Constructing Certificateless Encryption and ID-Based Encryption from ID-Based Key Agreement

Dario Fiore[1⋆], Rosario Gennaro[2], and Nigel P. Smart[3]

[1] École Normale Supérieure,
CNRS - INRIA,
Paris, France.
dario.fiore@ens.fr
[2] IBM T.J.Watson Research Center,
Hawthorne, New York,
U.S.A.
rosario@us.ibm.com
[3] Dept. Computer Science,
University of Bristol,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
nigel@cs.bris.ac.uk

Abstract. We discuss the relationship between ID-based key agreement protocols, certificateless encryption and ID-based key encapsulation mechanisms. In particular we show how in some sense ID-based key agreement is a primitive from which all others can be derived. In doing so we focus on distinctions between what we term *pure* ID-based schemes and *non-pure* schemes, in various security models. We present security models for ID-based key agreement which do not "look natural" when considered as analogues of normal key agreement schemes, but which look more natural when considered in terms of the models used in certificateless encryption. Our work highlights distinctions between the two approaches to certificateless encryption, and adds to the debate about what is the "correct" security model for certificateless encryption.

1 Introduction

The notion of certificateless encryption was introduced by Al-Riyami and Paterson [3] and considers the following setting, that is similar to that of identity-based encryption. Each user is represented by a string ID (his identity) and has a matching secret key produced by a Key Generation Center (KGC). Furthermore each user has also a public/secret key pair, as in the traditional public key model. The main advantages of certificateless encryption are that such public keys do not need to be certified and the KGC cannot decrypt ciphertexts of users. In general, the security of certificateless encryption schemes is formalized by two properties related to semantic security of standard encryption schemes: Type I and Type II security. Type I security considers adversaries that are able to replace the public keys of users while Type II security is stated with respect to malicious KGCs.

Ever since its introduction in [3] certificateless encryption has been the subject of debate as to what is the "correct" definition. This is not only a question of the definition of the security model, but also the syntax and functionality of the schemes itself. Many papers have presented differing restrictions for the adversaries in both Type I and Type II security games, creating a lot of different security definitions, with each paper claiming theirs to be the "correct" one. Also other papers have presented new syntax (with similar claims). Most of the claims are actually related to what can be proved about the schemes the papers present, rather than some deeper philosophical discussion. We refer the reader to [15] for a balanced summary of the existing models and schemes.

⋆ Work partially done while at University of Catania, Italy.

1.1 Our contribution.

This paper takes a different approach to the study of certificateless schemes, by studying their relationship to identity-based encryption. We do so in order to take a step back from scheme construction and instead concentrate on what the correct security and syntactic definitions should be. To simplify our discussion we will concentrate on the simpler notion of key-encapsulation (KEM) rather than encryption.

We show two main results: (1) a natural transform of *certain* CL-KEM schemes into ID-KEM schemes. In addition there is (2) another natural transform of *all* identity-based key agreement (ID-KA) protocols into CL-KEM schemes. We note that all our security relationships under our transforms hold in the *standard model*.

The motivation for this research is twofold: (i) by analyzing these transformations we are able to get a better understanding of what are the "correct" security notions and syntaxes for CL encryption; (ii) these reductions may give us a *generic* toolbox to construct new, and potentially improved, CL and ID schemes.

Pure and non-pure schemes. Certificateless schemes in the literature can be syntactically classified into two large classes, which we call *pure* and *non-pure*. This distinction between pure and non-pure schemes also applies to existing ID-KA protocols. Informally, a pure ID-based key agreement (resp. certificateless scheme) is one in which the parties compute their messages *without* using their long-term secret keys (which is used only in the derivation of the shared session key). As we will show, such pure schemes allow various functionalities such as encryption into-the-future etc. Interestingly there are no-known pure schemes (either ID-KA or CL-KEM) which do not use pairing-based groups.

We show a natural *standard model* transformation from a *pure* CL-KEM to a ID-KEM and we determine the precise security properties of the CL-KEM under which the resulting ID-KEM is secure in the usual sense. The hope is that this generic transformation might in the future yield new constructions for ID-based encryption. It is worthwhile to observe that this transform does not work for non-pure CL-KEMs. This is not surprising as non-pure CL-KEMs are the only ones that can be constructed without pairings. So, in some sense this shows that certificateless encryption is a simpler primitive than ID-based encryption, although the reverse is commonly believed (as CL encryption is thought as an extension of ID-based one).

Towards a correct security model for CL-KEMs and ID-KA protocols. Next we show a natural generic transform of ID-KA protocols into CL-KEM schemes. The goal here is to gain some understanding on the correct security models for these notions. In particular we investigate what security models in the ID-KA setting imply, through our transform, certain specific CL-KEM security models. For lack of space, we do not look at all CL-KEM security models, but we do consider the main ones. Our results, all proven in the standard model, can be summarized in two distinct points. First, if one concentrates on pure schemes [12], then the associated transforms have a tight security reduction. This supports our previous point that pure schemes have more/better features. Second, the required security models in the ID-KA setting needed to imply strong notions of security in the CL-KEM setting are highly non-standard security notions for key agreement models. This last point can be interpreted in one of two ways: either the strong security models for CL-KEM schemes are unnatural and that the weaker definitions should suffice, or the security notions for ID-KA protocols (and by implication all other forms of key agreement protocol) are too weak.

At the end of the paper we try to draw some conclusions as to what the "correct" models for certificateless encryption and ID-based key agreement should be. Our conclusion is that perhaps the strong security models for certificateless encryption are probably correct, and that it is the security models for ID-KA protocols, and indeed standard public key or symmetric key based key agreement protocols, which need to be strengthened.

Our main generic constructions can be summarized by reference to Figure 1.1, the definitions used in the arrows will become clear as we define them in the following pages.

As a final side-result of independent interest, as part of our analysis we consider a weakened notion of Type-I security for certificateless schemes (which we denote by Type-I* etc). This is because we have discovered an overlap in the standard security definitions for Strong Type-I and Strong Type-II security for CL-KEMs. By weakening the definition of Type-I security slightly, we remove this overlap and at the same time simplify a number of our security proofs, whilst not reducing the overall security result for the resulting CL-KEMs.

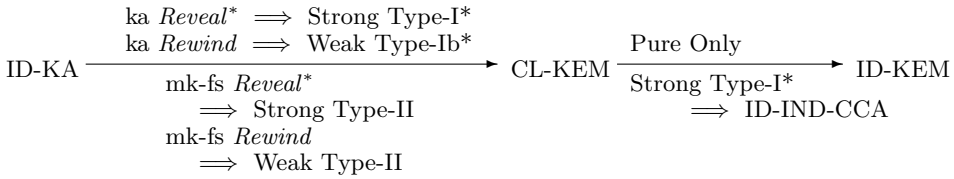

Fig. 1. Relationships Between Schemes

Other related work. Our results are similar to the work of Paterson and Srinivasan [18] on the link between ID-based non-interactice key distribution (NIKD) and ID-based encrytion. In [18] the authors present a security model for ID-based NIKD and provide a transform from an ID-based NIKD to an ID-based encryption scheme. We note that the extension of this result to constructing ID-KEMs is immediate. However, this transform is not generic in that it requires special syntactic properties of the base ID-based NIKD scheme. Our transforms from ID-KA protocols (i.e. interactive protocols) to CL-KEMs and ID-KEMs are generic and do not require any special syntactic properties of the underlying ID-KA protocol.

In addition the transform of [18] results in ID-IND-CPA ID-KEMs/ID-based encryption schemes. Indeed to obtain full CCA secure KEMs/encryption schemes it is easy to see that one needs to extend the security model in [18] for ID-based NIKD schemes in such a way as to provide the adversary with an analogue of our *Reveal** oracles. Thus whilst our results are syntactically stronger than those of [18], the security results are roughly equivalent. That we can achieve more syntactically is due to us considering interactive, as opposed to non-interactive, protocols as our starting point.

1.2 Notation

We end this introduction by recapping on the following standard notational conventions which will be used throughout this paper. We denote by $\leftarrow$ the assigment operator, i.e., $x \leftarrow y$ means that the variable x is assigned the value y. With $x \leftarrow S$ where S is a finite set, we denote the process to assign to x a randomly and uniformly chosen value in S. If A is an algorithm then $x \leftarrow A$ means assign x the output value of algorithm A, with associated probability distribution determined by the random coins of A.

2 Identity-Based Key Agreement

In this section we present the notion of ID-based key agreement. We will only consider two pass ID-based key agreement protocols in this paper as this simplifies the algorithm descriptions somewhat.

2.1 ID-Based Key Agreement Definition

A two-pass ID-based key agreement protocol is specified by six algorithms which run in polynomial time in the security parameter. The two passes are illustrated in Figure 2. We let $\mathcal{ID}$ denote the set of possible user identities and $\mathbb{K}_{\mathtt{KA}}(mpk^{\mathtt{KA}})$ be the set of valid session keys for the public parameter $mpk^{\mathtt{KA}}$.

- $\mathsf{KASetup}(1^t)$ is a PPT algorithm that takes as input the security parameter 1^t and returns the master public key $mpk^{\mathtt{KA}}$ and the master secret key $msk^{\mathtt{KA}}$.
- $\mathsf{KeyDer}(msk^{\mathtt{KA}}, ID)$ is the private key extraction algorithm. It takes as input $msk^{\mathtt{KA}}$ and $ID \in \mathcal{ID}$ and it returns the associated private key d_{ID}. This algorithm may be deterministic or probabilistic.
- $\mathsf{Initiate}(mpk^{\mathtt{KA}}, d_I)$. This is a PPT algorithm run by the initiator, with identity I, of the key agreement protocol which produces the ephemeral public key epk_I for transmission to another party. The algorithm stores esk_I, the corresponding ephemeral private key, for use later[4]

[4] Notice that we refer to the messages exchanged by the parties as *public keys*, and their secret states after the computation of the message as *secret keys*. Jumping ahead, this is because that's the role these values play in our transformation from KA to CL scheme.

– $\mathsf{Respond}(mpk^{\mathtt{KA}}, d_R)$. This is a PPT algorithm run by the responder, with identity R, of the key agreement protocol which produces the ephemeral public/private key (epk_R, esk_R).
– $\mathsf{Derive}_I(mpk^{\mathtt{KA}}, d_I, esk_I, epk_R, R)$. This is a (possibly probabilistic) algorithm run by the initiator to derive the session key $K_I \in \mathbb{K}_{\mathtt{KA}}(mpk^{\mathtt{KA}})$ with party R.
– $\mathsf{Derive}_R(mpk^{\mathtt{KA}}, d_R, esk_R, epk_I, I)$. This is a (possibly probabilistic) algorithm run by the responder to derive the session key $K_R \in \mathbb{K}_{\mathtt{KA}}(mpk^{\mathtt{KA}})$ with party I.

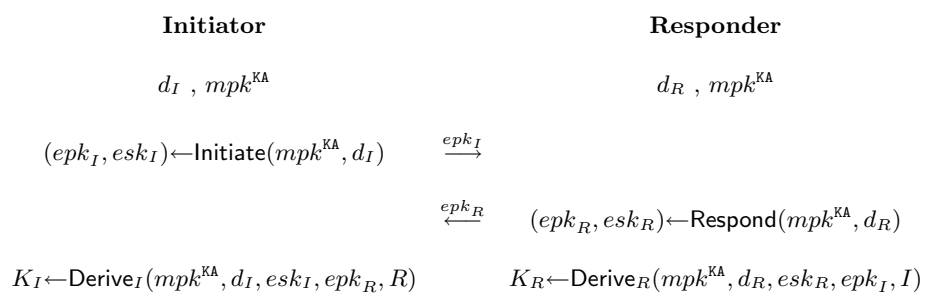

Fig. 2. Diagrammatic view of two-pass ID-KA protocols

For correctness we require that in a valid run of the protocol we have that $K_I = K_R$. Notice, that the creation of the ephemeral public/private key pairs does not depend on the intended recipient. Most ID-KA protocols are of this form. For example in [12] ID-based key agreement protocols based on pairings are divided into four Categories. Only in Categories 2 and 4 does the emphemeral key pair depend on the intended recipient, these being protocols in the Scott [19] and McCullagh–Barreto [17] families. The majority of pairing-based ID-based key agreement protocols lie in the Smart [21] family (denoted Category 1 in [12]), with Category 3 (the Chen–Kudla family [13]) also sharing this property. The non-pairing based protocol of Fiore and Gennaro [16] also has this property.

If the algorithms $\mathsf{Initiate}$ and $\mathsf{Respond}$ do not require access to d_I and d_R respectively, then we call the protocol a *pure* identity based key agreement protocol. This is because the ephemeral public keys can be created *before* the sender knows his long term secret key. This therefore allows forms of sending-into-the-future which are common in many IBE style schemes. We shall return to this distinction below when discussing the conversion of ID-KA protocols into certificateless schemes. Indeed identifying differences between these two forms of ID-KA protocols and certificateless schemes, forms a significant portion of the current paper. In the categorization of [12] Categories 1, 3 and 4 are all pure ID-based key agreement protocols, whilst Category 2 and the non-pairing based FG protocol are non-pure.

A key agreement protocol is said to be role symmetric if algorithm $\mathsf{Initiate}$ is identical to algorithm $\mathsf{Respond}$ and algorithm Derive_I is identical to algorithm Derive_R. The FG protocol is role symmetric, but role symmetry is a more complex property to determine for pairing-based protocols. For example whether a scheme is role symmetric can depend on whether one instantiates the protocol with symmetric or asymmetric pairings. For the schemes in [12] (and focusing solely on the more practical scenario of asymmetric pairings) all those in Categories 2 and 4 are role symmetric, those in Category 3 are not, whereas half of those in Category 1 are. Of particular importance in Category 1 is the SCK protocols (which are a combined version of the Smart and Chen–Kudla protocol), these are highly efficient and role symmetric.

2.2 Defining Security for ID-Based Key Agreement

We will be using a modified version of the Bellare–Rogaway key exchange model, as extended to an identity based setting. Our model is an extension of the model contained in Chen *et al.* [12], but we extend it in various ways which we will describe later. So as to be precise we describe the model in more formal detail than that used in [12], however we shall (as stated above) be focusing solely on two-pass protocols, which explains some of our specifications in what follows.

Security of a protocol is defined by a game between an adversary A and a challenger E. At the start of the game the adversary A is passed the master public key $mpk^{\mathtt{KA}}$ of the key generation centre. During the game the adversary is given access to various oracles $\mathcal{O}$ which maintain various meta-variables, including

- $role_{\mathcal{O}} \in \{initiator, responder, \perp\}$. This records the type of session to which the oracle responds.
- $pid_{\mathcal{O}} \in \mathcal{U}$. This keeps track of the intended partner of the session maintained by $\mathcal{O}$.
- $\delta_{\mathcal{O}} \in \{\perp, accepted, error\}$. This determines whether the session is in a finished state or not.
- $\gamma_{\mathcal{O}} \in \{\perp, corrupted, revealed\}$. This signals whether the oracle has been corrupted or not.
- $s_{\mathcal{O}}$. This denotes the session key of the protocol if the protocol has completed.

The adversary can execute a number of oracle queries which we now describe.

- $NewSession(U, V)$ This creates a new oracle, to represent the new session, which we shall denote by $\mathcal{O} = \Pi^i_{U,V}$, where i denotes this is the ith session for the user with identity U, and that the indented partner is V. After calling this oracle we have

$$pid_{\mathcal{O}} = V \text{ and } s_{\mathcal{O}} = role_{\mathcal{O}} = \delta_{\mathcal{O}} = \gamma_{\mathcal{O}} = \perp .$$

  However, if any other oracle with identity U has been corrupted then we set $\gamma_{\mathcal{O}} = corrupted$.
- $Send(\mathcal{O}, role, msg)$. Recall we are only modelling two-pass protocols, hence the functionality of this oracle can be described as follows:
  - If $\delta_{\mathcal{O}} \neq \perp$ then do nothing.
  - If $role = initiator$ then
    - If $msg = \perp$, $\delta_{\mathcal{O}} = \perp$ and $role_{\mathcal{O}} = \perp$ then set $role_{\mathcal{O}} = initiator$ and output a message (i.e. send the first message flow in the protocol);
    - If $msg \neq \perp$ and $role_{\mathcal{O}} = initiator$ (i.e. msg is the second message flow in the protocol) then compute $s_{\mathcal{O}}$ and set $\delta_{\mathcal{O}} = accepted$;
    - Else set $\delta_{\mathcal{O}} = error$ and return $\perp$
  - If $role = responder$ then
    - If $msg \neq \perp$ and $role_{\mathcal{O}} = \perp$ then compute $s_{\mathcal{O}}$, set $\delta_{\mathcal{O}} = accepted$, $role_{\mathcal{O}} = responder$ and respond with a message (i.e. send the second message flow in the protocol).
    - Else set $\delta_{\mathcal{O}} = error$ and return $\perp$.
- $Reveal(\mathcal{O})$. If $\delta_{\mathcal{O}} \neq accepted$ or $\gamma_{\mathcal{O}} = corrupted$ then this returns $\perp$, otherwise it returns $s_{\mathcal{O}}$ and we set $\gamma_{\mathcal{O}} = revealed$.
- $Corrupt(U)$. This returns d_U and sets all oracles $\mathcal{O}$ in the game (both now and in the future) belonging to party U to have $\gamma_{\mathcal{O}} = corrupted$. Notice, that this is equivalent to the extract secret key query in security games for other types of identity based primitives. Note, that we do not assume that the rest of the internal state of the oracles belonging to U are turned over to the adversary.
- $Test(\mathcal{O}^*)$. This oracle may only be called once by the adversary during the game. It takes as input a *fresh oracle* (see below for the definition of freshness). The challenger E then selects a bit $b \in \{0, 1\}$. If $b = 0$ then the challenger responds with the value of $s_{\mathcal{O}^*}$, otherwise it responds with a random key chosen from the space of session keys. We call the oracle on which *Test* is called the Test-oracle.

At the end of the game the adversary will output its guess b' as to the bit b used by the challenger in the *Test* query. We define the advantage of the adversary by

$$\mathrm{Adv}_{ID-\mathtt{KA}}(A) = |2\Pr[b' = b] - 1| .$$

We now explain the $Test(\mathcal{O}^*)$ query in more detail. An oracle $\mathcal{O}^* = \Pi^i_{U^*,V^*}$ is said to be *fresh* if

1. $\delta_{\mathcal{O}^*} = accepted$.
2. $\gamma^*_{\mathcal{O}} \neq revealed$.
3. Party V^* is not corrupted.
4. There is no oracle $\mathcal{O}'$ with $\gamma_{\mathcal{O}'} = revealed$ with which $\mathcal{O}^*$ has had a matching conversation.

After the $Test(\mathcal{O}^*)$ query has been made the adversary can continue making queries as before, except that it cannot:

- corrupt party V^*,
- call a reveal query on $\mathcal{O}^*$'s partner oracle if it exists,
- call reveal on $\mathcal{O}^*$.

Definition 1 *A protocol Π is said to be a secure ID-KA protocol (or more simply ka secure) if*

1. *In the presence of a benign adversary, which faithfully conveys messages, on $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$, both oracles always accept holding the same session key, and this key is distributed uniformly on $\{0,1\}^k$;*
2. *For any polynomial time adversary A, $\mathrm{Adv}_{ID-\mathtt{KA}}(A)$ is negligible.*

Forward Secrecy. We also define a notion of *master-key forward secrecy*, (or mk-fs secure) following [12]. In this model the adversary is also given the master secret key $msk^{\mathtt{KA}}$. Thus the adversary can compute the private key d_{ID} of any party. The security game is the same as above, except that instead of a fresh oracle for the test session it chooses an oracle $\mathcal{O}^*$ which satisfies:

1. $\delta_{\mathcal{O}^*} = accepted$
2. $\gamma_{\mathcal{O}^*} \neq revealed$
3. There is an oracle $\mathcal{O}'$ with which $\mathcal{O}^*$ has had a matching conversation and $\delta_{\mathcal{O}'} = accepted$ and $\gamma_{\mathcal{O}'} \neq revealed$.

Weaker notions of forward-secrecy are implied by the above, for example *perfect forward secrecy* gives the adversary access to a *Corrupt* oracle for any $ID \in \mathcal{ID}$ but does not give the adversary access to $msk^{\mathtt{KA}}$. A weaker form of simply *forward secrecy* is then implied where the adversary can only call the *Corrupt* oracle on one party in the test session, i.e. we must have either $\gamma_{\mathcal{O}^*} = \perp$ or $\gamma_{\mathcal{O}'} = \perp$.

The advantage for forward secrecy of an adversary is defined in the same way as above and is denoted by one of

$$\mathrm{Adv}^{mk-fs}_{ID-\mathtt{KA}}(A), \quad \mathrm{Adv}^{p-fs}_{ID-\mathtt{KA}}(A) \text{ or } \mathrm{Adv}^{fs}_{ID-\mathtt{KA}}(A),$$

as appropriate.

For non-pure ID-based key agreement protocols we can consider an additional notion of forward secrecy, which we call *active* perfect forward secrecy (resp. *active* forward secrecy). In this model we drop the third condition above that there exists another oracle $\mathcal{O}'$ with which $\mathcal{O}^*$ has had a matching conversation. This means that the adversary could have been active before corrupting the parties, i.e. he sent one of the two message flows.

It is interesting to observe that such notion cannot be achieved by any pure ID-based KA protocol because of the following attack. Assume the adversary acts as initiator and computes $epk_I \leftarrow \mathsf{Initiate}(mpk^{\mathtt{KA}})$ (he can do that without d_I as the protocol is pure). He can initiate a new session oracle setting epk_I as first message, then ask for the second message and later make a test query on this oracle. When the adversary corrupts I then he will have all the informations needed to compute the correct session key and so he will be able to distinguish wether he received the real session key or a random one. It is easy to see that such attack does not apply to the case of non-pure protocols as the private key is needed to produce protocol's messages.

Our augmented security model. In our analysis of converting ID-based key agreement protocols into certificateless schemes we will require stronger security notions in which the adversary will have access to additional oracles. We define three such oracles, the first one is relatively standard, whilst the second two are new. The second can be motivated by similar arguments one uses to motivate resettable zero-knowledge [10], whilst the third oracle is a natural analogue in the key agreement setting of the strong adversarial powers one gives an adversary for certificateless schemes. One may therefore consider the extreme nature of the third oracle as an additional argument as to why the certificateless strongest security model looks excessive.

- *StateReveal*$(\mathcal{O})$. If $role_{\mathcal{O}} = \perp$ then do nothing. Otherwise return the value of the ephemeral secret key held within the oracle.

– *Rewind*($\mathcal{O}$). If $role_{\mathcal{O}} = initiator$ and $\delta_{\mathcal{O}} = accepted$ then this returns $\mathcal{O}$ to the state it was in before it received its last message, i.e. it sets $\delta_{\mathcal{O}} = s_{\mathcal{O}} = \perp$. If we have $\gamma_{\mathcal{O}} = revealed$ then we also reset $\gamma_{\mathcal{O}}$ to $\perp$.
– *Reveal*$^*(I, R, epk_I, epk_R)$. This is a stronger version of the *Reveal* query in that it is not associated to an oracle, but simply takes the two message flows and returns the associated agreed shared secret assuming these messages had been transmitted between party I and party R. There is an obvious restriction in that the adversary is not allowed to call this oracle on the message flows used in the *Test* query, nor (for role-symmetric protocols) with the message flows used in the *Test* query but with the roles of initiator and responder swapped.

The *StateReveal*($\mathcal{O}$) query corresponds to an adversarial power which can partially corrupt a party, but which does not allow the adversary to obtain the long term secret. This power has been used in numerous works starting with [11], and is often considered to be the main distinction between the CK model and the BR model for key exchange [14].

The presence of the *Rewind*($\mathcal{O}$) oracle enables the adversary to extract more information for a particular set of ephemeral and static public key pairs. To intuitively see what the *Rewind*($\mathcal{O}$) oracle provides us, imagine a standard key agreement protocol based on standard Diffie–Hellman, for example the Station-to-Station protocol. Usually one reduces the security of this protocol to the decisional Diffie–Hellman problem (DDH). But with the presence of a *Rewind*($\mathcal{O}$) oracle the adversary can take a test oracle (which has output the ephemeral public key g^x) and obtain, using a combination of the *Rewind*($\mathcal{O}$) and *Reveal*($\mathcal{O}$) oracles, values of the form h^x for values of h of the adversaries choosing. This means the simulator is essentially solving the DDH problem with access to a static-Diffie–Hellman oracle.

The *Reveal*$^*(I, R, epk_I, epk_R)$ is a very strong oracle. As we will show later, if a protocol is secure even when an adversary is given such an oracle we are able to transform the protocol into a certificateless encryption scheme which also satisfies a strong security notion.

We say a protocol is a secure ID-KA protocol in the *Rewind*-model (resp. *Reveal**-model) if it is secure as ID-based key agreement protocol where we give the adversary access to a *Rewind* (resp. *Reveal**) oracle. If we require access to two of these oracles we will call the model, for instance, the (*StateReveal*, *Rewind*)-model, We call these extra models, augmented models, since they augment the standard security model with extra functionality. Similarly we can define augmented notions for master-key forward secrecy.

3 From Mutual to One-Way Authentication

In many key agreement protocols one is only interested in one-way authentication. SSL/TLS is a classic example of this, where the server is always authenticated but the user seldom is. We overview in this section the modifications to the previous syntax of ID-KA protocols which are needed to ensure only one-way authentication and show how to convert a mutually authenticated identity-based key agreement protocol into one which is only one-way authenticated. The reason for introducing only one-way authentication is that this enables us to make the jump to certificateless encryption conceptually easier, and can also result in simpler schemes. We assume the responder in a protocol is the one who is *not* authenticated, this is to simplify notation in what follows. The scheme definitions are then rather simple to extend.

We note that any protocol proved to be secure for mutual authentication, can be simplified and remain secure in the context of one-way authentication. The transformation from mutual to one-way authentication is performed as follows. An identity is selected, let us call it R_0, which acts as a "dummy" responder identity. A "dummy" secret key is then created for this user and this is published along with the master public key. Notice, that by carefully selecting the dummy secret key one can often obtain efficiency improvements. The protocol is then defined as before except that R_0 is always used as the responding party, and we drop any reference to d_{R_0}. Thus we call $\mathsf{Respond}(mpk^{\mathtt{KA}})$ rather than $\mathsf{Respond}(mpk^{\mathtt{KA}}, d_{R_0})$. Similarly we call

$$\mathsf{Derive}_R(mpk^{\mathtt{KA}}, esk_{R_0}, epk_{ID}, ID) \text{ and } \mathsf{Derive}_I(mpk^{\mathtt{KA}}, d_{ID}, esk_{ID}, epk_{R_0})$$

rather than

$$\mathsf{Derive}_R(mpk^{\mathtt{KA}}, d_{R_0}, esk_{R_0}, epk_{ID}, ID) \text{ and } \mathsf{Derive}_I(mpk^{\mathtt{KA}}, d_{ID}, esk_{ID}, epk_{R_0}, R_0).$$

In the security model all oracles either have R_0 as an intended partner, or the oracle belongs to R_0. If the oracle belongs to R_0 then it is corrupted, since R_0's secret key is public. This means that only oracles belonging to R_0 may be used in the *Test* queries.

We argue that if the original protocol is secure then its one-way version (obtained as described above) is also one-way secure. To see this observe that an adversary $\mathcal{A}$ that breaks the security of the one-way protocol can be turned into an adversary $\mathcal{B}$ against the original protocol. Assume $\mathcal{A}$ breaks the security choosing a test session that involves a user ID (and the dummy identity R_0). Then $\mathcal{B}$ can trivially choose a test oracle $\Pi^s_{R_0,ID}$ and forward the obtained key to $\mathcal{A}$.

4 Certificateless Key Encapsulation Mechanisms

In this section we discuss various aspects of certificateless KEMs. The reader is referred to [8] and [15] for further details.

4.1 CL-KEM Definition

A CL-KEM scheme is specified by seven polynomial time algorithms:

- $\mathsf{CLSetup}(1^t)$ is a PPT algorithm that takes as input 1^t and returns the master public keys $mpk^{\mathtt{CL}}$ and the master secret key $msk^{\mathtt{CL}}$.
- $\mathsf{Extract\text{-}Partial\text{-}Private\text{-}Key}(msk^{\mathtt{CL}}, ID)$. If $ID \in \mathcal{ID}$ is an identifier string for party ID this (possibly probabilistic) algorithm returns a partial private key d_{ID}.
- Set-Secret-Value is a PPT algorithm that takes no input (bar the system parameters) and outputs a secret value s_{ID}.
- Set-Public-Key is a deterministic algorithm that takes as input s_{ID} and outputs a public key pk_{ID}.
- $\mathsf{Set\text{-}Private\text{-}Key}(d_{ID}, s_{ID})$ is a deterministic algorithm that returns sk_{ID} the (full) private key.
- $\mathsf{Enc}(mpk^{\mathtt{CL}}, pk_{ID}, ID)$ is the PPT encapsulation algorithm. On input of pk_{ID}, ID and $mpk^{\mathtt{CL}}$ this outputs a pair (C, K) where $K \in \mathbb{K}_{\mathtt{CL-KEM}}(mpk^{\mathtt{CL}})$ is a key for the associated DEM and $C \in \mathbb{C}_{\mathtt{CL-KEM}}(mpk^{\mathtt{CL}})$ is the encapsulation of that key.
- $\mathsf{Dec}(mpk^{\mathtt{CL}}, sk_{ID}, C)$ is the deterministic decapsulation algorithm. On input of C and sk_{ID} this outputs the corresponding K or a failure symbol $\perp$.

Baek *et al.* gave in [5] a different formulation where the Set-Public-Key algorithm takes the partial private key d_{ID} as an additional input. In this case it is possible to combine the Set-Secret-Value, Set-Public-Key and Set-Private-Key algorithms into a single Set-User-Keys algorithm that given as input the partial private key d_{ID} of ID outputs pk_{ID} and sk_{ID}. While the Baek *et al.* formulation may seem at first glance to be a simplification, it stops various possible applications of certificateless encryption, such as encrypting into the future. Extending our definition of *pure* and *non-pure* ID-based key agreement protocols to this situation, we shall call certificateless schemes which follow the original formulation as *pure*, and those which follow the formulation of Baek *et al.* as *non-pure*.

4.2 CL-KEM Security Model

To define the security model for CL-KEMs we simply adapt the security model of Al-Riyami and Paterson [3] into the KEM framework, as explained in [8]. The main issue with certificateless encryption is that, since public keys lack authenticating information, an adversary may be able to replace users' public keys with public keys of its choice. This appears to give adversaries enormous power. However, the crucial part of the certificateless framework is that to compute the full private key of a user, knowledge of the partial private key is necessary.

To capture the scenario above, Al-Riyami and Paterson [2–4] consider a security model in which an adversary is able to adaptively replace users' public keys with (valid) public keys of its choice. Such an adversary is called a Type-I adversary below.

Since the KGC is able to produce partial private keys, we must of course assume that the KGC does not replace users public keys itself. By assuming that a KGC does not replace users public keys itself, a user is placing a similar level of trust in a KGC that it would in a PKI certificate authority: it is always assumed that a CA does not issue certificates for individuals on public keys which it has maliciously generated itself! We do however treat other adversarial behaviour of a KGC: eavesdropping on ciphertexts and making decryption queries for example. Such an adversarial KGC is referred to as a Type-II adversary below.

Below we present a game to formally define what an adversary must do to break a certificateless KEM [8]. This is a game run between a challenger and a two stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Note that X can be instantiated with I or II in the description below and that the master secret $msk^{\mathtt{CL}}$ is only passed to the adversary in the case of Type-II adversaries.

Type-X Adversarial Game
1. $(mpk^{\mathtt{CL}}, msk^{\mathtt{CL}}) \leftarrow \mathsf{CLSetup}(1^t)$.
2. $(ID^*, s) \leftarrow \mathcal{A}_1^{\mathcal{O}}(mpk^{\mathtt{CL}}, msk^{\mathtt{CL}})$.
3. $(K_0, C^*) \leftarrow \mathsf{Enc}(mpk^{\mathtt{CL}}, pk^*, ID^*)$.
4. $K_1 \leftarrow \mathbb{K}_{\mathtt{CL-KEM}}(mpk^{\mathtt{CL}})$.
5. $b \leftarrow \{0,1\}$.
6. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(C^*, s, ID^*, K_b)$.

When performing the encapsulation, in line three of both games, the challenger uses the *current* public key pk^* of the entity with identifier ID^*. The adversary's advantage in such a game is defined to be

$$\mathrm{Adv}^{\mathtt{Type-X}}_{\mathtt{CL-KEM}}(\mathcal{A}) = |2\Pr[b' = b] - 1|$$

where X is either I or II. A CL-KEM is considered to be secure, in the sense of IND-CCA2, if for all PPT adversaries $\mathcal{A}$, the advantage in both the games is a negligible function of t.

The crucial point of the definition above is to specify which oracles the adversary is given access and which are the restrictions of the game. According to such specifications one can obtain different levels of security. A detailed discussion about all possible security definitions is given by Dent in [15]. In the following we describe the various oracles $\mathcal{O}$ available to the adversaries, we then describe which oracles are available in which game and any restrictions on these oracles.

- **Request Public Key:** Given an ID this returns to the adversary a value for pk_{ID}.
- **Replace Public Key:** This allows the adversary to replace user ID's public key with any (valid) public key of the adversaries choosing.
- **Extract Partial Private Key:** Given an ID this returns the partial private key d_{ID}.
- **Extract Full Private Key:** Given an ID this returns the full private key sk_{ID}.
- **Strong Decap:** Given an encapsulation C and an identity ID, this returns the encapsulated key. If the adversary has replaced the public key of ID, then this is performed using the secret key corresponding to the new public key. Note, this secret key may not be known to either the challenger or the adversary, hence this is a very strong oracle.
- **Weak SV Decap:** This takes as input an encapsulation C, an identity ID and a secret value s_{ID}. The challenger uses s_{ID} to produce the corresponding full secret key of ID that is used to decapsulate C. Note, that s_{ID} may not correspond to the actual current public key of entity ID. Also note that one can obtain this functionality using the Strong Decap oracle when the certificateless scheme is pure.
- **Decap**: On input of an encapsulation C and an identity ID it outputs the session key obtained decapsulating C with the original secret key created by ID. One can obtain this functionality using a Strong Decap oracle if the scheme is pure.

Using these oracles we can now define the following security models for certificateless KEMs, see [15] for a full discussion.

Strong Type-I Security: This adversary has the following restrictions to its access to the various oracles.

- $\mathcal{A}$ cannot extract the full private key for ID^*.
- $\mathcal{A}$ cannot extract the full private key of any identity for which it has replaced the public key.
- $\mathcal{A}$ cannot extract the partial private key of ID^* if $\mathcal{A}_1$ replaced the public key (i.e. the public key was replaced before the challenge was issued).
- $\mathcal{A}_2$ cannot query the Strong Decap oracle on the pair (C^*, ID^*) unless ID^*'s public key was replaced after the creation of C^*.
- $\mathcal{A}$ may not query the Weak SV Decap or the Decap oracles (although for pure schemes, one can always simulate these using the Strong Decap oracle).

We note that this security notion is often considered to be incredibly strong, hence often one finds it is weakened in the following manner.

Weak Type-Ia Security: Dent describes in [15] a weaker security definition called *Weak Type-Ia* that was also used in [8]. Weak Type-Ia security does not allow the adversary to make decapsulation queries against identities whose public keys have been replaced. In this case the restrictions on the adversaries oracle access is as follows:

- $\mathcal{A}$ cannot extract the full private key for ID^*.
- $\mathcal{A}$ cannot extract the full private key of any identity for which it has replaced the public key.
- $\mathcal{A}$ cannot extract the partial private key of ID^* if $\mathcal{A}_1$ replaced the public key (i.e. the public key was replaced before the challenge was issued).
- $\mathcal{A}$ may not query the Strong Decap oracle at any time.
- $\mathcal{A}_2$ cannot query the Weak SV Decap oracle on the pair (C^*, ID^*) if the attacker replaced the public key of ID^* before the challenge was issued.
- $\mathcal{A}_2$ cannot query the Decap oracle on the pair (C^*, ID^*) unless the attacker replaced the public key before the challenge was issued.

Though this notion is clearly weaker than Strong Type-I, it still looks reasonable for practical purposes. In fact Strong Type-I gives to the adversary as much power as possible, but it is unclear whether a real adversary can obtain decapsulations in practice from users whose public keys have been replaced by the adversary itself.

We pause to note that there are weaker forms of Type-I security called Weak Type-Ib and Weak Type-Ic security. In Weak Type-Ib security access to the Weak SV Decap oracle is denied to the adversary, whereas in Weak Type-Ic security not only denies access to the Weak SV Decap oracle, but it also denies the ability to the replace public keys entirely. We also can define a CPA like-notion, which we call Weak Type-I-CPA which denies access to all forms of decapsulation oracle (this is a notion which is not used in other papers, but which will be useful when we present our conclusions).

In addition, for each definition of Type-I security we can define a slightly weaker variant denoted by * (e.g. Strong Type-I*) in which the adversary cannot query the partial private key of the target identity ID^* at any point. This weaker variant will simplify somewhat our security theorems. But, it still allows us to obtain a final non-weakened result due to the combination with security theorems for Type-II security, which we define below. Dent [15] presents these different notions in the form of a table, which we present a modified version of below.

	Request Public Key	Replace Public Key	Extract Full Full Private Key	Extract Partial Private Key	Extract Partial Private Key for ID^*	Strong Decap	Weak SV Decap	Decap
Strong Type-I	✔	✔	✔	✔	✔	✔		
Weak Type-Ia	✔	✔	✔	✔	✔		✔	✔
Weak Type-Ib	✔	✔	✔	✔	✔			✔
Weak Type-Ic	✔		✔	✔	✔			✔
Weak Type-I-CPA	✔		✔	✔	✔			
Strong Type-I*	✔	✔	✔	✔		✔		
Weak Type-Ia*	✔	✔	✔	✔			✔	✔
Weak Type-Ib*	✔	✔	✔	✔				✔
Weak Type-Ic*	✔		✔	✔				✔
Weak Type-I-CPA*	✔		✔	✔				

Strong Type-II Security: In the Type-II game the adversary has access to the master secret key $msk^{\mathtt{CL}}$ and so can create partial private keys itself. The strong version of this security model enables the adversary to query the various oracles with the following restrictions:

- $\mathcal{A}$ cannot extract the full private key for ID^*.
- $\mathcal{A}$ cannot extract the full private key of any identity for which it has replaced the public key.
- $\mathcal{A}_1$ cannot output an identity ID^* for which it has replaced the public key.
- $\mathcal{A}$ cannot query the partial private key oracle at all.
- $\mathcal{A}_2$ cannot query the Strong Decap oracle on the pair (C^*, ID^*) unless the public key used to create C^* has been replaced.
- $\mathcal{A}$ may not query the Weak SV Decap or the Decap oracles (although for pure schemes, one can always simulate these using the Strong Decap oracle).

Note, because we assume in this case that the adversary *is* the KGC, the adversary does not have access to the partial private key oracle since all partial private keys are ones which he can compute given $msk^{\mathtt{CL}}$. This applies even in the case where generation of the partial private key from $msk^{\mathtt{CL}}$ and ID is randomised.

Weak Type-II Security: As for the case of Type-I security one can consider a weaker variant of Type-II security In this notion the adversary is not allowed to replace public keys at any point and thus it cannot make decapsulation queries on identities whose public keys have been replaced. This is the traditional form of Type-II security, and is aimed at protecting the user against honest-but-curious key generation centres. Again a weak form, which we call Weak Type-II-CPA, can be defined which gives no access to any decapsulation oracle, this form of security will only be needed in the discussion leading up to our conclusions. There are other strengthenings of the Type-II model which try to model completely malicious key generation centres, see [15] for a discussion of these models. But we will not consider these in this paper. We summarize the oracle access for the security models we do consider by the following table:

	Request Public Key	Replace Public Key	Extract Full Full Private Key	Extract Partial Private Key	Strong Decap	Weak SV Decap	Decap
Strong Type-II	✔	✔	✔		✔		
Weak Type-II	✔	✔	✔				✔
Weak Type-II-CPA	✔	✔	✔				

Full Type-I security from Type-I* security and Strong Type-II security: In this section we justify our consideration of Type-I* security by showing that proving a scheme Type-I* secure is sufficient to get

"full" Type-I security if such a scheme also satisfies the strongest notion of Type-II security. In some sense this says that the definitions Type-I and Strong Type-II overlap in a specific case.

For ease of presentation we prove the theorem for the case of Strong Type-I security, but it is easy to see that it holds even if the scheme is Weak-Type-Ia*, Weak Type-Ib*, Weak Type-Ic* or Weak Type-I-CPA*. In this case one obtains the corresponding level of security (e.g. Weak Type-Ia). To complete the picture we recall that Dent noted in [15] that Weak Type-II security implies Weak Type-Ic security.

Theorem 1. *If a CL-KEM is Strong-Type-I* and Strong Type-II secure then it is Strong Type-I secure*

Proof. In order to prove this theorem we show that a Strong Type-I adversary $\mathcal{A}$ can be turned into another adversary against either Strong Type-I* or Strong Type-II security. We distinguish two types of Strong Type-I adversaries:

- $\mathcal{A}_1$ that replace the public key of the challenge identity ID^* *before* asking the challenge ciphertext;
- $\mathcal{A}_2$ that do *not* replace ID^*'s public key before asking the challenge ciphertext.

Let E be the event that the adversary asks ID^*'s partial private key during its attack. According to the definition of Strong Type-I security E can never occur in a run of $\mathcal{A}_1$. This means that the game played by an adversary $\mathcal{A}_1$ is exactly the same of a Strong Type-I* adversary.

On the other hand it is easy to see that an adversary $\mathcal{A}_2$ can be turned into an adversary $\mathcal{B}$ that breaks Strong Type-II security as follows. $\mathcal{B}$ receives in input the master secret key of the KGC so being able to provide $\mathcal{A}_2$ with the partial private key of any identity. Moreover $\mathcal{B}$ can answer all oracle queries made by $\mathcal{A}_2$ simply by forwarding such queries to its corresponding oracles. Since by definition $\mathcal{A}_2$ will not replace ID^*'s public key before asking the challenge ciphertext, then the simulations is perfect.

5 Generic Construction of CL-KEM from ID-KA

In this section we show our main result, namely a generic transform of any ID-KA protocol into a CL-KEM scheme.

Suppose we are given algorithms for a one-way authenticated ID-KA protocol ($\mathsf{KASetup}$, KeyDer, $\mathsf{Initiate}$, $\mathsf{Respond}$, Derive_I, Derive_R). Given a one-way identity-based key agreement protocol KA, we let CL(KA) denote the derived certificateless KEM obtained from the following algorithms.

- $\mathsf{CLSetup}(1^t)$. We run $(mpk^{\mathtt{KA}}, msk^{\mathtt{KA}}) \leftarrow \mathsf{KASetup}(1^t)$ and then set: $mpk^{\mathtt{CL}} \leftarrow mpk^{\mathtt{KA}}$ and $msk^{\mathtt{CL}} \leftarrow msk^{\mathtt{KA}}$.
- $\mathsf{Extract\text{-}Partial\text{-}Private\text{-}Key}(msk^{\mathtt{CL}}, ID)$. We set $d_{ID} \leftarrow \mathsf{KeyDer}(msk^{\mathtt{KA}}, ID)$.
- The pair $\mathsf{Set\text{-}Secret\text{-}Value}$ and $\mathsf{Set\text{-}Public\text{-}Key}$ are defined by running

$$(epk_{ID}, esk_{ID}) \leftarrow \mathsf{Initiate}(mpk^{\mathtt{KA}}, [d_{ID}]).$$

  The output of $\mathsf{Set\text{-}Secret\text{-}Value}$ is defined to be $s_{ID} = esk_{ID}$ and the output of $\mathsf{Set\text{-}Public\text{-}Key}$ is defined to be $pk_{ID} = epk_{ID}$.
- $\mathsf{Set\text{-}Private\text{-}Key}(d_{ID}, s_{ID})$ creates sk_{ID} by setting $sk_{ID} = (d_{ID}, s_{ID})$.
- $\mathsf{Enc}(mpk^{\mathtt{CL}}, pk_{ID}, ID)$. This runs as follows:
  - $(epk_0, esk_0) \leftarrow \mathsf{Respond}(mpk^{\mathtt{KA}})$.
  - $K \leftarrow \mathsf{Derive}_R(mpk^{\mathtt{KA}}, esk_0, pk_{ID}, ID)$.
  - $C \leftarrow epk_0$.
- $\mathsf{Dec}(mpk^{\mathtt{CL}}, sk_{ID}, C)$. Decapsulation is obtained by executing

$$K \leftarrow \mathsf{Derive}_I(mpk^{\mathtt{KA}}, d_{ID}, sk_{ID}, C).$$

In the above construction if the underlying ID-based key agreement protocol is *pure* (resp. *non-pure*), then we will obtain a *pure* (resp. *non-pure*) certificateless KEM, i.e. it will follow the original formulation of Al-Riyami and Paterson (resp. Baek *et al.*). To see this, notice that the $\mathsf{Set\text{-}Public\text{-}Key}$ function calls the $\mathsf{Initiate}(mpk^{\mathtt{KA}}, [d_I])$ operation, which itself may require d_I.

5.1 Security Results on the ID-KA to CL-KEM transforms

Once we have defined our black-box construction of CL-KEM from ID-KA protocols we prove its security in the theorems below. As one can see, the theorems show that the resulting CL-KEM can achieve different types of security according to the security of the underlying ID-KA protocol. As already discussed in the introduction, this relationship between the security models of ID-KA and CL-KEM sheds light on understanding which are the correct notion of security for the two primitives.

Theorem 2 (Type-I Security). *Consider the certificateless KEM CL(KA) derived from the one-way ID-based key agreement protocol KA as above:*

- *If KA is secure in the Reveal*-model then CL(KA) is Strong Type-I* secure as a certificateless KEM.*
- *If KA is secure in the Rewind model then CL(KA) is Weak Type-Ib* secure as a certificateless KEM.*
- *If KA is secure in the normal model then CL(KA) is Weak Type-I-CPA* secure as a certificateless KEM.*

In particular if $\mathcal{A}$ is an adversary against the CL(KA) scheme (in the above sense) then there is an adversary $\mathcal{B}$ against the KA scheme (also in the above sense) such that for pure *schemes we have*

$$\mathrm{Adv}^{\mathtt{Type-I}}_{\mathtt{CL-KEM}}(A) = \mathrm{Adv}_{ID-\mathtt{KA}}(B)$$

and for non-pure *schemes we have*

$$\mathrm{Adv}^{\mathtt{Type-I}}_{\mathtt{CL-KEM}}(A) \leq e \cdot (q_{pk} + 1) \cdot \mathrm{Adv}_{ID-\mathtt{KA}}(B)$$

where q_{pk} is the maximum number of extract public key queries issued by algorithm $\mathcal{B}$.

Proof. The proof of this theorem can be found in the appendices.

We notice that the proof technique does not allow the simulator to provide the partial private key of the challenge identity ID^*. Which is why our theorem is stated for the case of Strong Type-I* (resp. Weak Type-Ib* or Weak Type-I-CPA*). If we then apply the result of Theorem 1, along with the following theorems, we obtain full Strong Type-I security (resp. Weak Type-Ib or Weak Type-I-CPA) for the scheme CL(KA).

In looking at Type-II security we present two security theorems. The first one (Theorem 3) is conceptually simpler but requires our underlying identity based key agreement scheme to have a strong security property (i.e. it must support state reveal queries). The second theorem (Theorem 4) is more involved and does not provide such a tight reduction. On the other hand the second theorem requires less of a security guarantee on the underlying key agreement scheme. The proofs of both theorems can be found in the appendices.

Theorem 3 (Type-II Security – Mk I). *Consider the certificateless KEM CL(KA) derived from the one-way ID-based key agreement protocol KA as above:*

- *If KA satisfies master-key forward secrecy in the (StateReveal, Reveal*)-model then CL(KA) is Strong Type-II secure as a certificateless KEM.*
- *If KA satisfies master-key forward secrecy in the (StateReveal, Rewind)-model then CL(KA) is Weak Type-II secure as a certificateless KEM.*
- *If KA satisfies master-key forward secrecy in the StateReveal-model then CL(KA) is Weak Type-II-CPA secure as a certificateless KEM.*

In particular if $\mathcal{A}$ is an adversary against the CL(KA) scheme (in the sense described above) then there is an adversary $\mathcal{B}$ against the master-key forward secrecy of the KA scheme (also in the above sense) such that

$$\mathrm{Adv}^{\mathtt{Type-II}}_{\mathtt{CL-KEM}}(A) = \mathrm{Adv}^{mk-fs}_{ID-\mathtt{KA}}(B).$$

We now turn to showing that one does not necessarily need the *StateReveal* query to prove security, although the complication in the proof results in a less tight reduction.

Theorem 4 (Type-II Security – Mk II). *Consider the certificateless KEM CL(KA) derived from the one-way ID-based key agreement protocol KA as above:*

- *If KA satisfies master-key forward secrecy in the Reveal*-model then CL(KA) is Strong Type-II secure as a certificateless KEM.*
- *If KA satisfies master-key forward secrecy in the Rewind model then CL(KA) is Weak Type-II secure as a certificateless KEM.*
- *If KA satisfies master-key forward secrecy in the normal model then CL(KA) is Weak Type-II-CPA secure as a certificateless KEM.*

In particular if $\mathcal{A}$ is an adversary against the CL(KA) scheme (in the above sense) then there is an adversary $\mathcal{B}$ against the KA scheme (also in the above sense) then we have

$$\mathrm{Adv}^{\mathtt{Type-II}}_{\mathtt{CL-KEM}}(A) \leq e \cdot (q_{pk}+1) \cdot \mathrm{Adv}^{mk-fs}_{ID-\mathtt{KA}}(B)$$

where q_{pk} is the maximum number of extract public key queries issued by algorithm $\mathcal{B}$.

6 Identity-Based Key Encapsulation Mechanisms

In this section we are going to show the relationship between CL-KEM and identity-based KEMs. In particular we will give a generic transformation from any pure CL-KEM into an ID-KEM. As in the case of ID-KA and CL-KEM, here it is also interesting to observe how the different security models of CL-KEM transform into analogous models for ID-KEM.

Before giving our result, we recap on identity-based KEMs, the reader is referred to [8] for further details.

6.1 ID-KEM Definition

A ID-KEM scheme is specified by four polynomial time algorithms:

- $\mathsf{IDSetup}(1^t)$ is a PPT algorithm that takes as input 1^t and returns the master public keys mpk^{ID} and the master secret key msk^{ID}.
- $\mathtt{Extract}(msk^{ID}, ID)$. If $ID \in \mathcal{ID}$ is an identifier string for party ID this (possibly probabilistic) algorithm returns a partial private key d_{ID}.
- $\mathtt{Enc}(mpk^{ID}, ID)$ is the PPT encapsulation algorithm. On input of ID and mpk^{ID} this outputs a pair (C, K) where $K \in \mathbb{K}_{ID-\mathtt{KEM}}(mpk^{ID})$ is a key for the associated DEM and $C \in \mathbb{C}_{ID-\mathtt{KEM}}(mpk^{ID})$ is the encapsulation of that key.
- $\mathtt{Dec}(mpk^{ID}, d_{ID}, C)$ is the deterministic decapsulation algorithm. On input of C and d_{ID} this outputs the corresponding K or a failure symbol $\perp$.

6.2 ID-KEM Security Model

The security model for ID-KEMs is as follows. The adversary $\mathcal{A}$ plays a game with a challenger, at any point the adversary may request `Extract` queries for identities of his choice, and he may obtain decapsulations for pairs (ID, C) of his choice. At some point the adversary outputs a challenge identity ID^*. The challenger then calls $(C^*, K_0) \leftarrow \mathtt{Enc}(mpk^{ID}, ID^*)$, flips a bit b and samples a random key K_1 from the space $\mathbb{K}_{ID-\mathtt{KEM}}(mpk^{ID})$. The challenger then returns (C^*, K_b) to the adversary. The adversary then continues and finally outputs a guess b' for the hidden bit b.

The two oracles provided to the adversary, i.e. the `Extract` and `Dec` oracles, come with the following restrictions:

- `Extract` may at no point be called on the challenge identity ID^*;
- `Dec` may at no point be called on the pair (ID^*, C^*).

The above adversary is called an ID-IND-CCA adversary, if we disallow Dec queries then the adversary is an ID-IND-CPA adversary. The advantage of such a CCA adversary is defined to be

$$\mathrm{Adv}^{ID-\mathtt{IND}-\mathtt{CCA}}_{ID-\mathtt{KEM}}(\mathcal{A}) = |2\Pr[b = b'] - 1|\,,$$

with a similar definition for a CPA adversary. A scheme is deemed to be secure if for all adversaries $\mathcal{A}$ the advantage is a negligible function of the security parameter.

6.3 Generic Construction of ID-KEM from pure CL-KEM

To construct an ID-KEM from a CL-KEM the obvious solution is to set the user public/private keys to be trivial and known to all parties. This however can only be done for pure CL-KEMs since in non-pure schemes one does not have complete control over the public/private keys, since they depend on the partial private key d_{ID}. We call the resulting scheme the ID(CL) scheme, as it is an ID-KEM built from a CL-KEM.

Theorem 5. *Consider the pure ID-KEM ID(CL) derived from the pure CL-KEM scheme CL as above. Then if CL is Strong Type-I* secure then ID(CL) is ID-IND-CCA secure. In particular if $\mathcal{A}$ is an adversary against the ID(CL) scheme then there is an adversary $\mathcal{B}$ against the CL-KEM scheme such that*

$$\mathrm{Adv}^{ID-\mathtt{IND}-\mathtt{CCA}}_{ID-\mathtt{KEM}}(A) = \mathrm{Adv}^{\mathtt{Strong}-\mathtt{Type}-\mathtt{I}*}_{\mathtt{CL}-\mathtt{KEM}}(B).$$

Proof. Assume there is an efficient adversary $\mathcal{A}$ that is able to break the security of ID(CL) with non-negligible advantage ϵ. We build a simulator $\mathcal{B}$ which breaks the security of the underlying CL scheme.

The algorithm $\mathcal{B}$ is relatively trivial. The input master public key $mpk^{\mathtt{CL}}$ for algorithm $\mathcal{B}$ is first passed to algorithm $\mathcal{A}$. When $\mathcal{A}$ makes a Extract query for identity ID, algorithm $\mathcal{B}$ makes a request for the partial private key of party ID. It also replaces the public key of ID with the trivial key required for the ID(KA) construction. Any Dec queries made by $\mathcal{A}$ are passed onto the Strong Decap oracle provided to algorithm $\mathcal{B}$. When $\mathcal{A}$ outputs the challenge identity ID^* this is passed on by algorithm $\mathcal{B}$ to its challenger, who then responds with a C^* which is passed directly back to algorithm $\mathcal{A}$.

It is clear that all restrictions on oracles queries by $\mathcal{B}$ do not affect the responses to oracle queries made by $\mathcal{A}$. In addition the advantage of $\mathcal{A}$ is equal to the advantage of $\mathcal{B}$.

7 Conclusion: Which Certificateless Model is Correct?

In this section we summarize the conclusions we have drawn from our analysis. It is worth pointing out that these are personal conclusions, and we leave the reader to draw their own analysis.

Firstly, all our conclusions are predicated on the assumption that our transforms are all "natural", in that they are the obvious way to convert an ID-KA protocol into a CL-KEM and a CL-KEM into an ID-KEM. If these are the natural transformations then the underlying security and syntactic models should also transform naturally.

Pure vs Non-Pure First we discuss the issue of pure vs non-pure certificateless schemes. Our transform from CL-KEMs to ID-KEMs requires the underlying CL-KEM to be pure. This is not surprising as an essential feature of ID-based cryptography is that of the identity (and hence the associated secret key) being independent of all parameters bar the actual identity. It is not surprising even because non-pure CL-KEMs are the only ones that can be constructed without pairings.

We draw two conclusions from this. First, the pure syntax is more powerful as it enables functionalities such as encryption-into-the-future (a.k.a. workflow). Second, we can say that certificateless encryption is a primitive simpler than ID-based encryption, although people have usually thought at the former as an extension of the latter. When ID-based encryption was proposed [20], one of its main motivations was to avoid the certificates management issues of standard public key encryption. Then it took almost twenty years to have IBE schemes, basically thanks to the idea of exploiting pairings. From our considerations we can say that the "hard part" of constructing ID-based encryption is not avoiding certificates, but achieving those additional properties (e.g. workflows); i.e. technically speaking, having a user's public key independent of the scheme parameters.

CPA Security Before turning to CCA security of certificateless encryption we first consider the simpler case of CPA security. We remarked in the introduction that the [18] construction of ID-based encryption from ID-based NIKD schemes only produces CPA secure schemes, unless one assumes an oracle equivalent to our *Reveal** oracle.

Similar considerations apply in our case. The construction of ID-KEMs from CL-KEMs will produce a CPA secure ID-KEM if the underlying CL-KEM is Weak Type-I-CPA* secure. Note, that we only require Weak Type-I-CPA* and not Weak Type-I-CPA security.

In constructing CL-KEMs from ID-KA protocols we need to consider what security is required of the underlying ID-KA protocol to ensure Weak Type-I-CPA and Weak Type-II-CPA security of the CL-KEM. Our theorems show that a sufficient condition is that the underlying ID-KA protocol is secure in the standard sense, i.e. with no *Reveal**, *Rewind* or *StateReveal* oracles. Although the security reduction is tighter if we assume the adversary has access to *StateReveal* oracles, i.e. we use a CK-like security model for ID-based key agreement.

We note that the security reductions go through more naturally when one considers the CL-KEM to have Weak Type-I-CPA* security and Weak Type-II-CPA security. We then obtain the full Weak Type-I-CPA by appealing to the analogue of Theorem 1.

CCA Security Our theorems show that to obtain full Strong Type-I and Strong Type-II security of the derived CL-KEM we require the ID-based key agreement security model to give the adversary access to our *Reveal** oracle. This is a very non-standard oracle for key agreement protocols, but this should not be surprising. Essentially CCA security for an encryption scheme means the adversary has to be able to open anything, even something created in an illegitimate way (even if the opening results in the $\perp$ symbol). All our *Reveal** oracle does is to provide the adversary against the ID-based key agreement scheme with an oracle to open anything.

A similar remark as to Strong Type-I* as opposed to Strong Type-I security as mentioned in the above comments on CPA security also applies in this case.

Summary So in summary we believe the correct syntactic security definitions for CL-KEMs should be schemes with Strong Type-I* and Strong Type-II security where the pure syntax allows for more properties. By using Strong Type-I* as the security definition instead of Strong Type-I we obtain a natural seperation between the two security notions, rather than dealing with the cases in the intersection twice.

However, our construction from ID-based key agreement schemes would seem to imply that the correct security definition should be one which uses *StateReveal* queries (i.e. one which follows the analogue of CK-security). However, it also implies that the model also includes *Reveal** queries, which seems to provide an extreme form of security definition for key agreement schemes. Since it would seem silly to define security for normal key agreement schemes and ID-based key agreement schemes in a different manner, this would imply that standard key agreement schemes should also be defined to be secure in the presence of a *Reveal** oracle. This final conclusion is somewhat unsatifactory, and we hope our work will inspire other researchers to investigate this connection.

8 Acknowledgements

The third author was supported by a Royal Society Wolfson Research Merit Award.

References

1. M. Abdalla, M. Bellare and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *Topics in Cryptology – CT-RSA 2001*, Springer-Verlag LNCS 2020, 143–158, 2001.
2. S.S. Al-Riyami. *Cryptographic schemes based on elliptic curve pairings.* Ph.D. Thesis, University of London, 2004.

3. S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. In *Advances in Cryptology – Asiacrypt 2003*, Springer-Verlag LNCS 2894, 452–473, 2003.
4. S.S. Al-Riyami and K.G. Paterson. CBE from CL-PKE: A generic construction and efficient schemes. In *Public Key Cryptography – PKC 2005*, Springer-Verlag LNCS 3386, 398–415, 2005.
5. J. Baek, R. Safavi-Naini and W. Susilo. Certificateless public key encryption without pairing. In *Information Security – ISC 2005*, Springer-Verlag LNCS 3650, 134–148, 2005.
6. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – Crypto '93*, Springer-Verlag LNCS 773, 232–249, 1993.
7. S. Blake-Wilson, D. Johnson and A. Menezes. Key agreement protocols and their security analysis. In *Cryptography and Coding*, Springer-Verlag LNCS 1355, 30–45, 1997.
8. K. Bentahar, P. Farshim, J. Malone-Lee and N.P. Smart. Generic constructions of identity-based and certificateless KEMs. *J. Cryptology*, **21**, 178–199, 2008. Full version at IACR e-print 2005/058.
9. D. Boneh and X. Boyen. Short Signatures without Random Oracles. *Advances in Cryptology – Eurocrypt 2004*, Springer-Verlag LNCS 3027, 56–73, 2004.
10. R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable Zero-Knowledge. *Weizmann Science Press of Israel*, 1999.
11. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. *Advances in Cryptology – Eurocrypt 2001*, Springer-Verlag LNCS 2045, 453–474, 2001.
12. L. Chen, Z. Cheng and N.P. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Security*, **6**, 213–241, 2007.
13. L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. In *IEEE Computer Security Foundations Workshop*, 219–233, 2003. The modified version of this paper is available at Cryptology ePrint Archive, Report 2002/184.
14. K.-K.R. Choo, C. Boyd and Y. Hitchcock. Examining indistinguishabilit-based proof models for key establishment protocols. *Advances in Cryptology – Asiacrypt 2005*, Springer-Verlag LNCS 3788, 585–604, 2005.
15. A. Dent. A Survey of Certificateless Encryption Schemes and Security Models. *in International Journal of Information Security*, **7**, 347–377, 2008.
16. D. Fiore and R. Gennaro. Making the Diffie–Hellman protocol identity-based. In *Topics in Cryptology – CT-RSA 2010*, LNCS 5985, 2010. Also in IACR e-print archive, report 2009/174.
17. N. McCullagh and P.S.L.M. Barreto. A new two-party identity-based authenticated key agreement. In *Topics in Cryptology – CT-RSA 2005*, Springer-Verlag LNCS 3376, 262–274, 2005.
18. K. Paterson and S. Srinivasan. On the relations between non-interactive key distribution, identity based-based encryption and trapdoor discrete log groups. *Designs, Codes and Cryptography*, **52**, 219–241, 2009.
19. M. Scott. Authenticated ID-based key exchange and remote log-in with insecure token and PIN number. Cryptology ePrint Archive, Report 2002/164.
20. Adi Shamir Identity-Based Cryptosystems and Signature Schemes *Advances in Cryptology – Proceedings of CRYPTO '84*, 1985, pp. 47-53
21. N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, **38**, 630–632, 2002.

A Proof of Theorem 2

Assume there exists an efficient adversary $\mathcal{A}$ that is able to break the Strong Type-I* (resp. Weak Type-Ib* or Weak Type-I-CPA*) security of CL(KA) with non-negligible advantage ϵ. Then we show how to build a simulator $\mathcal{B}$ that exploits $\mathcal{A}$ to break the relevant security of the one-way authenticated KA protocol. We shall deal with the three parts of the Theorem together, so we shall deal with the different types of Type-I security in the one argument.

Setup $\mathcal{B}$ receives in input from its challenger the master public key $mpk^{\mathtt{KA}}$ of the KGC and hands such key to $\mathcal{A}$. The simulator also maintains a table $KeyList$ where it stores the keys of identities involved in the simulation and other extra informations related to them. The table contains tuples of the form $\langle ID, epk_{ID}, esk_{ID}, d_{ID}, c_{ID}\rangle$ where each element is explained below.

Phase 1 Let us show how to deal with each oracle. But before hand we present a generic subroutine, which will be used by almost all oracles. If algorithm $\mathcal{A}$ adversary makes an oracle call for an identity ID which

has not been seen before then algorithm $\mathcal{B}$ before proceeding as in our examples below, first processes the new identity as follows:

Pure case: In the case of a pure underlying key-agreement scheme the simulator executes $\mathit{NewSession}(ID, R)$ and then $(epk_{ID}, esk_{ID}) \leftarrow \mathsf{Initiate}(mpk^{\mathtt{KA}})$ and stores $\langle ID, epk_{ID}, esk_{ID}, \perp, 2\rangle$ into $\mathit{KeyList}$.

Non-pure case: It flips a binary coin $c_{ID} \in \{0, 1\}$ such that $\Pr[c_{ID} = 0] = \delta$ for some δ specified later. Algorithm B creates a new session by running $\mathit{NewSession}(ID, R)$ and then;

- If $c_{ID} = 0$ the simulator obtains the value d_{ID} by calling $\mathit{Corrupt}(ID)$. It then executes $(epk_{ID}, esk_{ID}) \leftarrow \mathsf{Initiate}(mpk^{\mathtt{KA}}, d_{ID})$ and stores the tuple $\langle ID, epk_{ID}, esk_{ID}, d_{ID}, 0\rangle$ into $\mathit{KeyList}$.
- If $c_{ID} = 1$ then $\mathcal{B}$ queries the oracle $\mathit{Send}(\Pi_{ID,R}, \mathit{initiator}, \perp)$. It gets back a message epk_{ID} which is then stored in $\mathit{KeyList}$ as the tuple $\langle ID, epk_{ID}, \perp, \perp, 1\rangle$.

Flipping the coin c_{ID} is equivalent to make a guess on the challenge identity ID^* that should remain uncorrupted. We now turn to describing how each oracle query made by $\mathcal{A}$ is answered by $\mathcal{B}$.

1. **Request public key of ID.** By the above we are gauranteed that $\mathit{KeyList}$ contains an entry of the form $\langle ID, epk_{ID}, *, *, *\rangle$, and so algorithm $\mathcal{B}$ simply returns epk_{ID}.
2. **Extract partial private key of ID.** First the simulator searches in $\mathit{KeyList}$ for a tuple $\langle ID, *, *, *, c_{ID}\rangle$ and then proceeds as follows:
   - If $c_{ID} = 0$ then $\mathcal{B}$ outputs the corresponding partial private key d_{ID}.
   - If $c_{ID} = 1$ then $\mathcal{B}$ terminates the simulation and outputs Abort.
   - If $c_{ID} = 2$ then $\mathcal{B}$ obtains d_{ID} via corruption and updates the entry on $\mathit{KeyList}$ with the obtained value, and sets $c_{ID} = 0$. The value of d_{ID} is returned to $\mathcal{A}$.
3. **Extract full private key of ID.** Algorithm $\mathcal{B}$ proceeds as in the previous step, note this means that if $c_{ID} = 1$ then the algorithm aborts. The entry on the $\mathit{KeyList}$ is then of the form $\langle ID, epk_{ID}, esk_{ID}, d_{ID}, 0\rangle$ and so the pair $sk_{ID} = (d_{ID}, esk_{ID})$ is returned to $\mathcal{A}$.
4. **Replace the public key of ID with pk'_{ID}.** Algorithm $\mathcal{B}$ looks for a tuple containing ID in $\mathit{KeyList}$ and replaces the current public key with pk'_{ID}. Notice, these queries only occur in the case of Strong Type-I* adversaries in our theorem.
5. **Strong Decap query (C, ID).** Recall, these queries can occur only if $\mathcal{A}$ is a Strong Type-I* adversary. In this case $\mathcal{B}$ constructs a query to the Reveal^* oracle for the pair two parties ID and R, with the respective message flows epk_{ID} and C. Notice, that the response from this oracle will even deal with the case where the public key has been replaced.
6. **Decap query** (C, ID) Recall, these queries can only occur if $\mathcal{A}$ is a Weak Type-Ib* adversary.
   - If $c_{ID} = 0$ we use its secret key to compute the decapsulated key.
   - If $c_{ID} = 1$ and $\delta_{\Pi_{ID,I}} = \perp$ then the simulator invokes $\mathit{Send}(\Pi_{ID,R}, \mathit{initiator}, C)$ and then $K \leftarrow \mathit{Reveal}(\Pi_{ID,R})$. The output K is the decapsulated key returned to the adversary.
   - If $c_{ID} = 1$ and $\delta_{\Pi_{ID,R}} =$ "*accepted*", then algorithm $\mathcal{B}$ first asks $\mathit{Rewind}(\Pi_{ID,R})$ and then proceeds as in the step before.
   - If $c_{ID} = 2$ then we proceed as if $c_{ID} = 1$.

Challenge At some point $\mathcal{A}$ outputs a target identity ID^*. If $ID^* \notin \mathit{KeyList}$ then algorithm $\mathcal{B}$ first generates a public key for it as above, using the case $c_{ID^*} = 1$ in the case of non-pure schemes.

If $ID^* \in \mathit{KeyList}$ and $c_{ID^*} = 0$ then algorithm $\mathcal{B}$ terminates the simulation and aborts.

Otherwise, since we then know that ID^* is not corrupted, the simulator performs the following actions:

1. ask to initiate a session $\mathit{NewSession}(R, ID^*)$
2. $C^* \leftarrow \mathit{Send}(\Pi_{R,ID^*}, \mathit{responder}, epk_{ID^*})$
3. $K \leftarrow \mathit{Test}(\Pi_{I,ID^*})$

Finally $\mathcal{B}$ hands (C^*, K) to $\mathcal{A}$.

Phase 2 This is simulated as Phase 1. Notice that from now on $\mathcal{B}$ cannot ask a reveal query on Π_{R,ID^*} (and its matching oracle $\Pi_{ID^*,R}$). However according to Type-I game's rules $\mathcal{A}$ will not ask a Strong Decap or Decap query for (C^*, ID^*).

Guess At the end the adversary outputs a decision bit b' and $\mathcal{B}$ returns the same bit.

The simulation is perfect if $\mathcal{B}$ does not abort during the entire simulation. And algorithm $\mathcal{B}$ will only abort in the case of non-pure schemes, in this case the probability that $\mathcal{B}$ wins is bounded by

$$\begin{aligned}\Pr[\mathcal{B}\ wins] &= \Pr[\mathcal{B}\ wins|\mathcal{B}\neg Abort] \cdot \Pr[\mathcal{B}\neg Abort] \\ &\quad + \Pr[\mathcal{B}\ wins|\mathcal{B}\ Abort] \cdot \Pr[\mathcal{B}\ Abort] \\ &\leq \Pr[\mathcal{A}\ wins|\mathcal{B}\neg Abort] \cdot \Pr[\mathcal{B}\neg Abort] + \frac{1}{2} - \frac{1}{2} \cdot \Pr[\mathcal{B}\neg Abort] \\ &= \frac{1}{2} + \epsilon \cdot \Pr[\mathcal{B}\neg Abort]\end{aligned}$$

The probability that $\mathcal{B}$ does not abort during the simulation is $(1-\delta)\delta^{q_{pk}}$ where q_{pk} is the number of public key queries issued by the adversary during the simulation. Since $\mathcal{A}$ is polynomially-bounded the value of q_{pk} is also bounded by a polynomial. Moreover the value $(1-\delta)\delta^{q_{pk}}$ is maximised at $\delta = 1 - 1/(q_{pk}+1)$. This means that the probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(q_{pk}+1)}$. In conclusion we have that $\mathcal{B}$'s advantage for non-pure schemes is at least $\frac{\epsilon}{e(q_{pk}+1)}$, which is non-negligible if we assume that $\mathcal{A}$'s advantage ϵ is also non-negligible.

B Proof of Theorem 3

Assume there exists an efficient adversary $\mathcal{A}$ that is able to break the relevant Type-II security of CL(KA) with non-negligible advantage ϵ. Then we show how to build a simulator $\mathcal{B}$ that exploits $\mathcal{A}$ to break the relevant master-key forward secrecy property of the one-way authenticated KA protocol. We shall deal with the three parts of the Theorem together, so we shall deal with the different types of Type-II security in the one argument.

Setup Algorithm $\mathcal{B}$ receives in input from its challenger the master public key $mpk^{\mathtt{KA}}$ and the master secret key $msk^{\mathtt{KA}}$ and hands such keys to $\mathcal{A}$. The simulator also maintains a table $KeyList$ where it stores the keys of identities involved in the simulation and other extra informations related to them. The table contains tuples of the form $\langle ID, epk_{ID}, esk_{ID}, d_{ID}\rangle$ where each element is explained below.
When an identity ID is asked by $\mathcal{A}$ for the first time, algorithm $\mathcal{B}$ creates a new session $NewSession(ID, R)$ and then queries the oracle $Send(\Pi_{ID,R},\ initiator, \perp)$ obtaining epk_{ID}. It then queries $Corrupt(ID)$ so as to obtain d_{ID}. Algorithm $\mathcal{B}$ then inserts $\langle ID, epk_{ID}, \perp, d_{ID}\rangle$ into $KeyList$.
Note that $\mathcal{A}$ can now construct d_{ID} values on its own, as can $\mathcal{B}$. However, if the algorithm to produce d_{ID} given ID and $msk^{\mathtt{KA}}$ (resp. $msk^{\mathtt{CL}}$) is probabilistic then the value produced by $\mathcal{B}$ might not correspond to that produced by $\mathcal{A}$. This explains our need to use the *Corrupt* oracle provided to $\mathcal{B}$ above.

Phase 1 We show how to deal with each oracle query made by $\mathcal{A}$.

1. **Request public key of ID.** By the above we know $KeyList$ contains a tuple of the form $\langle ID, epk_{ID}, *, d_{ID}\rangle$ so $\mathcal{B}$ simply outputs epk_{ID}.
2. **Extract full private key of ID.** If $\langle ID, epk_{ID}, esk_{ID}, d_{ID}\rangle$ does not appear in $KeyList$ then $\mathcal{B}$ asks $esk_{ID} \leftarrow StateReveal(\Pi_{ID,R})$ to its challenger, and esk_{ID} is placed in this entry of the $KeyList$. In either case the pair $sk_{ID} = (d_{ID}, esk_{ID})$ is returned to $\mathcal{A}$.
3. **Replace the public key of ID with pk'_{ID}.** Notice, these queries only occur in the case of Strong Type-II adversaries in our theorem. Algorithm $\mathcal{B}$ looks for a tuple containing ID in $KeyList$ and replaces the second component with the value pk'_{ID}.
4. **Strong Decap query (C, ID).** Recall, these queries can occur only if $\mathcal{A}$ is a Strong Type-II adversary. In this case $\mathcal{B}$ constructs a query to the $Reveal^*$ oracle for the pair two parties ID and R, with the respective message flows epk_{ID} (obtained from the $KeyList$) and C. Notice, that the response from this oracle will even deal with the case where the public key has been replaced.
5. **Decap query** (C, ID) Recall, these queries can only occur if $\mathcal{A}$ is a Weak Type-II adversary.
   - If $\delta_{\Pi_{ID,R}} = \perp$ then the simulator invokes $Send(\Pi_{ID,R}, initiator, C)$ and then $K \leftarrow Reveal(\Pi_{ID,R})$. The output K is the decapsulated key returned to the adversary.

– If $\delta_{\Pi_{ID,R}}$ = "*accepted*", then algorithm $\mathcal{B}$ first asks $Rewind(\Pi_{ID,R})$ and then proceeds as in the step before.

Challenge At some point $\mathcal{A}$ outputs a target identity ID^*. The simulator then performs the following actions, using the value of epk_{ID^*} from the entry in $KeyList$;

1. ask to initiate a session $NewSession(R, ID^*)$,
2. $C^* \leftarrow Send(\Pi_{R,ID^*}, responder, epk_{ID^*})$,
3. $K \leftarrow Test(\Pi_{I,ID^*})$

Finally $\mathcal{B}$ hands (C^*, K) to $\mathcal{A}$.

Phase 2 This is simulated as Phase 1. Notice that from now on $\mathcal{B}$ cannot ask a reveal query on Π_{R,ID^*} (and its matching oracle $\Pi_{ID^*,R}$). However according to Type-II game's rules $\mathcal{A}$ will not ask a Strong Decap or Decap query for (C^*, ID^*).

Guess At the end the adversary outputs a decision bit b' and $\mathcal{B}$ returns the same bit.

Note that the simulation is perfect and $\mathcal{B}$ wins with the same advantage of $\mathcal{A}$.

C Proof of Theorem 4

Assume there exists an efficient adversary $\mathcal{A}$ that is able to break Type-II security of CL(KA) with non-negligible advantage ϵ. Then we show how to build a simulator $\mathcal{B}$ that exploits $\mathcal{A}$ to break the master-key forward-secrecy of the one-way authenticated KA protocol. Again we deal with the three different types of Type-II security in the one argument.

Setup Algorithm $\mathcal{B}$ receives in input from its challenger the master public key $mpk^{\mathtt{KA}}$ and the master secret key $msk^{\mathtt{KA}}$ of the KGC and hands such key to $\mathcal{A}$. The simulator also maintains a table $KeyList$ where it stores the keys of identities involved in the simulation and other extra informations related to them. The table contains tuples of the form $\langle ID, pk_{ID}, epk_{ID}, d_{ID}, c_{ID}\rangle$ where each element is explained below.
As before everytime $\mathcal{A}$ mentions an identity ID for the first time algorithm $\mathcal{B}$ creates an entry in $KeyList$ before proceedings. For proving this theorem this initial entry is constructed as follows: It first queires $Corrupt(ID)$ to obtain d_{ID} and then flips a binary coin $c_{ID} \in \{0, 1\}$ such that $\Pr[c_{ID} = 0] = \delta$ for some δ specified later.

– If $c_{ID} = 0$ the simulator runs $(epk_{ID}, esk_{ID}) \leftarrow \mathsf{Initiate}([d_{ID}], mpk^{\mathtt{KA}})$, and stores $\langle ID, epk_{ID}, esk_{ID}, d_{ID}, c_{ID}\rangle$ into $KeyList$.
– If $c_{ID} = 1$ $\mathcal{B}$ creates a new session $NewSession(ID, R)$ and queries the oracle $Send(\Pi_{ID,R}, initiator, \perp)$. It then stores the tuple $\langle ID, epk_{ID}, \cdot, d_{ID}, c_{ID}\rangle$ into $KeyList$.

Flipping the coin c_{ID} is equivalent to make a guess on whether the challenge identity ID^* is the subject of a extract full private key query at some point.

Phase 1 Let us now show how to deal with each oracle

1. **Request public key of ID.** By above $Keyist$ contains an entry of the form $\langle ID, epk_{ID}, *, d_{ID}, c_{ID}\rangle$, so the value of epk_{ID} is returned to $\mathcal{A}$.
2. **Extract full private key of ID.** First the simulator searches in $KeyList$ for the tuple $\langle ID, epk_{ID}, esk_{ID}, d_{ID}, c_{ID}\rangle$. If $c_{ID} = 1$ then $\mathcal{B}$ terminates the simulation and outputs Abort. Otherwise, if $c_{ID} = 0$, $\mathcal{B}$ outputs the corresponding secret key $sk_{ID} = (d_{ID}, esk_{ID})$.
3. **Replace the public key of ID with pk'_{ID}.** Algorithm $\mathcal{B}$ looks for a tuple containing ID in $KeyList$ and replaces the current public key with pk'_{ID}. Notice, these queries only occur in the case of Strong Type-II adversaries in our theorem.
4. **Strong Decap query** (C, ID)**.** Recall, these queries can occur only if $\mathcal{A}$ is a Strong Type-II adversary. In this case $\mathcal{B}$ constructs a query to the $Reveal^*$ oracle for the pair two parties ID and R, with the respective message flows epk_{ID} and C. Notice, that the response from this oracle will even deal with the case where the public key has been replaced.
5. **Decap query** (C, ID) Recall, these queries can only occur if $\mathcal{A}$ is a Weak Type-II adversary.
   – If $c_{ID} = 0$, we use its secret key to compute the decapsulated key.

– If $c_{ID} = 1$ and $\delta_{\Pi_{ID,R}} = \perp$ then the simulator invokes $Send(\Pi_{ID,R},\ initiator, C)$ and then $K \leftarrow Reveal(\Pi_{ID,R})$. The output K is the decapsulated key returned to the adversary.
– If $c_{ID} = 1$ and $\delta_{\Pi_{ID,R}} =$ "$accepted$", then algorithm $\mathcal{B}$ first asks $Rewind(\Pi_{ID,R})$ and then proceeds as in the step before.

Challenge At some point $\mathcal{A}$ outputs a target identity ID^*. If $ID^* \notin KeyList$ then algorithm $\mathcal{B}$ first generates an entry for it as above, using the case $c_{ID^*} = 1$. If $ID^* \in KeyList$ and $c_{ID^*} = 0$ then algorithm $\mathcal{B}$ terminates the simulation and aborts. Otherwise the simulator performs the following actions:

1. ask to initiate a session $NewSession(R, ID^*)$
2. $C^* \leftarrow Send(\Pi_{R,ID^*}, responder, pk_{ID^*})$
3. $K \leftarrow Test(\Pi_{R,ID^*})$

Finally $\mathcal{B}$ hands (C^*, K) to $\mathcal{A}$.

Phase 2 This is simulated as Phase 1. Notice that from now on $\mathcal{B}$ cannot ask a reveal query on Π_{R,ID^*} (and its matching oracle $\Pi_{ID^*,R}$). However according to Type-II game's rules $\mathcal{A}$ will not ask a Strong Decap or Decap query for (C^*, ID^*).

Guess At the end the adversary outputs a decision bit b' and $\mathcal{B}$ returns the same bit.

The simulation is perfect if $\mathcal{B}$ does not abort during the entire simulation, hence the probability that $\mathcal{B}$ wins is bounded by

$$\begin{aligned}
\Pr[\mathcal{B}\ wins] &= \Pr[\mathcal{B}\ wins | \mathcal{B}\neg Abort] \cdot \Pr[\mathcal{B}\neg Abort] \\
&\quad + \Pr[\mathcal{B}\ wins | \mathcal{B}\ Abort] \cdot \Pr[\mathcal{B}\ Abort] \\
&\leq \Pr[\mathcal{A}\ wins | \mathcal{B}\neg Abort] \cdot \Pr[\mathcal{B}\neg Abort] + \frac{1}{2} - \frac{1}{2} \cdot \Pr[\mathcal{B}\neg Abort] \\
&= \frac{1}{2} + \epsilon \cdot \Pr[\mathcal{B}\neg Abort]
\end{aligned}$$

The probability that $\mathcal{B}$ does not abort during the simulation is $(1-\delta)\delta^{q_{pk}}$ where q_{pk} is the number of public key queries issued by the adversary during the simulation. Since $\mathcal{A}$ is polynomially-bounded the value of q_{pk} is also bounded by a polynomial. Moreover the value $(1-\delta)\delta^{q_{pk}}$ is maximised at $\delta = 1 - 1/(q_{pk}+1)$. This means that the probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(q_{pk}+1)}$.

In conclusion we have that $\mathcal{B}$'s advantage is at least $\frac{\epsilon}{e(q_{pk}+1)}$ which is non-negligible if we assume that $\mathcal{A}$'s advantage ϵ is also non-negligible.