type I adversary is meant to represent a normal third party attack and a Type II adversary represents an eavesdropping KGC.

Definition 2. A CL-PKE scheme is semantically secure against chosen-ciphertext attacks (IND-CCA) if no polynomially bounded adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage in the following game:

- **Setup:** Challenger $\mathcal{C}$ takes a security parameter k as input and runs the CL.Setup algorithm. It gives $\mathcal{A}$ the resulting system parameters params. If $\mathcal{A}$ is of Type I, then $\mathcal{C}$ keeps master-key to itself, otherwise, it gives master-key to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ issues a sequence of request described below. These queries may be asked adaptively, but are subject to the rules on adversary behavior defined below.
- **Challenge Phase:** Once $\mathcal{A}$ decides that Phase 1 is over it selects a challenge identifier $\mathsf{ID}_{\mathsf{ch}}$ and corresponding public key P_{ch}, and then outputs $\mathsf{ID}_{\mathsf{ch}}$ and two equal length plaintexts M_0, M_1. $\mathcal{C}$ now picks a random bit $b \in \{0,1\}$ and computes C^*, then encryption of M_b under the current public key P_{ch} for $\mathsf{ID}_{\mathsf{ch}}$. Then C^* is delivered to $\mathcal{A}$.
- **Phase 2:** Now $\mathcal{A}$ issues a second sequence of requests as in Phase 1, again subject to the rules on adversary behavior below.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}$'s advantage in this game to be $\mathrm{Adv}(\mathcal{A}) := 2|\Pr[b = b'] - \frac{1}{2}|$.

In this Game, an IND-CCA adversary $\mathcal{A}$ may carry out several requests and then the challenger $\mathcal{C}$ handles them in Phase 1 or 2 as follows:

1. **Extract Partial Private Key of Entity** A**:** $\mathcal{C}$ responds by running algorithm $\mathsf{CL.PartialPrivateKey}$ to generate D_A for entity A.
2. **Extract Private Key for Entity** A**:** If A's public key has not been replaced then $\mathcal{C}$ can respond by running algorithm $\mathsf{CL.SetPrivate}$ to generate the private key S_A for entity A. It is assumed that the adversary does not make such queries for entities whose public keys have been changed.
3. **Request Public Key of Entity** A**:** $\mathcal{C}$ responds by running algorithm $\mathsf{CL.SetPublic}$ to generate the public key P_A for entity A. If necessary, first runs algorithm $\mathsf{CL.SetSecret}$.
4. **Replace Public Key of Entity** A**:** $\mathcal{A}$ can repeatedly replace the public key P_A for any entity A with any value P'_A of its choice. The current value of an entity's public key is used by $\mathcal{C}$ in any computations or responses to $\mathcal{A}$'s requests.
5. **Decryption Query for Ciphertext** C **and Entity** A**:** If $\mathcal{A}$ has not replaced the public key of entity A, then $\mathcal{C}$ responds by running algorithm $\mathsf{CL.SetPrivate}$ to obtain the private key S_A, then running $\mathsf{CL.Dec}$ on ciphertext C and private key S_A and returning the output to $\mathcal{A}$. It is assumed that $\mathcal{C}$ should properly decrypt ciphertexts, even for those entities whose public keys have been replaced.

CL-PKE Type I IND-CCA Adversary: Adversary $\mathcal{A}_I$ does not have access to $\mathsf{master\text{-}key}$. However, $\mathcal{A}_I$ may request public keys and replace public keys with values of its choice, extract partial private and private keys and make decryption queries, all for identities of its choice. $\mathcal{A}_I$ cannot extract the private key for $\mathsf{ID_{ch}}$ at any point, nor request the private key for any identifier if the corresponding public key has already been replaced. $\mathcal{A}_I$ cannot both replace the public key for the challenge identifier $\mathsf{ID_{ch}}$ before the challenge phase and extract the partial private key for $\mathsf{ID_{ch}}$ in some phase. Furthermore, in Phase 2, $\mathcal{A}_I$ cannot make a decryption query on the challenge ciphertext C^* for the combination $(\mathsf{ID_{ch}}, P_{\mathsf{ch}})$ that was used to encrypt M_b.

CL-PKE Type II IND-CCA Adversary: Adversary A_{II} does have access to $\mathsf{master\text{-}key}$, but may not replace public keys of entities. Adversary $\mathcal{A}_{II}$ can compute partial private keys for itself, given $\mathsf{master\text{-}key}$. It can also request public keys, make private key extraction queries and decryption queries, all for identities of its choice. The restrictions on this type of adversary are that it cannot replace public keys at any point, nor extract the private key for $\mathsf{ID_{ch}}$ at any point. Additionally, in Phase 2, $\mathcal{A}_{II}$ cannot make a decryption query on the challenge ciphertext C^* for the combination $(\mathsf{ID_{ch}}, P_{\mathsf{ch}})$ that was used to encrypt M_b.

3 Certificate Based Encryption

In this section, we briefly review the definition and security model for CBE from [2].

Definition 3. A certificate based encryption scheme is has the following components:

- $\mathsf{CBE.Setup}$ is a probabilistic algorithm that takes a security parameter k as input and returns $\mathsf{sk}_{\mathrm{CA}}$ and public parameters params that include the description of a string space Λ.
- $\mathsf{CBE.SetKeyPair}$ is a probabilistic algorithm that takes params as input and returns a public key pk and a private key sk.
- $\mathsf{CBE.Certify}$ is a deterministic certification algorithm that takes an input $\langle \mathsf{sk}_{\mathrm{CA}}, \mathsf{params}, \tau, \lambda \in \Lambda, \mathsf{pk}\rangle$ and returns Cert'_τ, which is sent to the client. Here τ is a string identifying a time period, while λ contains other information needed to certify the client such as the client's identifying information, and pk is a public key.
- $\mathsf{CBE.Consolidate}$ is a deterministic certificate consolidation algorithm that takes $\langle \mathsf{params}, \tau, \lambda, \mathsf{Cert}'_\tau\rangle$ as input and optionally $\mathsf{Cert}_{\tau-1}$. It returns Cert_τ, the certificate used by a client in time period τ.
- $\mathsf{CBE.Enc}$ is a probabilistic algorithm that takes $\langle \tau, \lambda, \mathsf{params}, \mathsf{pk}, M\rangle$ as input, where M is a message. It returns a ciphertext C for the message M.
- $\mathsf{CBE.Dec}$ is a deterministic algorithm that takes $\langle \mathsf{params}, \mathsf{Cert}_\tau, \mathsf{sk}, C\rangle$ as input in time period τ. It returns either a message M or the special symbol $\perp$ indicating a decryption failure.

In [4, 2], security for CBE is defined using two different games and the adversary chooses which game to play. In Game 1, the adversary models an uncertified entity and in Game 2, the adversary models the certifier in possession of the master-key $\mathsf{sk}_{\mathrm{CA}}$ attacking a fixed entity's public key. Different from the security model in [4], Game 2 requires that params and $\mathsf{sk}_{\mathrm{CA}}$ are fixed at the beginning and are supplied to the adversary.

Definition 4. A CBE scheme is semantically secure against chosen-ciphertext attacks (IND-CCA) if no polynomially bounded adversary $\mathcal{A}$ of Game 1 or 2 has a non-negligible advantage in the following game:

- **Setup:** Challenger $\mathcal{C}$ takes a security parameter k as input and runs the $\mathsf{CBE.Setup}$ algorithm. It gives $\mathcal{A}$ the resulting system parameters params. If $\mathcal{A}$ is of Game 1, then $\mathcal{C}$ keeps $\mathsf{sk}_{\mathrm{CA}}$ to itself, otherwise, it gives $\mathsf{sk}_{\mathrm{CA}}$ and $\langle \mathsf{pk}_{\mathsf{ch}}, \mathsf{sk}_{\mathsf{ch}}\rangle$ obtained by running $\mathsf{CBE.SetKeyPair}$ to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ issues a sequence of requests described below. These queries may be asked adaptively, but are subject of the rules on adversary behavior defined below.

- **Challenge Phase:** Once $\mathcal{A}$ decides that Phase 1 is over it outputs the challenge time period τ_{ch}, certifying information λ_{ch} and two equal length plaintexts M_0, M_1. If $\mathcal{A}$ is of GAME 1, then $\mathcal{A}$ additionally gives $\langle \mathsf{pk}_{\mathsf{ch}}, \mathsf{sk}_{\mathsf{ch}} \rangle$ to $\mathcal{C}$. $\mathcal{C}$ checks that $\lambda_{\mathsf{ch}} \in \Lambda$ and $\langle \mathsf{pk}_{\mathsf{ch}}, \mathsf{sk}_{\mathsf{ch}} \rangle$ is a valid key-pair if it is given by $\mathcal{A}$. If so, $\mathcal{C}$ picks a random bit $b \in \{0, 1\}$ and computes C^*, then encryption of M_b under the current public key $\mathsf{pk}_{\mathsf{ch}}$. Then C^* is delivered to $\mathcal{A}$.
- **Phase 2:** Now $\mathcal{A}$ issues a second sequence of requests as in Phase 1, again subject of the rules on adversary behavior below.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. The adversary wins the game if $b = b'$. we define $\mathcal{A}$'s advantage in this game to be $\mathrm{Adv}(\mathcal{A}) := 2|\Pr[b = b'] - \frac{1}{2}|$.

In this Game, an IND-CCA adversary $\mathcal{A}$ against a CBE scheme may carry out several requests and then the challenger $\mathcal{C}$ handles them as follows:

- **Extract Certificate of Entity** A**:** $\mathcal{C}$ responds by running $\mathsf{CBE.Certify}$ on input $\langle \mathsf{sk}_{\mathrm{CA}}, \mathsf{params}, \tau, \lambda, \mathsf{pk} \rangle$ to generate Cert'_τ for entity A.
- **Decryption Query for Ciphertext** C **and Entity** A**:** $\mathcal{C}$ responds by running $\mathsf{CBE.Certify}$ and $\mathsf{CBE.Consolidate}$ on input $\langle \mathsf{sk}_{\mathrm{CA}}, \mathsf{params}, \tau, \lambda, \mathsf{pk} \rangle$ to obtain Cert_τ, then running $\mathsf{CBE.Dec}$ on ciphertext C, private key sk, and Cert_τ and returning the output to $\mathcal{A}$.

CBE Game 1 IND-CCA Adversary: Adversary $\mathcal{A}_1$ does not have access to $\mathsf{sk}_{\mathrm{CA}}$. However, $\mathcal{A}_1$ may extract certificate and make decryption queries, all for public keys of its choice. So $\mathcal{A}_1$ provide a private key sk along with the corresponding public key pk in all of its queries and $\mathcal{C}$ checks the validity of the key-pair. This enables the challenger to handle decryption queries.

CBE Game 2 IND-CCA Adversary: Adversary $\mathcal{A}_2$ does have access $\mathsf{sk}_{\mathrm{CA}}$, but does not get to choose a challenge public key to attack. Instead, it is given a specific public key from $\mathcal{C}$ at the start of the game. So $\mathcal{A}_2$ can compute Cert'_τ for any public key pk, given $\mathsf{sk}_{\mathrm{CA}}$. In [2], it is restricted to work with the fixed value of params, while $\mathcal{A}_2$ in [4] is allowed to work with multiple values of params. But this restriction is sufficiently reasonable because CA does not change its public parameters frequently. Furthermore, it is required to connect the notions of CBE and CL-PKE [2].

4 CBE from CL-PKE at PKC 2005

At PKC 2005, Al-Riyami and Paterson provided a direct conversion method constructing a CBE scheme Π^{CBE} using the algorithms of a CL-PKE scheme Π^{CL} as components [2]. They claimed that Π^{CBE} is secure in the sense of Definition

4, provided that Π^{CL} is secure in accordance with Definition 2. The main aspect of their method is to define the identifier of the user used in Π^{CL} to include a certain public key of Π^{CBE}, and to obtain short-lived certificates in the Π^{CBE} from the partial private keys in the Π^{CL}.

Let Π^{CL} be a CL-PKE scheme with algorithms ($\mathsf{CL.Setup}$, $\mathsf{CL.PartialPrivateKey}$, $\mathsf{CL.SetSecret}$, $\mathsf{CL.SetPrivate}$, $\mathsf{CL.SetPublic}$, $\mathsf{CL.Enc}$, $\mathsf{CL.Dec}$) as specified in Definition 2. Then a CBE scheme Π^{CBE} is defined as follows:

- $\mathsf{CBE.Setup}$: On input a security parameter k, first run $\mathsf{CL.Setup}(k)$ to obtain $\mathsf{master\text{-}key}$ and $\mathsf{params}^{\mathsf{CL}}$. Then set $\mathsf{sk}_{\mathrm{CA}} = \mathsf{master\text{-}key}$ and Λ be any subset of $\{0,1\}^*$. Define $\mathsf{params}^{\mathsf{CBE}}$ by extending $\mathsf{params}^{\mathsf{CL}}$ to include a description of Λ.
- $\mathsf{CBE.SetKeyPair}$: On input $\mathsf{params}^{\mathsf{CBE}}$ of an entity A, extract $\mathsf{params}^{\mathsf{CL}}$ from it then run $\mathsf{CL.SetSecret}(\mathsf{params}^{\mathsf{CL}}) = x_A$ and $\mathsf{CL.SetPublic}(\mathsf{params}^{\mathsf{CL}}, x_A) = P_A$. The output is $\langle \mathsf{pk}_A, \mathsf{sk}_A \rangle = \langle P_A, x_A \rangle$.
- $\mathsf{CBE.Certify}$: On input $\langle \mathsf{sk}_{\mathrm{CA}}, \mathsf{params}^{\mathsf{CBE}}, \tau, \lambda, \mathsf{pk}_A \rangle$, extract $\mathsf{params}^{\mathsf{CL}}$ from $\mathsf{params}^{\mathsf{CBE}}$. Then set the identifier of A as $\mathsf{ID}_A = \mathsf{params}^{\mathsf{CBE}} \| \tau \| \lambda \| \mathsf{pk}_A$ and run $\mathsf{CL.PartialPrivateKey}(\mathsf{params}^{\mathsf{CL}}, \mathsf{sk}_{\mathrm{CA}}, \mathsf{ID}_A) = D_A$. The output is $\mathsf{Cert}'_\tau = D_A$.
- $\mathsf{CBE.Consolidate}$: On input $\langle \mathsf{params}^{\mathsf{CBE}}, \tau, \lambda, \mathsf{Cert}'_\tau \rangle$, output $\mathsf{Cert}_\tau = \mathsf{Cert}'_\tau$.
- $\mathsf{CBE.Enc}$: On input $\langle \tau, \lambda, \mathsf{params}^{\mathsf{CBE}}, \mathsf{pk}_A, M \rangle$, extract $\mathsf{params}^{\mathsf{CL}}$ from $\mathsf{params}^{\mathsf{CBE}}$ and set the user identifier of A as $\mathsf{ID}_A = \mathsf{params}^{\mathsf{CBE}} \| \tau \| \lambda \| \mathsf{pk}_A$. The output is $C = \mathsf{CL.Enc}(\mathsf{params}^{\mathsf{CL}}, M, \mathsf{pk}_A, \mathsf{ID}_A)$.
- $\mathsf{CBE.Dec}$: On input $\langle \mathsf{params}^{\mathsf{CBE}}, \mathsf{Cert}_\tau, \mathsf{sk}_A, C \rangle$ in time period τ, extract $\mathsf{params}^{\mathsf{CL}}$ from $\mathsf{params}^{\mathsf{CBE}}$ and run $\mathsf{CL.SetPrivate}(\mathsf{params}^{\mathsf{CL}}, D_A, x_A) = S_A$, where $D_A = \mathsf{Cert}_\tau$ and $x_A = \mathsf{sk}_A$. The output is $\mathsf{CL.Dec}(\mathsf{params}^{\mathsf{CL}}, C, S_A)$.

We would like to indicate that Π^{CL} applied for this conversion can be seen as a member of the special class of CL-PKE schemes in which identifiers include public keys. Given a generic CL-PKE scheme, the binding technique in [1] allows it to be a member of that class. This technique is that a user A first fixes its secret value x_A and its public key P_A, then the PKG extracts the partial private key using the binding value $\mathsf{ID}_A \| P_A$ instead of ID_A itself. Implicitly, it forces the user A to use a single public key, because A can only compute one private key from the partial private key. Of course, other users will use $\mathsf{ID}'_A = \mathsf{ID}_A \| P_A$ as the identifier of A to generates a ciphertext for A. So the trust level of the PKG of Π^{CL} gets close to that of a traditional PKI where the CBE scheme can be defined.

The security proof of Π^{CBE} provided in [2] is only restricted to Game 1. For the security proof of the Game 2, the authors in [2] only claimed that similar ideas of the proof for Game 1 may be applied. However we provide a serious observation so that it does not seem to be clear the security of their conversion can be proven also in Game 2. First, we briefly introduce the idea used in the proof of Game 1. Let $\mathcal{A}_1$ be a Game 1 IND-CCA adversary against Π^{CBE} with advantage ϵ. Using $\mathcal{A}_1$ as a black-box, a Type I IND-CCA adversary $\mathcal{B}_I$ against Π^{CL} can be constructed as follows:

$\mathcal{B}_I$ simulates $\mathsf{CBE.Setup}$ of Π^{CBE} by setting Λ to be an arbitrary subset of $\{0,1\}^*$ and $\mathsf{params}^{\mathsf{CBE}}$ to be an extension of $\mathsf{params}^{\mathsf{CL}}$ which includes a description of Λ. $\mathcal{B}_I$ then gives $\mathsf{params}^{\mathsf{CBE}}$ to $\mathcal{A}_1$. To handle a sequence of queries issued by $\mathcal{A}_1$, $\mathcal{B}_I$ sets an identifier $\mathsf{ID}_A = \mathsf{params}^{\mathsf{CBE}}\|\tau\|\lambda\|\mathsf{pk}$ of the entity A using the information τ, λ and pk included in the query of $\mathcal{A}_1$ and replaces the public key with the value pk. In Game 1, it is always possible because $\mathcal{B}_I$ is permitted to replace public keys with values of its choice. Then $\mathcal{B}_I$ makes some queries to $\mathcal{C}$ for the identifier ID_A and relays $\mathcal{C}$'s response to $\mathcal{A}_1$.

But there are some problems in using this approach for the Game 1 adversary in the Game 2 adversary directly. Let $\mathcal{A}_2$ be a Game 2 IND-CCA adversary against Π^{CBE} with advantage ϵ. At the start of the game, the first improper situation occurs. To construct a Type II IND-CCA adversary $\mathcal{B}_{II}$ against Π^{CL} using $\mathcal{A}_2$, $\mathcal{B}_{II}$ simulates $\mathsf{CBE.Setup}$ of Π^{CBE} as $\mathcal{B}_I$ and then gives $\mathsf{params}^{\mathsf{CBE}}$ and $\mathsf{sk}_{\mathrm{CA}}$ to $\mathcal{A}_2$. In addition, $\mathcal{A}_2$ should be supplied with a challenge public key $\mathsf{pk}_{\mathsf{ch}}$ at the beginning of the game. To perform this, $\mathcal{B}_{II}$ should request a public key for any ID and returns it to $\mathcal{A}_2$ as the challenge public key $\mathsf{pk}_{\mathsf{ch}}$, then sets $\mathsf{ID}'_A = \mathsf{params}^{\mathsf{CBE}}\|\tau\|\lambda\|\mathsf{pk}_{\mathsf{ch}}$. But the identifier ID chosen by $\mathcal{B}_{II}$ is supposed to already include a corresponding public key. So it seems unnatural that $\mathcal{B}_{II}$ requests the public key for ID to its oracle. In fact, throughout Game 1 and Game 2 as long as the identifier contains the corresponding public key, queries of requesting public key can not be appropriately handled. Fortunately, in Game 1, $\mathcal{B}_I$ does not need to ask public key queries for identifiers to respond to $\mathcal{A}_1$ against Π^{CBE} (See previous paragraph.). It however, causes a problem in Game 2 because $\mathcal{B}_{II}$ should provide a challenge public key to $\mathcal{A}_2$ in the way of requesting public key to its own oracle. The second problem is due to the lack of consideration of the simple fact that $\mathcal{B}_{II}$ is not allowed to replace a public key for ID'_A with $\mathsf{pk}_{\mathsf{ch}}$. One might be concerned that another public key different from $\mathsf{pk}_{\mathsf{ch}}$ included in ID'_A will be set as the public key for ID'_A. Of course, it possibly happens in both Games. While there is a solution for the adversary in Game 1 to fix this problem because public key replacement queries of $\mathcal{B}_I$ are allowed, there is no way to match the public key contained in the identifier to the public key corresponding to the identifier in Game 2. This is really odd situation. So the public key for ID'_A cannot be set as desired in any ways. Even $\mathcal{B}_{II}$ makes decryption queries on input $\langle C, \mathsf{ID}'_A\rangle$ to $\mathcal{C}$, the response can never be expected to be the correct answer for $\langle \tau, \lambda, \mathsf{pk}_{\mathsf{ch}}, \mathsf{sk}_{\mathsf{ch}}, C\rangle$ requested by $\mathcal{A}_2$. This problem might occur even without the restriction for identifiers of CL-PKE scheme. Based on the mentioned problems in Game 2, we conclude that a secure CBE scheme is not possibly obtained from a secure CL-PKE, at least using the conversion in [2]. This is somehow surprising result compared to how much it seems likely to be related.

5 Conclusion

We show that the generic construction of a CBE scheme from a secure CL-PKE scheme proposed by Al-Riyami and Paterson does not satisfy the security model for CBE. Based on the observation for the opposite conversion by [2], we clarify that it still remains an open problem to relate these two concepts.

References

1. S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. *Advances in Cryptology - ASIACRYPT 2003*, Lecture Notes in Comput. Sci., vol. **2894**, pp. 452–473, 2003.
2. S.S. Al-Riyami and K.G. Paterson. CBE from CL-PKE: A generic construction and efficient schemes. *Public Key Cryptography - PKC 2005*, Lecture Notes in Comput. Sci., vol. **3386**, pp. 398–415, 2005.
3. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, vol. **32**(3): 586–615 (2003).
4. C. Gentry. Certificate-based encryption and the certificate revocation problem. *Advances in Cryptology - EUROCRYPT 2003*, Lecture Notes in Comput. Sci., vol. **2656**, pp. 272–293, 2003.
5. B.G. Kang, J.H. Park and S.G. Hahn. A certificate-based signature scheme. *Topics in Cryptology - CT-RSA 2004*, Lecture Notes in Comput. Sci., vol. **2964**, pp. 99–111, 2004.
6. D.H. Yum and P.J. Lee. Generic construction of certificateless encryption. *Computational Science and Its Applications - ICCSA 2004*, Lecture Notes in Comput. Sci., vol. **3043**, pp. 802–811, 2004.
7. D.H. Yum and P.J. Lee. Identity-based cryptography in public key management. *Public Key Infrastructure - EuroPKI 2004*, Lecture Notes in Comput. Sci., vol. **3093**, pp. 71–84, 2004.
8. Z. Zhang and D. Feng. On the security of a certificateless public-key encryption. *Cryptology ePrint Archive*, Report 2005/426.