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

D. Fiore[1], R. Gennaro[2], and N.P. Smart[3]

[1] Dipartimento di Matematica e Informatica,
Universita' di Catania,
Viale A. Doria no 6,
95125 Catania,
Italy. fiore@dmi.unict.it

[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. We illustrate our models and constructions with two running examples, one pairing based and one non-pairing based. 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

In this paper we discuss the relationship between ID-based key agreement protocols (ID-KA), certificateless encryption (CL) and ID-based key encapsulation mechanisms (ID-KEMs) 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 the distinctions between what we term *pure* ID-based schemes and *non-pure* schemes, in various security models. 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). Such pure schemes allow various functionalities such as encryption into-the-future etc, yet interestingly there are no-known non-pairing based pure schemes, either in the ID-based key agreement or certificateless settings.

We present a variety of generic constructions and relationships between security models, for the various scheme types we consider. In particular we present security models for ID-based key agreement which do not "look natural" when considered in analogy to normal key agreement schemes, but which look more natural when considered in terms of the models used in certificateless encryption. For example, we augment the standard ID-based key agreement security models by giving the adversary extra powers akin to those used in certificateless schemes. These extra oracles are relatively mild, but may result in some previously considered secure schemes becoming insecure.

Our work aims to shed light on the distinction between pure and non-pure schemes, and also aids the examination of what is the "correct" security model for certificateless encryption. Indeed if one was going to take the natural analogues from the ID-based key agreement setting, one would not have the strong security notions for CL. However, in the other direction our work can be seen as highlighting that perhaps the security models for ID-based key agreement (and maybe ordinary key agreement) are not strong enough.

Our main generic constructions can be summarized by reference to Figure 1, the definitions used in the arrows will become clear as we define them in the following pages.

$$\begin{tikzcd}[column sep=huge] \text{ID-KA} \arrow[r, "\begin{array}{l}\text{ka } \mathit{Reveal}^{*} \Longrightarrow \text{Strong Type-I*}\\ \text{ka } \mathit{Rewind} \Longrightarrow \text{Weak Type-Ib*}\end{array}", "\begin{array}{l}\text{mk-fs } \mathit{Reveal}^{*}\\ \quad\Longrightarrow \text{Strong Type-II}\\ \text{mk-fs } \mathit{Rewind}\\ \quad\Longrightarrow \text{Weak Type-II}\end{array}"'] & \text{CL} \arrow[r, "\text{Pure Only}", "\begin{array}{l}\text{Strong Type-I*}\\ \quad\Longrightarrow \text{ID-IND-CCA}\end{array}"'] & \text{ID-KEM} \end{tikzcd}$$

Fig. 1. Relationships Between Schemes

We illustrate our models and constructions with two running examples, one that uses pairing (the ID-based key agreement scheme SCK-2 [12]), and one that does not (the Fiore–Gennaro [16] ID-based key agreement scheme, referred as the FG protocol in what follows). The former scheme is pure (and pairing based), whilst the latter is non-pure (and non-pairing based). From these examples one can see that the distinction into pure and non-pure schemes is important, since from a pure ID-based key agreement scheme we are able to derive tighter security results for certificateless schemes, and we can also construct ID-based KEMs. Indeed, by following our generic constructions through we obtain for the SCK-2 key agreement scheme the ID-KEM of Lynn [17]. Indeed we recover the security result of [8] on the Lynn ID-KEM, namely that it is secure in the Random Oracle Model under the Gap-BDH problem. The Lynn ID-KEM is itself the natural KEM analogue of the Boneh–Franklin scheme [10].

We observe that the CL scheme obtained from the FG protocol gives us almost the same scheme of Baek *et al.* [5] with two main advantages. First, the proof of security of the Baek *et al.*'s scheme was found incorrect, while our transformation provides security for free from that of the FG protocol. Second, the resulting CL scheme is extremely efficient.

We consider weakened notions 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. 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.

2 Identity-Based Key Agreement

We will only consider two pass ID-based key agreement protocols in this paper. 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 polynomial time algorithms. The two passes are illustrated in Figure 2. We let $\mathcal{ID}$ denote the set of possible user identities and $\mathbb{K}_{\mathsf{KA}}(mpk^{\mathsf{KA}})$ be the set of valid session keys for the public parameter mpk^{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^{KA} and the master secret key msk^{KA}.
- $\mathsf{KeyDer}(msk^{\mathsf{KA}}, ID)$ is the private key extraction algorithm. It takes as input msk^{KA} and $ID \in \mathcal{ID}$ and it returns the associated private key d_{ID}. This algorithm may be deterministic or probabilistic.
- $\mathsf{Initiate}(mpk^{\mathsf{KA}}, d_I)$. 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[1]
- $\mathsf{Respond}(mpk^{\mathsf{KA}}, d_R)$. 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).
- $\mathsf{Derive}_I(mpk^{\mathsf{KA}}, d_I, esk_I, epk_R, R)$. This is a (possibly probabilistic) algorithm run by the initiator to derive the session key $K_I \in \mathbb{K}_{\mathsf{KA}}(mpk^{\mathsf{KA}})$ with party R.

[1] 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.

– $\mathsf{Derive}_R(mpk^{\mathtt{KA}}, d_R, esk_R, epk_I, I)$. 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.

$$\begin{array}{ccc}
\textbf{Initiator} & & \textbf{Responder} \\
d_I\ ,\ mpk^{\mathtt{KA}} & & d_R\ ,\ mpk^{\mathtt{KA}} \\
(epk_I, esk_I) \leftarrow \mathsf{Initiate}(mpk^{\mathtt{KA}}, d_I) & \xrightarrow{epk_I} & \\
 & \xleftarrow{epk_R} & (epk_R, esk_R) \leftarrow \mathsf{Respond}(mpk^{\mathtt{KA}}, d_R) \\
K_I \leftarrow \mathsf{Derive}_I(mpk^{\mathtt{KA}}, d_I, esk_I, epk_R, R) & & K_R \leftarrow \mathsf{Derive}_R(mpk^{\mathtt{KA}}, d_R, esk_R, epk_I, I)
\end{array}$$

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 [18] families. The majority of pairing-based ID-based key agreement protocols lie in the Smart [20] 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.

We will be using a modified version of the Bellare-Rogaway key exchange model, as extended to an ID-based setting. It presented in Appendix A.

2.2 Two Example Protocols

In what follows we will focus, as our motivating examples, on the pairing based SCK protocol [12] and the non-pairing based FG protocol [16]. Both have emphemeral keys which do not depend on the intended recipient, and both are role symmetric. However, the SCK protocol is a *pure* ID-based key agreement protocol, whereas the FG protocol is *non-pure*.

We use the additive notation for group operations when describing SCK-2 (since this protocol is pairing-based and therefore must be implemented over elliptic curves), and the multiplicative notation when describing FG (which can be implemented over any cyclic group). Also with $\leftarrow$ we denote 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.

SCK-2 Protocol: This is described in the setting of an asymmetric bilinear pairing $\hat{h} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, where $\mathbb{G}_1 = \langle P_1 \rangle$ and $\mathbb{G}_2 = \langle P_2 \rangle$ are groups of prime order q. We require two hash functions, $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}_2$ and $H_2 : \mathbb{G}_1 \times \mathbb{G}_T \longrightarrow \{0,1\}^t$, where t is the size of the agreed key.

The master public and private keys are generated by

$$msk^{\mathsf{KA}} \leftarrow \mathbb{Z}_q \text{ and } mpk^{\mathsf{KA}} \leftarrow [msk^{\mathsf{KA}}]P_1.$$

The private key extraction is performed by setting

$$d_{ID} = [msk^{\mathsf{KA}}]H_1(ID).$$

The $\mathsf{Initiate}$ and $\mathsf{Respond}$ message flows are identical to a standard Diffie–Hellman key exchange in the group $\mathbb{G}_1$, i.e. the parties set $esk_I, esk_R \leftarrow \mathbb{Z}_q$ and then compute (and transmit) $epk_I = [esk_I]P_1$ and $epk_R = [esk_R]P_1$. The Derive_I algorithm is then given by

$$H_2\Big([esk_I]epk_R\ ,\ \hat{h}([esk_I]mpk^{\mathsf{KA}}, H_1(R)) \cdot \hat{h}(epk_R, d_I)\Big),$$

with Derive_R being given by

$$H_2\Big([esk_R]epk_I\ ,\ \hat{h}([esk_R]mpk^{\mathsf{KA}}, H_1(I)) \cdot \hat{h}(epk_I, d_R)\Big).$$

We note that this protocol is pure and role symmetric, and that the output key is equal to

$$H_2\Big([esk_I \cdot esk_R]P_1\ ,\ \hat{h}(mpk^{\mathsf{KA}}, [esk_R]H_1(I) + [esk_I]H_1(R))\Big).$$

Security of the SCK-2 Protocol: We first define the following problems in our pairing based group, all problems we assume are given relative to an "oracle" which computes a homomorphism $\phi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$ such that $\phi(P_2) = P_1$. See [12] for a discussion of this "oracle" in the context of Type 3 pairings.

Definition 1 (Pairing Problems) *We define four problems in the above pairing situation:*

- **BDH Problem:** *For $a, b, c \in \mathbb{Z}_q^*$, given $([a]P_2, [b]P_2, [c]P_1)$ compute the pairing value $\hat{h}(P_1, P_2)^{abc}$.*
- **XDH Problem:** *For $a, b \in \mathbb{Z}_q^*$, given $([a]P_2, [b]P_2)$ compute $[ab]P_1$.*
- **DBDH Problem:** *For $a, b, c, r \in \mathbb{Z}_q^*$ differentiate the tuple*
$$([a]P_2, [b]P_2, [c]P_1, \hat{h}(P_1, P_2)^r)$$
from the tuple
$$([a]P_2, [b]P_2, [c]P_1, \hat{h}(P_1, P_2)^{abc}).$$
- **Gap-BDH Problem:** *Solve the BDH problem with access to an oracle which solves the DBDH problem.*

Given these definitions we have the following security results for the SCK-2 protocol:

Theorem 1. *The following statements are all in the random oracle model.*

1. *If the BDH problem is hard then the SCK-2 protocol is a secure ID-KA protocol in the Rewind-model.*
2. *If the Gap-BDH problem is hard then the SCK-2 protocol is a secure ID-KA protocol in the Reveal*-model.*
3. *If the XDH problem is hard then the SCK-2 protocol is master-key forward secure in the Rewind-model.*

Proof. The first part follows from the proof of Theorem 1 in [12], which is only given for the normal non-*Rewind* model, but it is seen to easily generalise to our situation.

The second part again follows from the proof of Theorem 1 in [12]. To answer the *Reveal** queries we need to add extra entries into the H_2-list within the proof in [12]. However, to ensure consistency of the entries we add we then need a DBDH oracle.

The third part follows from Theorem 2 of [12].

FG Protocol: The construction makes use of a group $\mathbb{G} = \langle g \rangle$ of prime order q, and two hash functions: $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ and $H_2 : \mathbb{G} \times \mathbb{G} \to \{0,1\}^l$, where l is the bit length of the derived keys.

$\mathsf{KASetup}(1^t)$
- $x \leftarrow \mathbb{Z}_q$
- $y \leftarrow g^x$
- Define $msk^{\mathtt{KA}} \leftarrow x$, $mpk^{\mathtt{KA}} \leftarrow y$.

$\mathsf{KeyDer}(msk^{\mathtt{KA}}, U)$
- $k \leftarrow \mathbb{Z}_q$
- $r_U \leftarrow g^k$
- $s_U \leftarrow k + H_1(U \| r_U) \cdot x$.
- We set $d_U \leftarrow (r_U, s_U)$.

$\mathsf{Initiate}(mpk^{\mathtt{KA}}, d_I)$
- $esk_I \leftarrow \mathbb{Z}_q$
- $u_I \leftarrow g^{esk_I}$
- Set $epk_I \leftarrow (r_I, u_I)$.

$\mathsf{Respond}(mpk^{\mathtt{KA}}, d_I)$
- $esk_R \leftarrow \mathbb{Z}_q$
- $u_R = g^{esk_R}$
- Set $epk_R \leftarrow (r_R, u_R)$.

$\mathsf{Derive}_I(mpk^{\mathtt{KA}}, d_I, esk_I, epk_R, R)$
- $z_1 \leftarrow (u_R \cdot r_R \cdot y^{H_1(R\|r_R)})^{s_I + esk_I}$
- $z_2 \leftarrow u_R^{esk_I}$
- $s_U \leftarrow H_2(z_1, z_2)$.

$\mathsf{Derive}_R(mpk^{\mathtt{KA}}, d_R, esk_R, epk_I, I)$
- $z_1 \leftarrow (u_I \cdot r_I \cdot y^{H_1(I\|r_I)})^{s_R + esk_R}$
- $z_2 \leftarrow u_I^{esk_R}$
- $s_U \leftarrow H_2(z_1, z_2)$.

Note, that the protocol is non-pure as the $\mathsf{Initiate}$ and $\mathsf{Respond}$ protocols require access to the parties private key.

Security of the FG Protocol: We first define the assumptions that are needed to prove the security of the protocol. In the following assume $\mathbb{G}$ to be a cyclic multiplicative group of order q where q is a ℓ-bit long prime. We assume that there are efficient algorithms to perform multiplication and membership test in $\mathbb{G}$. Finally we denote with g a generator of $\mathbb{G}$.

Definition 2 (Diffie–Hellman Problems) *We define four problems in the above situation of a cyclic group G.*

- **CDH Problem:** *For $a, b \in \mathbb{Z}_q^*$, given (g^a, g^b) compute g^{ab}.*
- **DDH Problem:** *For $a, b, c \in \mathbb{Z}_q^*$, differentiate the tuple $(U, V, W) = (g^a, g^b, g^c)$ from the tuple $(U, V, W) = (g^a, g^b, g^{ab})$.*
- **Gap-DDH Problem:** *Solve the CDH problem with access to an oracle which solves the DDH problem. The oracle is denoted by $\mathsf{DH}(U, V, W)$.*
- **SDH Problem:** *The Strong Diffie–Hellman problem[2] is a weaker version of the Gap-DDH problem in that the input to the CDH problem are two elements U and V but the oracle is restricted to queries of the form $\mathsf{DH}(U, \cdot, \cdot)$, i.e. the first entry is fixed to U.*

Given these definitions we have the following security results for the FG protocol:

Theorem 2. *The following statements are all in the random oracle model.*

1. *If the Strong-DH problem is hard then the FG protocol is a secure ID-KA protocol in the Rewind-model.*

[2] We remark that in recent papers the name strong Diffie-Hellman assumption was used to denote a different conjecture defined over bilinear groups [9]. In this paper, we refer to the original terminology from [1]

2. *If the Gap-DH problem is hard then the FG protocol is a secure ID-KA protocol in the Reveal*-model.*
3. *If the CDH problem is hard then the FG protocol is master-key forward secure in the Rewind-model.*
4. *If the Gap-DH problem is hard then the FG protocol is master-key forward secure in the Reveal*-model.*

Proof. Theorem 3 in [16] proves the first part in a non-*Rewind* model. But it is easy to see that the proof can be generalised to support rewinding as the same technique used to answer one reveal query can be extended to answer to *any* reveal query when the message coming from the simulator is fixed.

If Theorem 3 of [16] assumes the Gap-DH assumption instead of Strong-DH then the DH oracle can be used to simulate the *Reveal** oracle and gives a proof for the second part.

The third part follows directly from the fact that the FG protocol satisfies master-key forward secrecy. The proof given in [16] can be easily extended to the *Rewind*-model for the same reason of point 1.

Finally, in order to prove the last point, we observe that in the presence of the Gap-DH oracle it is possible to extend the proof of master-key forward secrecy given in [16] to support *Reveal** queries.

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 the modifications to the previous section which are needed for 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 $\mathsf{Respond}(mpk^{\mathsf{KA}})$ rather than $\mathsf{Respond}(mpk^{\mathsf{KA}}, d_{R_0})$. Similarly we call

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

rather than

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

3.1 Two Example Protocols

We carry on with our two running examples:

SCK-2 Protocol: To convert the SCK-2 protocol to one which is only one-way authenticated, we set R_0 to be an "identity" such that $H_1(R_0) = \mathcal{O}$, i.e. the point at infinity on the curve. It does not matter that we cannot find such an R_0 since we will never actually use the precise value of R_0.

The private key extraction for legitimate users is performed as before, as are the $\mathsf{Initiate}$ and $\mathsf{Respond}$ algorithms. The only difference is now in the Derive_I and Derive_R methods, which are defined by:

$$H_2\left([esk_I]epk_R\ ,\ \hat{h}(epk_R, d_I)\right)$$

and

$$H_2\left([esk_R]epk_I\ ,\ \hat{h}([esk_R]mpk^{\mathtt{KA}}, H_1(I))\right),$$

respectively. Note that the output key is now equal to

$$H_2\left([esk_I \cdot esk_R]P_1\ ,\ \hat{h}(mpk^{\mathtt{KA}}, [esk_R]H_1(I))\right).$$

FG Protocol: To convert the FG protocol to one which is only mutually authenticated, we can obtain a highly efficient scheme by selecting $s_{R_0} = 0$, and hence setting $r_{R_0} = 1$, and "fixing" the random oracle to be such that $H_1(R_0 \| r_{R_0}) = 0$. So the $\mathsf{Respond}$ algorithm simply selects $esk_{R_0} \leftarrow \mathbb{Z}_q$, and then computes $epk_{R_0} = g^{esk_{R_0}}$. The Derive algorithms become:

$\mathsf{Derive}_R(mpk^{\mathtt{KA}}, esk_{R_0}, epk_{ID}, ID)$.
- $z_1 \leftarrow (u_{ID} \cdot r_{ID} \cdot y^{H_1(ID \| r_{ID})})^{esk_{R_0}}$
- $z_2 \leftarrow u_{ID}^{esk_{R_0}}$
- $s_U \leftarrow H_2(z_1, z_2)$.

$\mathsf{Derive}_I(mpk^{\mathtt{KA}}, d_{ID}, esk_{ID}, epk_{R_0})$.
- $z_1 \leftarrow epk_{R_0}^{s_{ID}+esk_{ID}}$
- $z_2 \leftarrow epk_{R_0}^{esk_{ID}}$
- $s_U \leftarrow H_2(z_1, z_2)$.

Thus we see that Derive_R becomes considerably simpler when considered in the one-way-authenticated protocol case.

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]. 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

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

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.

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 denies the ability to both replace public keys entirely.

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 now define.

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.

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.

Full Type-I security from Type-I* security and Strong Type-II security: In this section we briefly discuss Type-I* security and show that proving a scheme Type-I* secure is sufficient to get "full" Type-I security if such a scheme 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* or Weak Type-Ib*. In this case one obtains the corresponding level of security (e.g. Weak Type-Ia or Weak Type-Ib). To complete the picture we recall that Dent noted in [15] that Weak Type-II security implies Weak Type-Ic security.

Theorem 3. *If a CL-KEM is Strong-Type-I* and Strong Type-II secure then it is Strong Type-I secure*

Proof. The proof can be found in the appendices.

5 Generic Construction of CL-KEM from ID-KA

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 $msk^{\mathtt{CL}} \leftarrow msk^{\mathtt{KA}}$.
– $\mathsf{Extract\text{-}Partial\text{-}Private\text{-}Key}(msk^{\mathtt{CL}}, ID)$. This is defined to be $d_{ID} \leftarrow \mathsf{KeyDer}(msk^{\mathtt{KA}}, ID)$.

– The pair Set-Secret-Value and Set-Public-Key are defined by running

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

The output of Set-Secret-Value is defined to be $s_{ID} = esk_{ID}$ and the output of Set-Public-Key is defined to be $pk_{ID} = epk_{ID}$.

– Set-Private-Key(d_{ID}, s_{ID}) creates sk_{ID} by setting $sk_{ID} = (d_{ID}, s_{ID})$.

– Enc$(mpk^{\mathtt{CL}}, pk_{ID}, ID)$. This runs as follows:
  - $(epk_0, esk_0) \leftarrow \mathsf{Respond}(mpk^{\mathtt{KA}})$.
  - $K \leftarrow \mathsf{Derive}_R(mpk^{\mathtt{KA}}, esk_0, pk_{ID}, ID)$.
  - $C \leftarrow epk_0$.

– 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 Set-Public-Key function calls the Initiate$(mpk^{\mathtt{KA}}, [d_I])$ operation, which itself may require d_I.

5.1 Two Example Protocols

Using the two previous identity-based key agreement protocols we are able to present the following certificateless key encapsulation schemes. The first one is a pure scheme, while the second is a non-pure scheme, this follows from the properties of the underlying key agreement protocols.

SCK-2 Protocol: We assume the same set-up for the SCK-2 key agreement scheme in terms of $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, H_1 and H_2. The derived CL-KEM is then defined by:

– CLSetup(1^t). We set $x \leftarrow \mathbb{Z}_q$, compute $Y \leftarrow [x]P_1$, and define $msk^{\mathtt{CL}} \leftarrow x$ and $mpk^{\mathtt{CL}} \leftarrow Y$.

– Extract-Partial-Private-Key$(msk^{\mathtt{CL}}, ID)$. The issuer computes $d_{ID} = [x]H_1(ID)$.

– Set-Secret-Value, Set-Public-Key and Set-Private-Key are defined by setting $t_{ID} \leftarrow \mathbb{Z}_q$ and then computing $pk_{ID} \leftarrow [t_{ID}]P_1$. Finally we set $sk_{ID} = (d_{ID}, t_{ID})$.

– The algorithms Enc$(mpk^{\mathtt{CL}}, pk_{ID}, ID)$ and Dec$(mpk^{\mathtt{CL}}, sk_{ID}, C)$ are given by:

Enc$(mpk^{\mathtt{CL}}, pk_{ID}, ID)$
- $t \leftarrow \mathbb{Z}_q$.
- $C \leftarrow [t]P_1$.
- $Z_1 \leftarrow [t]pk_{ID}$.
- $Z_2 \leftarrow \hat{h}([t]mpk^{\mathtt{CL}}, H_1(ID))$
- $k \leftarrow H_2(Z_1, Z_2)$.

Dec$(mpk^{\mathtt{CL}}, sk_{ID}, C)$.
- $Z_1' \leftarrow [t_{ID}]C$
- $Z_2' \leftarrow \hat{h}(C, d_{ID})$
- $k \leftarrow H_2(Z_1', Z_2')$.

FG Protocol: We assume the same set-up for the FG protocol in terms of $\mathbb{G}$, H_1 and H_2. The derived CL-KEM is then defined by:

- $\mathsf{CLSetup}(1^t)$. We set $x \leftarrow \mathbb{Z}_q$, compute $y \leftarrow g^x$. We then define $msk^{\mathtt{CL}} \leftarrow x$ and $mpk^{\mathtt{CL}} \leftarrow y$.
- $\mathsf{Extract\text{-}Partial\text{-}Private\text{-}Key}(msk^{\mathtt{CL}}, ID)$. The issuer sets $k \leftarrow \mathbb{Z}_q$ and computes $r_{ID} \leftarrow g^k$ and $s_{ID} \leftarrow k + H_1(ID \| r_{ID}) \cdot x$. We set $d_{ID} \leftarrow (r_{ID}, s_{ID})$.
- Set-Secret-Value, Set-Public-Key and Set-Private-Key are defined by setting $t_{ID} \leftarrow \mathbb{Z}_q$ and then computing $pk_{ID} \leftarrow (r_{ID}, u_{ID}) = (r_{ID}, g^{t_{ID}})$. Finally we set $sk_{ID} = (d_{ID}, t_{ID})$.
- $\mathsf{Enc}(mpk^{\mathtt{CL}}, pk_{ID}, ID)$ and $\mathsf{Dec}(mpk^{\mathtt{CL}}, sk_{ID}, c)$ are then given by:

$\mathsf{Enc}(mpk^{\mathtt{CL}}, pk_{ID}, ID)$
- $t \leftarrow \mathbb{Z}_q$.
- $c \leftarrow g^t$.
- $z_1 \leftarrow (u_{ID} \cdot r_{ID} \cdot y^{H_1(ID \| r_{ID})})^t$
- $z_2 \leftarrow u_{ID}^t$
- $k \leftarrow H_2(z_1, z_2)$.

$\mathsf{Enc}(mpk^{\mathtt{CL}}, pk_{ID}, ID)$
- $z_1' \leftarrow c^{t_{ID}+s_{ID}}$.
- $z_2' \leftarrow c^{t_{ID}}$
- $k \leftarrow H_2(z_1', z_2')$.

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

Theorem 4 (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.*

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*

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

and for non-pure *schemes we have*

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

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*). If we then apply the result of Theorem 3, along with the following theorems, we obtain Strong Type-I security (resp. Weak Type-Ib).

In looking at Type-II security we present two security theorems. The first one (Theorem 5) is conceptually simpler but requires our underlying identity based key agreement scheme to have a strong security property. The second theorem (Theorem 6) 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 5 (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* $(\mathit{StateReveal}, \mathit{Reveal}^*)$*-model then CL(KA) is Strong Type-II secure as a certificateless KEM.*
- *If KA satisfies master-key forward secrecy in the* $(\mathit{StateReveal}, \mathit{Rewind})$*-model then CL(KA) is Weak Type-II 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 need the *StateReveal* query to prove security, although the complication in the proof results in a less tight reduction.

Theorem 6 (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.*

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 $\mathcal{B}$.

By combining Theorems 1, 2, 3, 4 and 6 we obtain the following corollary.

Corollary 1. *The following security results follow, in the random oracle model, for our two example CL-KEMs:*

– *The SCK-2 based CL-KEM is Strong Type-I* secure assuming the Gap-BDH problem is hard.*
– *The SCK-2 based CL-KEM is Strong Type-II secure assuming the XDH problem is hard.*
– *The SCK-2 based CL-KEM is Strong Type-I secure assuming the Gap-BDH and XDH problems are hard.*
– *The FG based CL-KEM is Strong Type-I* and Strong-Type-II secure assuming the Gap-DH problem is hard.*
– *The FG based CL-KEM is Strong Type-I secure assuming the Gap-DH problem is hard.*
– *The FG based CL-KEM is Weak Type-Ib secure assuming the Strong-DH problem is hard.*
– *The FG based CL-KEM is Weak Type-II secure assuming the CDH problem is hard.*

6 Identity-Based Key Encapsulation Mechanisms

For lack of space the last result of the paper on the construction of ID-based public key encryption schemes can be found in Appendix F.

The main result of this section is that we can construct ID-based KEMs from *pure* CL-KEMs with a very tight security reduction.

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. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *Advances in Cryptology – Crypto 2001*, Springer-Verlag LNCS 2139, 213–229, 2001.
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. IACR e-print 2009/174, 2009.
17. B. Lynn. Authenticated identity-based encryption. IACR e-print 2002/072, 2002.
18. 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.
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. N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, **38**, 630–632, 2002.

A 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.

- $\gamma_{\mathcal{O}} \in \{\perp, \mathit{corrupted}, \mathit{revealed}\}$. 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
$$\mathit{pid}_{\mathcal{O}} = V \text{ and } s_{\mathcal{O}} = \mathit{role}_{\mathcal{O}} = \delta_{\mathcal{O}} = \gamma_{\mathcal{O}} = \perp .$$
However, if any other oracle with identity U has been corrupted then we set $\gamma_{\mathcal{O}} = \mathit{corrupted}$.
- *Send*$(\mathcal{O}, \mathit{role}, \mathit{msg})$. 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 $\mathit{role} = \mathit{initiator}$ then
    - If $\mathit{msg} = \perp$, $\delta_{\mathcal{O}} = \perp$ and $\mathit{role}_{\mathcal{O}} = \perp$ then set $\mathit{role}_{\mathcal{O}} = \mathit{initiator}$ and output a message (i.e. send the first message flow in the protocol);
    - If $\mathit{msg} \neq \perp$ and $\mathit{role}_{\mathcal{O}} = \mathit{initiator}$ (i.e. *msg* is the second message flow in the protocol) then compute $s_{\mathcal{O}}$ and set $\delta_{\mathcal{O}} = \mathit{accepted}$;
    - Else set $\delta_{\mathcal{O}} = \mathit{error}$ and return $\perp$
  - If $\mathit{role} = \mathit{responder}$ then
    - If $\mathit{msg} \neq \perp$ and $\mathit{role}_{\mathcal{O}} = \perp$ then compute $s_{\mathcal{O}}$, set $\delta_{\mathcal{O}} = \mathit{accepted}$, $\mathit{role}_{\mathcal{O}} = \mathit{responder}$ and respond with a message (i.e. send the second message flow in the protocol).
    - Else set $\delta_{\mathcal{O}} = \mathit{error}$ and return $\perp$.
- *Reveal*$(\mathcal{O})$. If $\delta_{\mathcal{O}} \neq \mathit{accepted}$ or $\gamma_{\mathcal{O}} = \mathit{corrupted}$ then this returns $\perp$, otherwise it returns $s_{\mathcal{O}}$ and we set $\gamma_{\mathcal{O}} = \mathit{revealed}$.
- *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 $\gamma_{\mathcal{O}} = \mathit{corrupted}$. 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 3 *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.

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 $epk_I \leftarrow \mathsf{Initiate}(mpk^{\mathsf{KA}})$ (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.

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, 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 security model is 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 $\mathit{Rewind}(\mathcal{O})$ oracle the adversary can take a test oracle (which has output the ephemeral public key g^x) and obtain, using a combination of the $\mathit{Rewind}(\mathcal{O})$ and $\mathit{Reveal}(\mathcal{O})$ 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 $\mathit{Reveal}^*(I, R, \mathit{epk}_I, \mathit{epk}_R)$ is a very strong oracle. 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.

B Proof of Theorem 3

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.

C Proof of Theorem 4

Assume there exists an efficient adversary $\mathcal{A}$ that is able to break the Strong Type-I* (resp. Weak Type-Ib*) 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 two 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^{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 $NewSession(ID, R)$ and then $(epk_{ID}, esk_{ID}) \leftarrow \mathsf{Initiate}(mpk^{\mathsf{KA}})$ and stores $\langle ID, epk_{ID}, esk_{ID}, \perp, 2\rangle$ into $KeyList$.

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 $NewSession(ID, R)$ and then;

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

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

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.

D Proof of Theorem 5

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 two 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 $mpk^{\texttt{KA}}$ and the master secret key $msk^{\texttt{KA}}$ 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 $\mathit{NewSession}(ID, R)$ and then queries the oracle $\mathit{Send}(\Pi_{ID,R}, \mathit{initiator}, \perp)$ obtaining epk_{ID}. It then queries $\mathit{Corrupt}(ID)$ 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 $msk^{\texttt{KA}}$ (resp. $msk^{\texttt{CL}}$) 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 $\mathit{Corrupt}$ 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 $esk_{ID} \leftarrow \mathit{StateReveal}(\Pi_{ID,R})$ 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 $\mathcal{A}$.

E Proof of Theorem 6

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 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.

F Identity-Based Key Encapsulation Mechanisms

We recap here on identity-based KEMs, the reader is referred to [8] for further details.

F.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 $\perp$.

F.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

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

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.

F.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.

Example: The SCK-2 Protocol: Only one of our CL-KEM examples given before is a pure scheme, namely the one derived from the SCK-2 ID-based key agreement protocol. Thus we continue the discussion using this protocol as a motivating example. Again we assume the same set-up for the SCK-2 key agreement scheme in terms of $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, and H_1. But now we defined H_2 as a hash function from $\mathbb{G}_T$ to $\{0,1\}^t$. The derived ID-KEM is then defined by, by setting t_{ID} to be equal to zero in the underlying CL-KEM;

- $\mathsf{IDSetup}(1^t)$. We set $x \leftarrow \mathbb{Z}_q$, compute $Y \leftarrow [x]P_1$, and define $msk^{ID} \leftarrow x$ and $mpk^{ID} \leftarrow Y$.
- $\mathtt{Extract}(msk^{ID}, ID)$. The issuer computes $d_{ID} = [x]H_1(ID)$.
- $\mathtt{Enc}(mpk^{ID}, ID)$. This runs as follows:
  - $t \leftarrow \mathbb{Z}_q$.
  - $C \leftarrow [t]P_1$.
  - $Z \leftarrow \hat{h}([t]mpk^{ID}, H_1(ID))$
  - $k \leftarrow H_2(Z)$.
- $\mathtt{Dec}(mpk^{ID}, d_{ID}, C)$. To decapsulate the receiver computes
  - $Z' \leftarrow \hat{h}(C, d_{ID})$

- $k \leftarrow H_2(Z')$.

We note that this has resulted in the Boneh–Franklin based ID-KEM first proposed by Lynn in [17]. This scheme is was proved to be secure in the random oracle model in the full version of [8] under the hardness of the Gap-BDH problem. In this latter paper the ID-KEM is referred to by the name "Construction 1".

F.4 Security results on the CL-KEM to ID-KEM transform

Theorem 7. *Consider the pure ID-KEM ID(CA) derived from the pure CL-KEM scheme CL as above. Then if CL is Strong Type-I* secure then ID(CA) is ID-IND-CCA secure. In particular if $\mathcal{A}$ is an adversary against the ID(CA) scheme then there is an adversary $\mathcal{B}$ against the CL-KEM scheme such that*

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

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 $\mathcal{B}$.

We now apply this theorem to the SCK-2 scheme. Combining with the first bullet point in Corollary 1 we obtain that the ID-KEM derived from the SCK-2 scheme is ID-IND-CCA secure assuming the Gap-BDH problem is hard. Thus we have recovered, albeit in a round about way, the result of [8] on the ID-KEM of Lynn [17].