3-2011

Certificateless public key encryption: A new generic construction and two pairing-free schemes

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Chik How TAN

Theoretical Computer Science 412 (2011) 662–674

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Certificateless public key encryption: A new generic construction and two pairing-free schemes

Guomin Yang *, Chik How Tan

Temasek Laboratories, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Received 4 June 2010
Accepted 19 October 2010
Communicated by X. Deng

Keywords:
Certificateless cryptography
Public key encryption

ABSTRACT

The certificateless encryption (CLE) scheme proposed by Baek, Safavi-Naini and Susilo is computation-friendly since it does not require any pairing operation. Unfortunately, an error was later discovered in their security proof and so far the provable security of the scheme remains unknown. Recently, Fiore, Gennaro and Smart showed a generic way (referred to as the FGS transformation) to transform identity-based key agreement protocols to certificateless key encapsulation mechanisms (CL-KEMs). As a typical example, they showed that the pairing-free CL-KEM underlying Baek et al.'s CLE can be "generated" by applying their transformation to the Fiore–Gennaro (FG) identity-based key agreement (IB-KA) protocol.

In this paper, we show that directly applying the Fiore–Gennaro–Smart (FGS) transformation to the original FG IB-KA protocol in fact results in an insecure CL-KEM scheme against strong adversaries, we also give a way to fix the problem without adding any computational cost. The reason behind our attack is that the FGS transformation requires the underlying IB-KA protocol to be secure in a model that is stronger than the conventional security models where existing IB-KA protocols are proved secure, and the FG IB-KA protocol is in fact insecure in the new model. This motivates us to construct a new generic transformation from IB-KA protocols to CLE schemes. In the paper we present such a transformation which only requires the underlying IB-KA protocol to be secure in a security model that is weaker than the existing security models for IB-KA protocols. We illustrate our transformation by generating a new pairing-free CLE scheme that is obtained by directly applying our transformation to the original FG IB-KA protocol.

1. Introduction

Certificateless Cryptography, introduced by Al-Riyami and Paterson [2], aims to avoid the drawbacks of both traditional public key cryptography which requires a public key infrastructure (PKI), and identity-based cryptography [17] which has the inherent key escrow problem. In this paper, we focus on certificateless encryption (CLE) schemes.

In a certificateless encryption scheme, in order to encrypt a message, a sender needs both the receiver's identity and a public key produced by the receiver, and to decrypt a ciphertext, the receiver needs both a partial private key corresponding to his/her identity, which is generated by a Key Generation Center (KGC), and the secret value corresponding to the public key produced by him/herself.

Two types of adversaries are considered for certificateless encryption [1,2]. A Type-I adversary (or A_I) captures attacks lunched by outsiders. Since in certificateless cryptography, there is no authentication information (such as a X.509 certificate in PKI) for the user public keys, so an adversary can replace them with public keys of its choice. In contrast, A Type-II

* Corresponding author. Tel.: +65 65161147.
E-mail addresses: tslyg@nus.edu.sg (G. Yang), tsltch@nus.edu.sg (C.H. Tan).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.10.025

adversary (or A_{II}) is an honest-but-curious KGC who controls the generation of user partial private keys and wants to break user confidentiality.

Due to the nature of certificateless cryptography, a lot of certificateless encryption constructions make use of identity-based encryption (IBE) as a building block (e.g., [16,9,13,14,8,6]), which result in pairing based schemes. In [4], Baek et al. proposed the first certificateless encryption scheme without pairing in the random oracle model. Unfortunately, an error was later discovered in their security proof. Though no concrete attack so far has been discovered against the scheme, a formal proof of security remains unknown [3].

Recently, Fiore et al. [10] showed a generic way to transform identity-based key agreement (IB-KA) protocols to certificateless key encapsulation mechanisms (CL-KEMs). Interestingly, they showed that the CL-KEM underlying Baek et al.'s certificateless encryption can be "generated" by applying their transform to the Fiore–Gennaro (FG) identity-based key agreement protocol [11]. And from the proven security of the FG IB-KA protocol, a security proof for the corresponding pairing-free CL-KEM can be obtained "for free".

Our contributions. In this paper, we revisit the CL-KEM scheme in [10] and show that it is insecure against strong Type-I or Type-II adversaries. We analyze the reason behind the insecurity of this pairing-free CL-KEM scheme, and present a new variant of it which is provably secure against strong Type-I and Type-II adversaries.

We then present a new generic transformation from IB-KA protocols to certificateless encryption schemes in the random oracle model. The new transformation is more convenient to use than the FGS transformation since the latter requires the underlying IB-KA protocol to be secure in a model which is stronger than the conventional security models where existing IB-KA protocols are proved secure, so before applying the transformation, a new proof for the IB-KA protocol in the stronger model is required, while the new transformation can be directly applied to existing proven secure IB-KA protocols without re-proving them. We also illustrate our transformation by presenting a new pairing-free CLE scheme from the original FG IB-KA protocol.

Paper organization. In the next section, we first introduce some notations and assumptions used throughout the whole paper. In Section 3, we review the definitions and security models for certificateless key encapsulation mechanisms and identity-based key agreement protocols. Then in Section 4, we review the Fiore–Gennaro–Smart transformation and a pairing-free CL-KEM scheme from the Fiore–Gennaro IB-KA protocol. We show the insecurity of the FG CL-KEM and the underlying IB-KA protocol in Section 5, and present a way to fix the problem in Section 6. We then present our new generic construction of CLE schemes in Section 7. We compare all the existing pairing-free CL-KEM/CLE schemes in Section 8 and conclude the paper in Section 9.

2. Preliminaries

2.1. Notations

We denote by $a_1 \| \cdots \| a_n$ the concatenation of $a_1, \ldots, a_n$. If A is a randomized algorithm then $y \leftarrow A(x_1, \ldots; r)$ denotes the operation of running A with random coins r on inputs $x_1, \ldots$ and letting y denote the output. If r is not directly specified, then $A(x_1, \ldots)$ denotes the operation of running A with fresh random coins. If S is a (finite) set, then $s \leftarrow S$ denotes the operation of picking s uniformly at random from S.

2.2. Computational problems

Let $\mathbb{G}$ denote a cyclic group of prime order q, and g a generator of $\mathbb{G}$. Below we review some computational problems that will be used in the rest of the paper.

Computational Diffie–Hellman (CDH) Problem: given $g, g^a, g^b \in \mathbb{G}^3$ as input where a, b are randomly selected from $\mathbb{Z}_q$, compute g^{ab}.

Decisional Diffie–Hellman (DDH) Problem: For randomly selected $a, b, c \in \mathbb{Z}_q^3$, distinguish the tuple (g^a, g^b, g^{ab}) from the tuple (g^a, g^b, g^c).

Strong Diffie–Hellman (SDH) Problem: solve the CDH problem when having access to an oracle $\mathrm{DH}(g^a, \cdot, \cdot)$ such that on input $g^{b'}, g^{c'}, \mathrm{DH}(g^a, g^{b'}, g^{c'})$ returns 1 if and only if $c' = ab'$.

Gap Diffie–Hellman (Gap-DH) Problem: solve the CDH problem when having access to an oracle $\mathrm{DH}(\cdot, \cdot, \cdot)$ such that on input $g^{a'}, g^{b'}, g^{c'}, \mathrm{DH}(g^{a'}, g^{b'}, g^{c'})$ returns 1 if and only if $c' = a'b'$.

3. Security models and definitions

3.1. Certificateless public key encryption

In this section we define Certificateless Encryption (CLE) schemes and its security against Type-I and Type-II adversaries.

Definition 1. A certificateless encryption scheme $\mathcal{CLE}$ consists of the following algorithms.

- $\mathsf{Setup}(1^k)$ takes 1^k as input where k is the security parameter, and returns the master public key mpk and master secret key msk.
- $\mathsf{ExtractPartialKey}(msk, ID)$ takes msk and a user identity ID as input and returns a user partial private key D_{ID}.
- $\mathsf{SetSecretValue}(mpk, ID)$ takes mpk and a user identity ID as input and returns a user secret value S_{ID}.
- $\mathsf{SetPublicKey}(mpk, S_{ID}, [D_{ID}])$ takes mpk, S_{ID} and possibly D_{ID} as input and returns a user public key pk_{ID}.
- $\mathsf{SetPrivateKey}(mpk, D_{ID}, S_{ID})$ takes mpk, D_{ID}, and S_{ID} as input and returns a user private key sk_{ID}.
- $\mathsf{Enc}(mpk, ID, pk_{ID}, m)$ takes mpk, a user identity ID, a user public key pk_{ID}, and a message m as input and returns a ciphertext c.
- $\mathsf{Dec}(mpk, sk_{ID}, c)$ takes mpk, a user private key sk_{ID}, and a ciphertext c as input and returns a plaintext m or a failure symbol $\perp$.

In the original definition by Al-Riyami and Paterson [2], the SetPublicKey algorithm only takes the master public key mpk and a user secret value S_{ID} as input. In the definition given by Beak et al. [4], the SetPublicKey algorithm additionally takes D_{ID} as part of the input. In [10], Fiore et al. named certificateless schemes which follow the original definition as *pure*, and those following Baek et al.'s formulation as *non-pure*. It is worth noting that, for non-pure schemes, we can combine the SetSecretValue, SetPrivateKey, SetPublicKey algorithms into a single SetUserKeys algorithm

- $\mathsf{SetUserKeys}(mpk, ID, D_{ID})$ takes mpk, a user identity ID and a partial private key D_{ID} as input and returns a public/private key pair (pk_{ID}, sk_{ID}).

CLE security model. There are two types of adversaries, A_I and A_{II}. A Type-I adversary (or A_I) is able to replace the user public key pk_{ID}, since in certificateless cryptosystems, there is no authentication information (such as a X.509 certificate in PKI) for the user public key. In contrast, a Type-II adversary (or A_{II}) is an honest-but-curious KGC who controls msk and hence the user partial private key.

Below we present the adversarial game [6,8,10] for defining the security of a CLE scheme. X can be instantiated with I or II in the description below, st is some state information, and m_0 and m_1 are messages of equal length.

Type-X Adversarial Game
1 $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$
2 $(ID^*, st, m_0, m_1) \leftarrow {A_1}^{\mathcal{O}}(mpk, [msk])$
3 $b \leftarrow \{0, 1\}$
4 $c^* \leftarrow \mathsf{Enc}(mpk, ID^*, pk_{ID^*}, m_b)$
5 $b' \leftarrow {A_2}^{\mathcal{O}}(c^*, st)$.

In line 2 of the game, msk is only passed to the adversary in the Type-II game. And in line 4 of the game, pk_{ID^*} refers to the *current* public key of ID^*. The advantage of the adversary is defined to be

$$\mathbf{Adv}^{\text{Type-X}}_{\mathcal{CLE},A}(k) = |2\Pr[b' = b] - 1|.$$

We now turn to the oracles $\mathcal{O}$ available to the adversaries in each game.

- **RequestPublicKey**(ID): Given an identity ID this oracle returns to the adversary pk_{ID}. If the identity has no associated public key, then a public key is generated by running the ExtractPartialKey (only needed in the non-pure case), SetSecretValue, and SetPublicKey algorithms.
- **ReplacePublicKey**(ID, pk'): Given an identity ID and a public key pk' in the public key space, this oracle allows the adversary to replace ID's public key with pk'.
- **ExtractPartialPrivateKey**(ID): Given an identity ID, this oracle returns the partial private key D_{ID}.
- **ExtractFullPrivateKey**(ID): Given an identity ID, this oracle returns the full private key sk_{ID}.
- **StrongDecrypt**(ID, c): Given an identity ID and a ciphertext c, this oracle returns the decryption of c. If the adversary has replaced the public key of ID, then the decryption is performed using the private key corresponding to the *current* public key.

Strong Type-I Security: The adversary has the following restrictions in accessing the oracles

1. A cannot extract the full private key for ID^*.
2. A cannot extract the full private key for any identity if the corresponding public key has been replaced.
3. A cannot replace the public key for ID^* before c^* has been issued *and* extract the partial private key for ID^* (at any point).
4. A_2 cannot make the StrongDecrypt query on (ID^*, c^*) unless ID^*'s public key used to create c^* has been replaced.

Strong Type-II Security: In the Type-II game, the adversary is given the master secret key msk, and has the following restrictions in accessing the oracles

1. A cannot extract the full private key for ID^*.
2. A cannot extract the full private key for any identity if the corresponding public key has been replaced.
3. A cannot make any partial private key extraction query.
4. A_1 cannot output an identity ID^* for which it has replaced the public key.
5. A_2 cannot make the StrongDecrypt query on (ID^*, c^*) unless ID^*'s public key used to create c^* has been replaced.

A certificateless encryption scheme $\mathcal{CLE}$ is said to be CCA secure in the Type-X model if for any PPT adversary A, $\mathbf{Adv}^{\text{Type-X}}_{\mathcal{CLE},A}(k)$ is a negligible function of the security parameter k.

Remark. Similar to [10], we put a special restriction on a Type-II adversary, that is a Type-II adversary cannot make any partial private key extraction query. The reason is that since a Type-II adversary is an honest-but-curious KGC who generates user partial private keys, so he already knows all the user partial private keys.

In the StrongDecrypt query, we require the decryption to be performed using the private key corresponding to the *current* public key of a user even if the adversary has replaced the public key of that user. This is a very strong oracle. Alternatively, one can define weaker decryption oracles where the decryption is performed using the original private key, or the adversary additionally provides a secret value S'_{ID} in a decryption query and a private key $sk'_{ID} \leftarrow \mathsf{SetPrivateKey}(mpk, D_{ID}, S'_{ID})$ is computed and used to perform the decryption. See [6,8,10] for details and discussions on these variants.

CPA Security. The above definition captures security under chosen-ciphertext attacks (or CCA security). The weaker chosen plaintext attack (or CPA) security notion is formalized by a similar game where attackers have no decryption oracles.

3.2. Certificateless key encapsulation mechanism

Similar to the notion of key encapsulation mechanisms (KEMs) in the public-key setting [18], one can define certificateless key encapsulation mechanisms (CL-KEMs) [6,10].

Definition 2. A certificateless key encapsulation mechanism $\mathcal{CL}$-$\mathcal{KEM}$ consists of the following algorithms.

- $\mathsf{Setup}(1^k)$ takes 1^k as input where k is the security parameter, and returns the master public key mpk and master secret key msk.
- $\mathsf{ExtractPartialKey}(msk, ID)$ takes msk and a user identity ID as input and returns a partial private key D_{ID}.
- $\mathsf{SetSecretValue}(mpk, ID)$ takes mpk and a user identity ID as input and returns a user secret value S_{ID}.
- $\mathsf{SetPublicKey}(mpk, S_{ID}, [D_{ID}])$ takes mpk, S_{ID} and possibly D_{ID} as input and returns a user public key pk_{ID}.
- $\mathsf{SetPrivateKey}(mpk, D_{ID}, S_{ID})$ takes mpk, D_{ID}, and S_{ID} as input and returns a user private key sk_{ID}.
- $\mathsf{Encap}(mpk, ID, pk_{ID})$ takes mpk, a user identity ID, and a user public key pk_{ID} as input and returns a pair (C, K) where K is a symmetric key for a data encapsulation mechanism (DEM), and C is the encapsulation of the key K.
- $\mathsf{Decap}(mpk, sk_{ID}, C)$ takes mpk, a user private key sk_{ID}, and an encapsulation C as input and returns the corresponding key K or a failure symbol $\perp$.

CL-KEM security model. To define the security model for CL-KEMs we simply adapt the security model for certificateless encryption in Section 3.1 into the KEM framework. Again there are two types of adversary against a CL-KEM: Type-I and Type-II. Each adversary is trying to win the following games, where the various oracle accesses allowed are identical (except that we replace the word "decryption" with "decapsulation") to those defined in Section 3.1.

Type-X Adversarial Game

1 $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$
2 $(ID^*, st) \leftarrow A_1{}^{\mathcal{O}}(mpk, [msk])$
3 $(K_0, C^*) \leftarrow \mathsf{Encap}(mpk, ID^*, pk_{ID^*})$
4 $K_1 \leftarrow \mathsf{KeySpace}(mpk)$
5 $b \leftarrow \{0, 1\}$
6 $b' \leftarrow A_2{}^{\mathcal{O}}(C^*, st, K_b)$.

In line 2 of the game, msk is only passed to the adversary in the Type-II game. And in line 3 of the game, pk_{ID^*} refers to the *current* public key of ID^*. The advantage of the adversary is defined to be

$$\mathbf{Adv}^{\text{Type-X}}_{\mathcal{CL}\text{-}\mathcal{KEM},A}(k) = |2\Pr[b' = b] - 1|.$$

A CL-KEM scheme $\mathcal{CL}$-$\mathcal{KEM}$ is said to be CCA secure in the Type-X model if, for any PPT adversary A, $\mathbf{Adv}^{\text{Type-X}}_{\mathcal{CL}\text{-}\mathcal{KEM},A}(k)$ is a negligible function of the security parameter k.

For the Strong Type-I Security, we can define a slightly weaker variant denoted by Strong Type-I* Security in which the adversary cannot extract partial private key of the target identity ID^* at any point. And the following Theorem says that full Strong Type-I Security is implied by Strong Type-I* Security and Strong Type-II Security.

Theorem 1 (*Theorem 3 of [10]*)**.** *If a CL-KEM is Strong Type-I* and Strong Type-II secure, then it is Strong Type-I secure.*

In [6], Bentahar et al. showed that the KEM/DEM (DEM stands for Data Encapsulation Mechanism) paradigm [18] can be extended to the certificateless setting, that is, we can construct a hybrid certificateless encryption scheme by combining a CL-KEM scheme with a DEM scheme.

3.3. Identity-based key agreement protocol

A two-pass one-way Identity-Based Key Agreement (IB-KA) protocol $\mathcal{P}$ consists of the following algorithms.

- $\mathsf{KASetup}(1^k)$ takes a security parameter 1^k as input, and returns a master public/secret key pair (mpk^{KA}, msk^{KA}).
- $\mathsf{KeyDer}(msk^{KA}, ID)$ takes the master secret key msk^{KA} and an identity ID as input and returns user private key d_{ID}.
- $\mathsf{Initiate}(mpk^{KA}, [d_I])$ is an algorithm run by the protocol initiator. It takes as input the master public key mpk^{KA} and possibly also the user private key d_I of the initiator I, and returns a pair of ephemeral public/private keys (epk_I, esk_I). The identity I of the initiator and the ephemeral public key epk_I are sent to the responder R of the protocol.
- $\mathsf{Response}(mpk^{KA})$ is an algorithm run by the protocol responder R. It takes mpk^{KA} as input and returns a pair of ephemeral public/private keys (epk_R, esk_R). The ephemeral public key epk_R is sent to the initiator I.
- $\mathsf{Derive}_I(mpk^{KA}, d_I, esk_I, epk_R)$ is a key derivation algorithm run by the initiator I. It returns a session key K_I.
- $\mathsf{Derive}_R(mpk^{KA}, esk_R, epk_I, I)$ is a key derivation algorithm run by the responder R. It returns a session key K_R.

In the above definition, we explicitly assume that the responder is the one who is anonymous and not authenticated. In [10], Fiore et al. also separate two-pass IB-KA protocols into two categories: *pure* and *non-pure*. Pure IB-KA protocols refer to those do not take the user private key d_I as part of the input to the Initiate algorithm, while non-pure protocols do take d_I as part of the input to the Initiate algorithm.

IB-KA security model. Security of a two-pass one-way IB-KA protocol is defined by a game between an adversary A and a challenger C. At the beginning of the game, C generates a master public/private key pair (mpk^{KA}, msk^{KA}) and gives mpk^{KA} to A. Each party U created in the game holds a private key d_U that is generated by the challenger C by running $\mathsf{KeyDer}(msk^{KA}, U)$. A party may run many instances concurrently, we denote instance i of party U by Π_U^i. Instances of parties are modeled as oracles available to the adversary. In the game, A can make the following oracle queries:

- **CreateUser**(U): By this query, the adversary creates a user with identity U. Upon receiving the query, C executes $\mathsf{KeyDer}(msk^{KA}, U)$ to generate a user private key d_U.
- **NewSession**(U, i): By this query, the adversary activates an instance Π_U^i of U to start a session (i.e., Π_U^i has a $\mathsf{role}_U^i =$ initiator) where i is the instance ID which uniquely identifies the instance within U. Upon receiving the query, C executes $\mathsf{Initiate}(mpk^{KA}, [d_U])$ to create a pair of ephemeral public/private keys (epk_U^i, esk_U^i) and returns epk_U^i to the adversary.
- **Send**(U, i, V, epk): By this query, the adversary sends a message (V, epk) to instance Π_U^i. There are two cases to consider:
  1. If Π_U^i has appeared in a NewSession query (which means Π_U^i is an initiator who has generated a ephemeral key pair (epk_U, esk_U) before), then the challenger executes $\mathsf{Derive}_I(mpk^{KA}, d_U, esk_U, epk)$ to generate a session key sk_U^i. The identity V is ignored by the challenger in this case.
  2. If Π_U^i has not appeared in a NewSession query before, then the challenger sets $\mathsf{role}_U^i =$ responder and V as the partner-id of Π_U^i (denoted by pid_U^i). The challenger then executes $\mathsf{Response}(mpk^{KA})$ to generate (epk_R, esk_R), and then executes $\mathsf{Derive}_R(mpk^{KA}, esk_R, epk, V)$ to generate a session key sk_U^i. The challenger returns epk_R to the adversary.
- **Reveal**(U, i): By making this query to an instance Π_U^i that has accepted, A learns the session key sk_U^i that has been generated by Π_U^i.
- **Corrupt**(U): By this query, the adversary learns the private key d_U of party U.
- **Test**(U^*, i^*): This query is only made once by the adversary during the game, and the instance $\Pi_{U^*}^{i^*}$ must have $\mathsf{role}_{U^*}^{i^*} =$ responder, have accepted the conversation, and is *fresh* (defined below). Upon receiving the query, a random coin b is flipped by the challenger C. If the coin $b = 1$, then $sk_{U^*}^{i^*}$ is returned to the adversary. Otherwise, if $b = 0$, then a random session key is drawn from the session key space and returned to the adversary.

At the end of the game the adversary A outputs a bit b' as its guess to the bit b. We say the adversary wins the game if $b' = b$. We define the advantage of the adversary by

$$\mathbf{Adv}_{\mathcal{P},A}^{\mathsf{IB\text{-}KA}}(k) = |2\Pr[b' = b] - 1|.$$

We say two instances have a matching conversation if their communication transcripts, defined as $\mathsf{CommTrans} = (\mathit{Initiator\text{-}id}, epk_I, epk_R)$, are equivalent. We say an instance $\Pi_{U^*}^{i^*}$ with $\mathsf{role}_{U^*}^{i^*} =$ responder and $\mathsf{pid}_{U^*}^{i^*} = V^*$ is *fresh* if

1. A does not make a Reveal query to the instance $\Pi_{U^*}^{i^*}$.
2. A does not make a Corrupt query to V^*.
3. A does not make a Reveal query to an instance $\Pi_{V^*}^{j}$ which has a matching conversation with $\Pi_{U^*}^{i^*}$ (if such a $\Pi_{V^*}^{j}$ exists within V^*).

IB-KA Security. We say a protocol $\mathcal{P}$ is a secure two-pass one-way IB-KA protocol if

1. In the presence of a benign adversary who faithfully conveys messages, then two instances having a matching conversation output the same session key;
2. For any polynomial time adversary A, $\mathbf{Adv}_{\mathcal{P},A}^{\mathsf{IB\text{-}KA}}(k)$ is negligible.

Forward Secrecy. The conventional notion of perfect forward secrecy means corrupting two parties U and V should not allow the adversary to compromise any session key that has already been established between U and V. We consider two levels of forward secrecy for two-pass one-way IB-KA protocols,

1. Forward Secrecy: the adversary is allowed to corrupt the user private key of the protocol initiator in the Test session;
2. Master-Key Forward Secrecy: the adversary is given the master secret key msk^{KA} and knows all the user private keys (i.e., the adversary is an honest-but-curious KGC).

We say an instance $\Pi_{U^*}^{i^*}$ with $\mathsf{role}_{U^*}^{i^*} = \mathsf{responder}$, $\mathsf{pid}_{U^*}^{i^*} = V^*$ and communication transcript $(\mathsf{pid}_{U^*}^{i^*}, epk_{V^*}, epk_{U^*})$ is fresh in the forward-secrecy (or master-key-forward-secrecy) model if

1. A does not make a Reveal query to the instance $\Pi_{U^*}^{i^*}$.
2. The ephemeral public key epk_{V^*} must have been generated by an instance $\Pi_{V^*}^{j}$ within the party $\mathsf{pid}_{U^*}^{i^*}$.
3. A does not make a Reveal query to an instance $\Pi_{V^*}^{j}$ which has a matching conversation with $\Pi_{U^*}^{i^*}$ (if such a $\Pi_{V^*}^{j}$ exists).

Note that in the forward-secrecy (FS) model and the master-key-forward-secrecy (MKFS) model, we require that the ephemeral public key epk_{V^*} must be honestly generated by the user $V^* = \mathsf{pid}_{U^*}^{i^*}$. Such a requirement is necessary for two-pass key agreement protocols [15,7] when considering forward secrecy.

The Reveal* Model. In [10], the authors proposed a stronger security model called Reveal* model in which a new oracle query, named Reveal* query, is introduced.

– **Reveal***(I, epk_I, epk_R): By making this query, the adversary A learns the associated session key assuming a conversation (I, epk_I, epk_R) between party I and a responder had taken place.

The Reveal* query is a very strong query in the sense that it is not associated to any instance, and the conversation can never take place during the game. This query is introduced in [10] in order to handle the Strong Decapsulation query when transforming IB-KA protocols to CL-KEM schemes using the FGS transformation. An obvious restriction in the Reveal* model is that the adversary cannot make a Reveal* query to the Test session.

4. The Fiore–Gennaro–Smart transformation

In [10], Fiore et al. presented a generic way to transform two-pass one-way Identity-Based Key Agreement (IB-KA) protocols to Certificateless Key Encapsulation Mechanisms.

4.1. The FGS transformation

Suppose we are given a two-pass one-way authenticated IB-KA protocol. We construct a CL-KEM scheme as follows.

– Setup: run $\mathsf{KASetup}(1^k)$ to generate mpk^{KA} and msk^{KA}, set $mpk = mpk^{KA}$ and $msk = msk^{KA}$.
– ExtractPartialKey: run $\mathsf{KeyDer}(msk, ID)$ to generate d_{ID}, set $D_{ID} = d_{ID}$.
– SetSecretValue, SetPublicKey: run $\mathsf{Initiate}(mpk, [D_{ID}])$ to generate (epk, esk), set $S_{ID} = esk$ and $pk_{ID} = epk$.
– SetPrivateKey: set $sk_{ID} = (D_{ID}, S_{ID})$.
– Encap: compute $(epk', esk') \leftarrow \mathsf{Response}(mpk)$, $K \leftarrow \mathsf{Derive}_R(mpk, esk', pk_{ID}, ID)$, and $C \leftarrow epk'$. Return (C, K).
– Decap: compute $K \leftarrow \mathsf{Derive}_I(mpk, D_{ID}, S_{ID}, C)$ and return K.

The following results regarding the transformation have been obtained in [10].

Theorem 2 (*Theorem 4 of [10]*)**.** *If IB-KA is a secure two-pass one-way identity-based key agreement protocol in the Reveal* model, then the corresponding CL-KEM obtained via the FGS transform is Strong Type-I* secure.*

Theorem 3 (*Theorem 6 of [10]*)**.** *If IB-KA is a secure two-pass one-way authenticated identity-based key agreement protocol in the (MKFS, Reveal*) model, then the corresponding CL-KEM obtained via the FGS transform is Strong Type-II secure.*

Below we will look at a concrete example illustrating the transform.

4.2. A pairing-free CL-KEM scheme from the Fiore–Gennaro IB-KA protocol

In Fig. 1, we present the *non-pure* Fiore–Gennaro (FG) IB-KA protocol [11] and the corresponding CL-KEM scheme derived by applying the FGS transformation. In [10], the following results are obtained.

Theorem 4 (*Theorem 2 of [10]*)**.** *The FG Protocol is secure in the Reveal* model, and (MKFS, Reveal*) model, assuming Gap-DH problem is hard.*

Corollary 1 (*Corollary 1 of [10]*)**.** *The FG CL-KEM is Strong Type-I* and Strong Type-II (and hence Strong Type-I) secure assuming the Gap-DH problem is hard.*

Fiore–Gennaro IB-KA Protocol:

$\mathsf{KASetup}(1^k)$:
$x \leftarrow \mathbb{Z}_q, y \leftarrow g^x, msk^{KA} \leftarrow x, mpk^{KA} \leftarrow y.$
$\mathsf{KeyDer}(1^k)$:
$\alpha \leftarrow \mathbb{Z}_q, r_{ID} \leftarrow g^\alpha, z_{ID} \leftarrow \alpha + H_1(ID\|r_{ID})x,$
$d_{ID} \leftarrow (r_{ID}, z_{ID}).$
$\mathsf{Initiate}(mpk^{KA}, d_{ID})$:
$esk_{ID} \leftarrow \mathbb{Z}_q, epk_{ID} \leftarrow (r_{ID}, g^{esk_{ID}}).$
$\mathsf{Response}(mpk^{KA})$:
$esk' \leftarrow \mathbb{Z}_q, epk' \leftarrow g^{esk'}.$
$\mathsf{Derive}_I(mpk^{KA}, d_{ID}, esk_{ID}, epk')$:
$c_1 \leftarrow epk'^{z_{ID}+esk_{ID}}, c_2 \leftarrow epk'^{esk_{ID}},$
$K_I \leftarrow H_2(c_1, c_2).$
$\mathsf{Derive}_R(mpk^{KA}, esk', epk_{ID}, ID)$:
$c_1 \leftarrow (g^{esk_{ID}} \cdot r_{ID} \cdot mpk^{H_1(ID\|r_{ID})})^{esk'},$
$c_2 \leftarrow (g^{esk_{ID}})^{esk'},$
$K_R \leftarrow H_2(c_1, c_2).$

The corresponding FG CL-KEM:

$\mathsf{Setup}(1^k)$:
$x \leftarrow \mathbb{Z}_q, y \leftarrow g^x, msk \leftarrow x, mpk \leftarrow y.$
$\mathsf{ExtractPartialKey}(msk, ID)$:
$\alpha \leftarrow \mathbb{Z}_q, r_{ID} \leftarrow g^\alpha,$
$z_{ID} \leftarrow \alpha + H_1(ID\|r_{ID})x,$
$D_{ID} \leftarrow (r_{ID}, z_{ID}).$
$\mathsf{SetSecretValue}(mpk, ID)$:
$S_{ID} \leftarrow \mathbb{Z}_q.$
$\mathsf{SetPublicKey}(mpk, S_{ID}, D_{ID})$:
$U_{ID} \leftarrow g^{S_{ID}}, pk_{ID} \leftarrow (r_{ID}, U_{ID}).$
$\mathsf{SetPrivateKey}$:
$sk_{ID} \leftarrow (D_{ID}, S_{ID}).$
$\mathsf{Encap}(mpk, pk_{ID}, ID)$:
$t \leftarrow \mathbb{Z}_q, C \leftarrow g^t,$
$c_1 \leftarrow (U_{ID} \cdot r_{ID} \cdot mpk^{H_1(ID\|r_{ID})})^t,$
$c_2 \leftarrow U_{ID}^t,$
$K \leftarrow H_2(c_1, c_2).$
$\mathsf{Decap}(mpk, sk_{ID}, C)$:
$c_1 \leftarrow C^{S_{ID}+z_{ID}}, c_2 \leftarrow C^{S_{ID}}$
$K \leftarrow H_2(c_1, c_2).$

Fig. 1. A pairing-free CL-KEM scheme from the *non-pure* Fiore–Gennaro IB-KA protocol.

5. Analysis of the FG CL-KEM scheme

In this section, we show that the CL-KEM scheme in Fig. 1 is neither Strong Type-I nor Strong Type-II secure, which means Corollary 1 of [10] is incorrect.

A Strong Type-I adversary A_I
- The adversary A_I first issues two Request Public Key queries for two identity ID^* and ID. Denote the corresponding keys by $\{D_{ID^*} = (r_{ID^*}, z_{ID^*}), S_{ID^*}, U_{ID^*} = g^{S_{ID^*}}\}$ and $\{D_{ID} = (r_{ID}, z_{ID}), S_{ID}, U_{ID} = g^{S_{ID}}\}$, respectively.
- The adversary A_I then selects ID^* as the challenge identity, and gets back a challenge (C^*, K).
- A_I issues two Extract Partial Private Key queries to learn D_{ID^*} and D_{ID}.
- A_I issues a Replace Public Key query with input $(ID, pk'_{ID} = (r_{ID}, U_{ID^*}^{\frac{z_{ID}}{z_{ID^*}}}))$.
- A_I then issues a Strong Decapsulation query with input $(ID, C^{*\frac{z_{ID^*}}{z_{ID}}})$. Denote K' the value returned by the Strong Decapsulation oracle.
- A_I compares K' with K. If they are equal, return 0; otherwise, return 1.

We can see that A_I does not violate the rules for a Strong Type-I adversary defined in Section 3. In the following we analyze the winning probability of the adversary.

1. Let K^* denotes the real key encapsulated in C^*. Then we have $K^* = H_2(c_1^*, c_2^*)$ where $c_1^* = C^{*S_{ID^*}+z_{ID^*}}$ and $c_2^* = C^{*S_{ID^*}}$.
2. After A_I replaces the public key of ID with $(r_{ID}, U_{ID^*}^{\frac{z_{ID}}{z_{ID^*}}})$, D_{ID} remains unchanged, and S_{ID} now becomes $S_{ID^*}\frac{z_{ID}}{z_{ID^*}}$.
3. When A_I issues the Strong Decapsulation query with $(ID, C^{*\frac{z_{ID^*}}{z_{ID}}})$, the following steps are performed according to the decapsulation algorithm
   (a) Compute

$$c_1 = (C^{*\frac{z_{ID^*}}{z_{ID}}})^{S_{ID^*}\frac{z_{ID}}{z_{ID^*}}+z_{ID}} = C^{*S_{ID^*}+z_{ID^*}}$$
$$c_2 = (C^{*\frac{z_{ID^*}}{z_{ID}}})^{S_{ID^*}\frac{z_{ID}}{z_{ID^*}}} = C^{*S_{ID^*}}.$$

   (b) Compute $K' = H_2(c_1, c_2)$

Now we can see that $K' = K^*$, so A_I can win the game with a probability almost equal to 1.

Similarly, we can also construct a Type-II adversary A_{II} to win the game. Notice that since A_{II} is an honest-but-curious KGC, it does not need to make any partial private key query.

Errors in [10]. Our attack disproves some claims in [10], in particular, Corollary 1. So something must be wrong in the security analysis in [10].

Modified Fiore–Gennaro IB-KA Protocol:

$\mathsf{KASetup}(1^k)$:
$x \leftarrow \mathbb{Z}_q, y \leftarrow g^x, msk^{KA} \leftarrow x, mpk^{KA} \leftarrow y.$
$\mathsf{KeyDer}(1^k)$:
$\alpha \leftarrow \mathbb{Z}_q, r_{ID} \leftarrow g^\alpha, z_{ID} \leftarrow \alpha + H_1(ID\|r_{ID})x,$
$d_{ID} \leftarrow (r_{ID}, z_{ID}).$
$\mathsf{Initiate}(mpk^{KA}, d_{ID})$:
$esk_{ID} \leftarrow \mathbb{Z}_q, epk_{ID} \leftarrow (r_{ID}, g^{esk_{ID}}).$
$\mathsf{Response}(mpk^{KA})$:
$esk' \leftarrow \mathbb{Z}_q, epk' \leftarrow g^{esk'}.$
$\mathsf{Derive}_I(mpk^{KA}, d_{ID}, esk_{ID}, epk')$:
$c_1 \leftarrow epk'^{z_{ID}+esk_{ID}}, c_2 \leftarrow epk'^{esk_{ID}},$
$K_I \leftarrow H_2(ID, epk_{ID}, epk', c_1, c_2).$
$\mathsf{Derive}_R(mpk^{KA}, esk', epk_{ID}, ID)$:
$c_1 \leftarrow (g^{esk_{ID}} \cdot r_{ID} \cdot mpk^{H_1(ID\|r_{ID})})^{esk'},$
$c_2 \leftarrow (g^{esk_{ID}})^{esk'},$
$K_R \leftarrow H_2(ID, epk_{ID}, epk', c_1, c_2).$

The corresponding FG-1 CL-KEM:

$\mathsf{Setup}(1^k)$:
$x \leftarrow \mathbb{Z}_q, y \leftarrow g^x, msk \leftarrow x, mpk \leftarrow y.$
$\mathsf{ExtractPartialKey}(msk, ID)$:
$\alpha \leftarrow \mathbb{Z}_q, r_{ID} \leftarrow g^\alpha,$
$z_{ID} \leftarrow \alpha + H_1(ID\|r_{ID})x,$
$D_{ID} \leftarrow (r_{ID}, z_{ID}).$
$\mathsf{SetSecretValue}(mpk, ID)$:
$S_{ID} \leftarrow \mathbb{Z}_q.$
$\mathsf{SetPublicKey}(mpk, S_{ID}, D_{ID})$:
$U_{ID} \leftarrow g^{S_{ID}}, pk_{ID} \leftarrow (r_{ID}, U_{ID}).$
$\mathsf{SetPrivateKey}$:
$sk_{ID} \leftarrow (D_{ID}, S_{ID}).$
$\mathsf{Encap}(mpk, pk_{ID}, ID)$:
$t \leftarrow \mathbb{Z}_q, C \leftarrow g^t,$
$c_1 \leftarrow (U_{ID} \cdot r_{ID} \cdot mpk^{H_1(ID\|r_{ID})})^t,$
$c_2 \leftarrow U_{ID}^t,$
$K \leftarrow H_2(ID, pk_{ID}, C, c_1, c_2).$
$\mathsf{Decap}(mpk, sk_{ID}, C)$:
$c_1 \leftarrow C^{S_{ID}+z_{ID}}, c_2 \leftarrow C^{S_{ID}}$
$K \leftarrow H_2(ID, pk_{ID}, C, c_1, c_2).$

Fig. 2. The FG-1 pairing-free CL-KEM scheme from the modified FG IB-KA protocol.

The problem lies in the security analysis for Fiore–Gennaro IB-KA protocol, we can see that our attack above can be transformed to an attack against the FG IB-KA protocol in the (forward-secrecy, Reveal*) model. Given a key agreement session $\mathcal{S} = (I, epk_I = (r_I, g^{esk_I}), epk_R = g^{esk_R})$, the adversary can construct another communication transcript

$$\mathcal{S}' = (I', epk_{I'} = (r_{I'}, (g^{esk_I})^{\frac{z_{I'}}{z_I}}), epk_{R'} = epk_R^{\frac{z_I}{z_{I'}}})$$

which would have the same session key $H_2(epk_R^{esk_I+z_I}, epk_R^{esk_I})$ as that of session $\mathcal{S}$. So Theorem 2 of [10] is incorrect.

6. A new pairing-free CL-KEM scheme

From the analysis in the previous section, we can see that the insecurity of the FG CL-KEM is due to the insecurity of the underlying IB-KA protocol. In this section, we do a slight modification on the FG IB-KA protocol, and show a new CL-KEM (denoted by FG-1) which can achieve Strong Type-I and Type-II security, based on this modified protocol (Fig. 2).

Our modification is very simple, we only change the way to derive the final key K. In the context of key agreement protocols, in fact we add the communication transcript in the derivation of the final session key. The idea is that now even the keying materials (namely c_1 and c_2) are the same, as far as the communication transcripts are different, the session keys would be different. In this way, we can answer the Reveal* query using the random oracle even if the adversary fabricates a session which is different from the Test session (i.e., has a different communication transcript), but has the same c_1 and c_2. In the following, we show that such a modification is already sufficient in order to provide Strong Type-I and Type-II security.

Theorem 5. *The modified FG IB-KA protocol is secure in the Reveal* model assuming H_1 and H_2 are random oracles and the Gap-DH problem is hard.*

Proof. The proof essentially follows that in [11]. The major change is that now we need to additionally simulate the Reveal* oracle. Let A denotes an adversary who can break the modified FG IB-KA protocol in the Reveal* model, we construct another PPT algorithm B, and a Gap-DH problem solver S_B associated with B as follows.

B receives a tuple $(g, U = g^u, V = g^v)$ as input where u, v are randomly selected from $\mathbb{Z}_q$. B simulates the Reveal* game for A by answering the Hash Oracle (H_1 and H_2) queries, Party Corruption queries, Reveal and Reveal* queries, and simulating protocol sessions including the Test session.

B first sets the master public key $mpk^{KA} = U$. The H_1 and Corruption queries are simulated in the same way as in [11]. The H_1 oracle is simulated by consistently returning random elements of $\mathbb{Z}_q$, and the user secret keys are simulated by running the Honest Verifier Zero Knowledge (HVZK) proof simulator of the Schnorr identification protocol. There is one exceptional case: for a particular party (say Carol), B randomly selects $k_C \leftarrow \mathbb{Z}_q$ and sets $sk_C \leftarrow (r_C \leftarrow g^{k_C}, \top)$ where $\top$ means the value is unknown to B. B guesses that Carol will be the protocol initiator in the Test session. Notice that since we do not consider forward secrecy here, if B's guess is correct, the adversary would never ask for the secret key sk_C.

To simulate the H_2 queries, B maintains a table T_2. On input a tuple $(ID, epk_{ID}, epk_R, c_1, c_2)$, a random string Z in the session key space is returned, and an entry $(ID, epk_{ID}, epk_R, c_1, c_2, Z)$ is added into T_2. When later the same input is asked to the H_2 oracle, the same string Z is returned.

Since B knows all the private keys except sk_C, B can simulate all the sessions except those of Carol. To simulate a session of Carol, when Carol is a responder, B can simulate the session perfectly since we are in the one-way authentication setting. When Carol is an initiator, and the responders message is injected by the adversary, B uses its DH oracle to answer session-key Reveal queries. Let $C, r_C, u_C(=g^{t_C})$ denote the message sent by B to the adversary. Recall that the session key is $H_2(C, r_C, u_C, epk_R = g^{t_R}, c_1, c_2)$ (t_R is selected by the adversary and unknown to B) where $c_1 = epk_R^{z_C+t_C}$ and $c_2 = epk_R^{t_C}$. B searches the table T_2 for a tuple $(C, r_C, u_C, epk_R, \cdot, c_2, \cdot)$. If there is a tuple $(C, r_C, u_C, epk_R, \top, c_2, Z)$, B returns Z to the adversary. If a tuple $(C, r_C, u_C, epk_R, c_1', c_2, Z)$ is found, B computes $X = u_C r_C U^{H_1(C\|r_C)}$. B asks its DH oracle with input (X, epk_R, c_1'). If the DH oracle returns 1 (which indicates $c_1' = c_1$), B returns Z to the adversary. Otherwise, B searches for the next entry. If there is no entry in T_2 which makes the DH oracle return 1, B selects a random Z in the session key space, adds $(C, r_C, u_C, epk_R, \top, c_2, Z)$ into T_2 and returns Z. Later, when the adversary makes a H_2 query with a new input $(C, r_C, u_C, epk_R, \lambda, c_2)$, and there exists a tuple $(C, r_C, u_C, epk_R, \top, c_2, Z)$ in T_2, B performs the test as above using the DH oracle. If the DH oracle returns 1, B updates the $\top$ filed of the entry with λ and returns Z. Otherwise, B answers the H_2 query as usual.

Suppose B asks a Reveal* query with input $(ID, r_{ID}', u_{ID}', epk_R)$. Notice that here r_{ID}' may not be equal to r_{ID} that is created by B. If $(ID, r_{ID}', u_{ID}', epk_R)$ is a really executed session, then it is answered as in a Reveal query.

Now suppose the session $(ID, r_{ID}', u_{ID}', epk_R)$ has never appeared in a real execution before. Upon receiving the query, B searches the table T_2 for an entry $(ID, r_{ID}', u_{ID}', epk_R, \cdot, \cdot, \cdot)$.

1. If there is an entry $(ID, r_{ID}', u_{ID}', epk_R, \top, \top, Z)$ then the value Z is returned.
2. If there is an entry $(ID, r_{ID}', u_{ID}', epk_R, c_1, c_2, Z)$, B first asks the DH oracle on input (u_{ID}', epk_R, c_2). If the DH oracle returns 1, B computes $X = u_{ID}' r_{ID}' U^{H_1(ID\|r_{ID}')}$, and asks its DH oracle again with input (X, epk_R, c_1). If the DH oracle returns 1, B returns Z to the adversary. Otherwise, B searches for the next entry.
3. If no valid entry is found in the above steps, B selects a random Z in the session key space, adds $(ID, r_{ID}', u_{ID}', epk_R, \top, \top, Z)$ into T_2 and returns Z.
4. Later, when the adversary makes a H_2 query with a new input $(ID, r_{ID}', u_{ID}', epk_R, \lambda_1, \lambda_2)$, and there exists a tuple $(ID, r_{ID}', u_{ID}', epk_R, \top, \top, Z)$ in T_2, B performs the test as above using the DH oracle. If the test is successful, B updates the two $\top$ fields of the entry with (λ_1, λ_2) and returns Z. Otherwise, B answers the H_2 query as usual.

Now let's consider a Test session (C, r_C^*, u_C^*, epk_R) where the simulator B sets $epk_R = V$. Notice that the adversary must choose a responding instance in the Test query, so B can set V as the ephemeral public key of that instance. Also, the adversary cannot issue a Reveal or Reveal* query to the Test session. B returns a random string in the session key space to A upon receiving the Test query.

Let E denote the event that the adversary A asks a H_2 query with input $(C, r_C^*, u_C^*, V, c_1^*, c_2^*)$ where $\mathrm{DH}(u_C^*, V, c_2^*) = 1$ and $\mathrm{DH}(u_C^* r_C^* U^{H_1(C\|r_C^*)}, V, c_1^*) = 1$. If event E does not occur, the adversary has no advantage in winning the game. So if A can win the game with a non-negligible advantage, event E must occur with a non-negligible probability. If event E occurs, B can derive a value $\tau = c_1^*/c_2^* = (r_C^* U^{H_1(C\|r_C^*)})^v$.

Now similar to [11], given such an algorithm B, we can construct a Gap-DH problem solver S_B which rewinds B. By the General Forking Lemma, with a non-negligible probability, S_B can obtain two values $\tau = (rU^h)^v$ and $\tau' = (r'U^{h'})^v$ where $r = r'$ but $h \neq h'$. Then S_B can compute $W = (\tau/\tau')^{\frac{1}{h-h'}} = g^{uv}$ and solve the Gap-DH problem. □

Theorem 6. *The modified FG IB-KA protocol is secure in the (MKFS, Reveal*) model assuming H_1 and H_2 are random oracles and the Gap-DH problem is hard.*

Proof. Let A denote an adversary who can break the modified FG IB-KA protocol in the (MKFS, Reveal*) model, we construct another PPT algorithm B which solves the Gap-DH problem.

B receives a tuple $(g, U = g^u, V = g^v)$ as input where u, v are randomly selected from $\mathbb{Z}_q$. B simulates the game for A as follows. B first generates the master public/private key pairs by running the KASetup algorithm, and returns (mpk, msk) to A. B simulates the random oracles (H_1 and H_2) as in the previous proof, and generates the user private keys honestly and returns all the user private keys to the adversary. B also simulates those non-test sessions and answers the session key Reveal queries honestly. The Reveal* queries are answered in the same way as in the previous proof.

Now let's consider a Test session (C, r_C, u_C, epk_R). Since in the MKFS model, u_C and epk_R are required to be honestly generated by the two communicating instances, B sets $u_C = U$ and $epk_R = V$, and returns a random string in the session key space to A upon receiving the Test query.

Let E denote the event that the adversary A asks a H_2 query with input $(C, r_C, U, V, c_1^*, c_2^*)$ such that $\mathrm{DH}(U, V, c_2^*) = 1$. If event E does not occur, the adversary has no advantage in winning the game. So if A can win the game with a non-negligible advantage, event E must occur with a non-negligible probability. If event E occurs, B outputs $c_2^* = g^{uv}$ which is the solution for the Gap-DH problem. □

So by Theorems 1, 5 and 6 we have the following corollary,

Corollary 2. *The FG-1 CL-KEM is Strong Type-I and Strong Type-II secure in the random oracle model assuming the Gap-DH problem is hard.*

7. A new generic construction of CLE schemes

The generic transformation given by Fiore et al. is indeed an excellent work. However, one drawback of the transformation is that in order to prove the security of a CL-KEM scheme, a security proof of the underlying IB-KA protocol in the Reveal* model is required, but all the existing IB-KA protocols are only proven in security models where the Reveal* query is not considered. So before applying the transformation to any existing IB-KA protocol, a new security proof for the IB-KA protocol in the Reveal* model is required, otherwise, insecure CL-KEM schemes may be generated via the transformation (as illustrated by our attack against the FG CL-KEM scheme in Section 5). The requirement of re-proving an existing IB-KA protocol in the Reveal* model reduces the usability of the FGS transformation.

7.1. A new generic construction

In this section, we present a new generic construction of CLE schemes from IB-KA protocols. Different from the FGS transformation, we derive provably secure CLE schemes from IB-KA protocols that are only required to be proven secure in a model that is weaker than the existing IB-KA security models. Our idea is to first construct CPA secure CLE schemes from IB-KA protocols, and then convert the CPA secure scheme to a CCA secure one. To do so, we will make use of the Certificateless Fujisaki–Okamoto Transformation due to Libert and Quisquater [16].

From CPA security to CCA security. In [16], Libert and Quisquater presented a modified Fujisaki–Okamoto transformation [12] to construct CCA secure CLE schemes from CPA secure ones. Given a CLE scheme $\mathcal{CLE}$, a new scheme $\mathcal{CLE}'$ can be constructed via the certificateless Fujisaki–Okamoto transformation as follows,

- $\mathcal{CLE}'$.Setup, $\mathcal{CLE}'$.ExtractPartialKey, $\mathcal{CLE}'$.SetSecretValue, $\mathcal{CLE}'$.SetPublicKey, $\mathcal{CLE}'$.SetPrivateKey: the same as that of $\mathcal{CLE}$.
- $\mathcal{CLE}'$.Enc(mpk, ID, pk_{ID}, m): randomly choose $\delta \in \{0,1\}^k$, compute $r \leftarrow H(m, \delta, pk_{ID}, ID)$, $m' \leftarrow m\|\delta$, and $c \leftarrow \mathcal{CLE}$.Enc$(mpk, ID, pk_{ID}, m'; r)$. Return c.
- $\mathcal{CLE}'$.Dec(mpk, sk_{ID}, c): compute $m\|\delta \leftarrow \mathcal{CLE}$.Dec$(mpk, sk_{ID}, c)$, $r' \leftarrow H(m, \delta, pk_{ID}, ID)$, $c' \leftarrow \mathcal{CLE}$.Enc$(mpk, ID, pk_{ID}, m\|\delta; r')$. If $c' = c$, return m. Otherwise, return $\perp$.

where $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ is a cryptographic hash function and $\mathcal{CLE}$.Enc$(\cdot,\cdot,\cdot,\cdot; r)$ means running the encryption algorithm $\mathcal{CLE}$.Enc with randomness $r \in \{0,1\}^\ell$.

Theorem 7 (*Theorem 1 of [16]*)**.** *If $\mathcal{CLE}$ is a CPA secure CLE scheme in Type-I (resp. Type-II) model, then $\mathcal{CLE}'$ is a CCA secure CLE scheme in Strong Type-I (resp. Strong Type-II) model assuming H is a random oracle.*

Constructing CLE from CL-KEM. Below we show that following the CL-KEM/DEM paradigm [6], we can obtain a CPA secure CLE scheme from a CPA secure CL-KEM scheme and a secure data encapsulation mechanism.

The security of a Data Encapsulation Mechanism $\mathcal{DEM} = (\mathcal{DEM}.\mathsf{Enc}, \mathcal{DEM}.\mathsf{Dec})$ with key space $\mathcal{K}$ is defined via a find-then-guess game [5,12] as follows.

$$\mathbf{Adv}^{\mathrm{fg}}_{\mathcal{DEM},A}(k) = \left| 2\Pr\begin{bmatrix} K \leftarrow \mathcal{K};\ (m_0, m_1, s) \leftarrow A(\mathrm{find});\ b \leftarrow \{0,1\}; \\ y \leftarrow \mathcal{DEM}.\mathsf{Enc}(K, m_b) : A(\mathrm{guess}, s, y) = b \end{bmatrix} - 1 \right|.$$

We say a Data Encapsulation Mechanism $\mathcal{DEM}$ is fg-secure if for any polynomial time adversary A, $\mathbf{Adv}^{\mathrm{fg}}_{\mathcal{DEM},A}(k)$ is a negligible function of k.

In the following we show that combining a CPA secure CL-KEM scheme $\mathcal{CL}$-$\mathcal{KEM}$ with a fg-secure DEM scheme $\mathcal{DEM}$, we can obtain a CPA secure CLE scheme $\mathcal{CLE}$.

- $\mathcal{CLE}$.Setup, $\mathcal{CLE}$.ExtractPartialKey, $\mathcal{CLE}$.SetSecretValue, $\mathcal{CLE}$.SetPublicKey, $\mathcal{CLE}$.SetPrivateKey: the same as that of $\mathcal{CL}$-$\mathcal{KEM}$.
- $\mathcal{CLE}$.Enc(mpk, ID, pk_{ID}, m):
  1. compute $(C_1, K) \leftarrow \mathcal{CL}$-$\mathcal{KEM}$.Encap$(mpk, ID, pk_{ID})$;
  2. compute $C_2 \leftarrow \mathcal{DEM}$.Enc$(K, m)$;
  3. set the ciphertext $c \leftarrow (C_1, C_2)$.
- $\mathcal{CLE}$.Dec(mpk, sk_{ID}, c):
  1. parse c into C_1 and C_2;
  2. compute $K' \leftarrow \mathcal{CL}$-$\mathcal{KEM}$.Decap$(mpk, sk_{ID}, C_1)$, if $K' = \perp$, return $\perp$;
  3. compute $m \leftarrow \mathcal{DEM}$.Dec$(K', C_2)$;
  4. return m.

Theorem 8. *If the CL-KEM scheme is Type-I (Type-II, resp.) CPA secure, and the DEM scheme is fg-secure, then the CLE scheme is Type-I (Type-II, resp.) CPA secure.*

Proof. We define a modified game denoted by Type'-X-cpa game (X = I, II). Let $c^* = (C_1^*, C_2^*)$ denote the challenge ciphertext generated by the challenger in the Type-X-cpa game, and K^* the symmetric key encapsulated in C_1^*. In the Type'-X-cpa game, we generate C_2^* using a random key K instead of K^*.

In the following we show that any PPT adversary A has similar advantages in the Type-X-cpa and Type'-X-cpa games, otherwise, we can construct another adversary A' which can break the underlying CL-KEM scheme.

A' simulates the game for A as follows: to answer the RequestPublicKey, ReplacePublicKey, and ExtractPartialPrivateKey, and ExtractFullPrivateKey queries made by A, A' makes the same queries to its oracles and returns the answers it gets from the oracles to A. When A outputs ID^*, m_0 and m_1 for a challenge, A' outputs ID^* to its challenger and gets back (C^*, K'). A' then tosses a random bit d and computes $C_2^* \leftarrow \mathcal{DEM}.\mathsf{Enc}(K', m_d)$. A' then returns $c^* = (C^*, C_2^*)$ to A as the challenge ciphertext. At the end of the simulation, if A outputs a bit d' such that $d' = d$, then A' outputs 0, otherwise, A' outputs 1.

Let b denote the bit tossed by the challenger of A', and b' the bit output by A'. If $b = 0$, which means K' is indeed the key encapsulated in C^*, then the game simulated for A is identical to the Type-X-cpa game. So we have

$$\Pr[b' = 0|b = 0] = \Pr[d' = d|b = 0] = 1/2(\mathbf{Adv}^{\text{Type-X-cpa}}_{\mathcal{CLE},A}(k) + 1).$$

If $b = 1$, then K' is a random key, so A is in the Type'-X-cpa game. So we have

$$\Pr[b' = 0|b = 1] = \Pr[d' = d|b = 1] = 1/2(\mathbf{Adv}^{\text{Type'-X-cpa}}_{\mathcal{CLE},A}(k) + 1).$$

Then we have

$$\begin{aligned}\mathbf{Adv}^{\text{Type-X-cpa}}_{\mathcal{CL}\text{-}\mathcal{KEM},A'}(k) &= 2(\Pr[b' = 0|b = 0]\Pr[b = 0] + \Pr[b' = 1|b = 1]\Pr[b = 1]) - 1\\ &= (\Pr[b' = 0|b = 0] + 1 - \Pr[b' = 0|b = 1]) - 1\\ &= \Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]\\ &= 1/2(\mathbf{Adv}^{\text{Type-X-cpa}}_{\mathcal{CLE},A}(k) - \mathbf{Adv}^{\text{Type'-X-cpa}}_{\mathcal{CLE},A}(k))\end{aligned}$$

and

$$\mathbf{Adv}^{\text{Type-X-cpa}}_{\mathcal{CLE},A}(k) = \mathbf{Adv}^{\text{Type'-X-cpa}}_{\mathcal{CLE},A}(k) + 2\mathbf{Adv}^{\text{Type-X-cpa}}_{\mathcal{CL}\text{-}\mathcal{KEM},A'}(k)$$

Lemma 1. $\mathbf{Adv}^{\textit{Type'-X-cpa}}_{\mathcal{CLE},A}(k)$ *is negligible.*

In the next we show that A only has a negligible advantage in the Type'-X-cpa game, otherwise we can construct another adversary B that breaks the underlying DEM.

B simulates the Type'-X-cpa game for A by performs all the operations honestly. When A outputs ID^*, m_0 and m_1 for a challenge, B outputs m_0 and m_1 in the find-then-guess game and gets back a challenge C^*. B then runs $\mathcal{CL}\text{-}\mathcal{KEM}.\mathsf{Encap}(mpk, ID^*, pk_{ID^*})$ to get K, C_1^*. B discards K and sets $c^* = (C_1^*, C^*)$ as the challenge ciphertext for A, and continues the game for A. Finally B outputs whatever A outputs. We can see that B simulates the Type'-X-cpa game perfectly, so we have

$$\mathbf{Adv}^{\text{fg}}_{\mathcal{DEM},B}(k) = \mathbf{Adv}^{\text{Type'-X-cpa}}_{\mathcal{CLE},A}(k).$$

Combining all the results, we have

$$\mathbf{Adv}^{\text{Type-X-cpa}}_{\mathcal{CLE},A}(k) = \mathbf{Adv}^{\text{fg}}_{\mathcal{DEM},B}(k) + 2\mathbf{Adv}^{\text{Type-X-cpa}}_{\mathcal{CL}\text{-}\mathcal{KEM},A'}(k). \quad \square$$

Constructing CL-KEMs from IB-KA protocols. We reconsider the FGS transformation in this section, and show that we can construct CPA secure CL-KEMs from IB-KA protocols that are only required to be secure in a model that is weaker than the existing IB-KA security models. So we can apply our transformation directly to existing proven secure IB-KA protocols without modifying or re-proving them.

Weak IB-KA Security Model. The weak IB-KA model is similar to the basic IB-KA security model described in Section 3.3 except that in the weak security model the adversary is only allowed to make **NewSession**, **Send,Corrupt** and **Test** queries.

Theorem 9. *If IB-KA is a secure two-pass one-way identity-based key agreement protocol in the weak IB-KA security model, then the corresponding CL-KEM obtained via the FGS transform is Type-I*-cpa secure.*

The proof of Theorem 9 essentially follows that of Theorem 2 except that we don't need to handle the decapsulation queries in the CPA game. In the proof of Theorem 2, an IB-KA adversary B is constructed such that B simulates the Type-I* game for an CL-KEM adversary A and if A can break the CL-KEM then B can break the underlying IB-KA protocol. B makes use of the Reveal and Reveal* queries to answer the decapsulation queries made by A. But in the CPA game, no decapsulation queries are made by A, as a result B does not need to make any Reveal or Reveal* queries in order to simulate the Type-I*-cpa game for A. Besides this difference, the remaining of the proof is the same as that of Theorem 2.

Theorem 10. *If IB-KA is a two-pass one-way identity-based key agreement protocol that achieves master-key forward secrecy in the weak IB-KA security model, then the corresponding CL-KEM obtained via the FGS transform is Type-II-cpa secure.*

The proof of Theorem 10 follows that of Theorem 3 with the same modifications we made above when proving Theorem 9.

Theorem 11. *If a CL-KEM is Type-I*-cpa and Type-II-cpa secure, then it is Type-I-cpa secure.*

The proof of Theorem 11 follows the same proof as that of Theorem 1.

Combing all together. Now we are ready to present a generic construction of CLE schemes from IB-KA protocols. Given an IB-KA protocol $\mathcal{P}$ that is secure in the weak model, and a fg-secure DEM scheme $\mathcal{DEM}$ (e.g., the one-time padding encryption), we can obtain a CLE scheme $\mathcal{CLE}$ as follows.

- $\mathcal{CLE}$.Setup, $\mathcal{CLE}$.ExtractPartialKey, $\mathcal{CLE}$.SetSecretValue, $\mathcal{CLE}$.SetPublicKey, $\mathcal{CLE}$.SetPrivateKey: the same as in the FGS transformation.
- $\mathcal{CLE}$.Enc(mpk, ID, pk_{ID}, m):
  1. randomly choose $\delta \in \{0,1\}^k$, compute $r \leftarrow H(m, \delta, pk_{ID}, ID)$;
  2. compute $(epk, esk) \leftarrow \mathcal{P}.\mathsf{Response}(mpk; r)$;
  3. compute $K \leftarrow \mathcal{P}.\mathsf{Derive}_R(mpk, esk, pk_{ID}, ID)$, $C_1 \leftarrow epk$, and $C_2 \leftarrow \mathcal{DEM}.\mathsf{Enc}(K, m\|\delta)$;
  4. return $c = (C_1, C_2)$.
- $\mathcal{CLE}$.Dec(mpk, $sk_{ID} = (D_{ID}, S_{ID})$, c):
  1. parse c into (C_1, C_2);
  2. compute $K \leftarrow \mathsf{Derive}_I(mpk, D_{ID}, S_{ID}, C_1)$;
  3. compute $m\|\delta \leftarrow \mathcal{DEM}.\mathsf{Dec}(K, C_2)$;
  4. compute $r' \leftarrow H(m, \delta, pk_{ID}, ID)$, $(epk', esk') \leftarrow \mathcal{P}.\mathsf{Response}(mpk; r')$;
  5. compute $K' \leftarrow \mathcal{P}.\mathsf{Derive}_R(mpk, esk', pk_{ID}, ID)$, $C_1' \leftarrow epk'$, and $C_2' \leftarrow \mathcal{DEM}.\mathsf{Enc}(K', m\|\delta)$;
  6. if $(C_1, C_2) = (C_1', C_2')$, return m, otherwise, return $\perp$.

Combing Theorems 7–11, we have

Theorem 12. *The CLE scheme constructed as above is Strong Type-I and Strong Type-II secure in the random oracle model if the IB-KA protocol is secure in the weak model and the MKFS weak model, and the DEM is fg-secure.*

7.2. A new pairing-free CLE scheme

We can now construct a new pairing-free CLE scheme (denoted by FG-2 CLE) directly from the Fiore–Gennaro IB-KA protocol [11], and from the proven security of the FG IB-KA protocol, we directly obtain the provable security of the CLE scheme.

- $\mathsf{Setup}(1^k)$: $x \leftarrow \mathbb{Z}_q$, $y \leftarrow g^x$, $msk \leftarrow x$, $mpk \leftarrow y$.
- $\mathsf{ExtractPartialKey}(msk, ID)$: $\alpha \leftarrow \mathbb{Z}_q$, $r_{ID} \leftarrow g^\alpha$, $z_{ID} \leftarrow \alpha + H_1(ID\|r_{ID})x$, $D_{ID} \leftarrow (r_{ID}, z_{ID})$.
- $\mathsf{SetSecretValue}(mpk, ID)$: $S_{ID} \leftarrow \mathbb{Z}_q$.
- $\mathsf{SetPublicKey}(mpk, S_{ID}, D_{ID})$: $U_{ID} \leftarrow g^{S_{ID}}$, $pk_{ID} \leftarrow (r_{ID}, U_{ID})$.
- $\mathsf{SetPrivateKey}$: $sk_{ID} \leftarrow (D_{ID}, S_{ID})$.
- $\mathsf{Enc}(mpk, pk_{ID}, ID, m)$:
  1. randomly choose $\delta \in \{0,1\}^k$, compute $r \leftarrow H(m, \delta, pk_{ID}, ID)$, $C_1 \leftarrow g^r$;
  2. compute $\lambda_1 \leftarrow (U_{ID} \cdot r_{ID} \cdot mpk^{H_1(ID\|r_{ID})})^r$, $\lambda_2 \leftarrow U_{ID}^r$;
  3. compute $K \leftarrow H_2(\lambda_1, \lambda_2)$, $C_2 \leftarrow K \oplus m\|\delta$.
  4. return $c = (C_1, C_2)$.
- $\mathsf{Dec}(mpk, sk_{ID}, c)$:
  1. parse c into (C_1, C_2);
  2. compute $\lambda_1 \leftarrow C_1^{S_{ID}+z_{ID}}$, $\lambda_2 \leftarrow C_1^{S_{ID}}$, $K \leftarrow H_2(\lambda_1, \lambda_2)$;
  3. compute $m\|\delta \leftarrow \mathcal{DEM}.\mathsf{Dec}(K, C_2)$;
  4. compute $r' \leftarrow H(m, \delta, pk_{ID}, ID)$, $C_1' \leftarrow g^{r'}$;
  5. compute $\lambda_1' \leftarrow (U_{ID} \cdot r_{ID} \cdot mpk^{H_1(ID\|r_{ID})})^{r'}$, $\lambda_2' \leftarrow U_{ID}^{r'}$;
  6. compute $K' \leftarrow H_2(\lambda_1', \lambda_2')$, $C_2' \leftarrow K' \oplus m\|\delta$.
  7. if $(C_1, C_2) = (C_1', C_2')$, return m, otherwise, return $\perp$.

Corollary 3. *The FG-2 CLE is Strong Type-I and Strong Type-II secure in the random oracle model assuming the SDH problem is hard.*

The corollary is obtained from Theorem 12 in Section 7.1, and Theorems 3 and 4 of [11].

8. Comparisons among pairing-free encryption schemes

We compare our new pairing-free schemes with existing pairing-free CL-KEM/CLE schemes in Table 1. Among all the proven secure schemes, FG-1 achieves the best efficiency comparing to FG-2 and Sun et al.'s scheme, while the FG-2 scheme is more efficient in encryption but less efficient in decryption than Sun et al.'s scheme.

Table 1
Comparisons among pairing-free CL-KEM/CLE schemes.

CL-KEM/CLE	Strong type-I	Strong type-II	Enc cost	Dec cost
Beak et al. [4]	?	$\surd$	2 exp + 1 m-exp	3 exp
Sun et al. [19]	$\surd$	$\surd$	4 exp + 1 m-exp	3 exp
Fiore et al. [10]	$\times$	$\times$	2 exp + 1 m-exp	2 exp
FG-1	$\surd$	$\surd$	2 exp + 1 m-exp	2 exp
FG-2	$\surd$	$\surd$	2 exp + 1 m-exp	4 exp + 1 m-exp

$\surd$: Proven Secure; $\times$: Insecure; ?: unknown; exp: exponentiation; m-exp: multi-exponentiation ($\approx$1.5 exp).

9. Conclusions and future work

In this paper, we revisited the Fiore–Gennaro–Smart (FGS) transformation which converts identity-based key agreement (IB-KA) protocols to certificateless key encapsulation Mechanisms (CL-KEMs), and a pairing-free CL-KEM based on the Fiore–Gennaro (FG) IB-KA protocol. We presented an attack against the pairing-free CL-KEM transformed from the FG IB-KA protocol, and a proven secure variant of it based on a modified FG IB-KA protocol. Our attack shows one drawback of the FGS transformation: existing proven secure IB-KA protocols must be re-proved in a very strong new model before applying the transformation. Motivated by this, we presented a new generic construction of certificateless encryption schemes from IB-KA protocols in the random oracle. Our new generic construction is more convenient to use than the FGS transformation since we can directly apply our transformation to existing proven secure IB-KA protocols. An interesting open problem is to develop a new convenient and generic construction of certificateless encryption schemes from IB-KA protocols without the random oracle assumption.

References

[1] Sattam S. Al-Riyami, Cryptographic schemes based on elliptic curve pairings, Ph.D. Thesis, University of London, 2004.
[2] Sattam S. Al-Riyami, Kenneth G. Paterson, Certificateless public key cryptography, in: Advances in Cryptology - ASIACRYPT 2003, pp. 452–473.
[3] Joonsang Baek, Important note on certificateless public key encryption without pairing, 2007. http://www1.i2r.a-star.edu.sg/~jsbaek/publications/note.pdf.
[4] Joonsang Baek, Reihaneh Safavi-Naini, Willy Susilo, Certificateless public key encryption without pairing, in: Information Security Conference, ISC 2005, pp. 134–148.
[5] Mihir Bellare, Anand Desai, E. Jokipii, Phillip Rogaway, A concrete security treatment of symmetric encryption, in: Foundations of Computer Science — FOCS, 1997, pp. 394–403.
[6] Kamel Bentahar, Pooya Farshim, John Malone-Lee, Nigel P. Smart, Generic constructions of identity-based and certificateless KEMs, J. Cryptology 21 (2) (2008) 178–199.
[7] Colin Boyd, Yvonne Cliff, Juan Gonzalez Nieto, Kenneth G. Paterson, Efficient one-round key exchange in the standard model, in: Proc. ACISP 2008, pp. 69–83.
[8] Alexander W. Dent, A survey of certificateless encryption schemes and security models, Int. J. Inf. Sec. 7 (5) (2008) 349–377.
[9] Alexander W. Dent, Benoît Libert, Kenneth G. Paterson, Certificateless encryption schemes strongly secure in the standard model, in: Public Key Cryptography 2008, pp. 344–359.
[10] D. Fiore, R. Gennaro, N.P. Smart, Constructing certificateless encryption and id-based encryption from id-based key agreement, Cryptology ePrint Archive, Report 2009/600, 2009. http://eprint.iacr.org/.
[11] Dario Fiore, Rosario Gennaro, Making the Diffie–Hellman protocol identity-based, in: CT-RSA 2010, pp. 165–178.
[12] Eiichiro Fujisaki, Tatsuaki Okamoto, Secure integration of asymmetric and symmetric encryption schemes, in: Advances in Cryptology—CRYPTO, Springer, 1999, pp. 537–554.
[13] Qiong Huang, Duncan S. Wong, Generic certificateless encryption in the standard model, in: IWSEC 2007, pp. 278–291.
[14] Qiong Huang, Duncan S. Wong, Generic certificateless key encapsulation mechanism, in: ACISP 2007, pp. 215–229.
[15] Hugo Krawczyk, HMQV: a high-performance secure Diffie–Hellman protocol, in: Advances in Cryptology—Crypto 2005, pp. 546–566.
[16] Benoît Libert, Jean-Jacques Quisquater, On constructing certificateless cryptosystems from identity based encryption, in: Public Key Cryptography 2006, pp. 474–490.
[17] Adi Shamir, Identity-based cryptosystems and signature schemes, in: CRYPTO 1984, pp. 47–53.
[18] Victor Shoup, Using hash functions as a hedge against chosen ciphertext attack, in: Advances in Cryptology—EUROCRYPT, 2000, pp. 275–288.
[19] Yinxia Sun, Futai Zhang, Joonsang Baek, Strongly secure certificateless public key encryption without pairing, in: Cryptology and Network Security, CANS, 2007, pp. 194–208.