Identity Based Key Encapsulation with Wildcards

James Birkett[1], Alexander W. Dent[1], Gregory Neven[2], and Jacob Schuldt[1]

[1] Information Security Group,
Royal Holloway, University of London,
Egham, TW20 0EX, UK.
{j.m.birkett,a.dent,jacob.schuldt}@rhul.ac.uk
[2] Department of Electrical Engineering, Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium;
and Département d'Informatique, Ecole Normale Supériure,
45 Rue d'Ulm, 75005 Paris, France.
Gregory.Neven@esat.kuleuven.be

Abstract. We propose a hybrid (KEM/DEM) model for the recently proposed primitive of identity-based encryption with wildcards (WIBE), and confirm that the hybrid construction is secure. We also propose new chosen-ciphertext secure WIBE schemes that have somewhat more efficient security reductions and some performance benefits. Our first construction is a generic one from any one-way secure WIBE in the random oracle model, the second is a direct construction in the standard model.

1 Introduction

One of the major obstacles for the deployment of public-key cryptography in the real world is the secure linking of users to their public keys. While typically solved through public-key infrastructures (PKI), identity-based encryption [15, 14, 9, 7] can avoid some of the costs related to PKIs because it simply uses the identity of a user (e.g., her email address) as her public key. This way, Bob can for example send an encrypted email to Alice by encrypting it under her identity `alice@cs.univ.edu`, which only Alice can decrypt using the secret key that only she can obtain from a trusted key distribution centre.

Abdalla *et al.* [1] recently proposed a very intuitive extension to this idea by allowing the recipient identity to contain *wildcards*. A ciphertext can then be decrypted by multiple recipients with related identities. For example, Bob can send an encrypted email to the entire computer science department by encrypting under identity `*@cs.univ.edu`, or to all system administrators in the university by encrypting under identity `sysadmin@*.univ.edu`.

As is the case for most public-key and identity-based encryption schemes, the identity-based encryption with wildcards (WIBE) schemes of [1] can only be used to encrypt relatively short messages, typically about 160 bits. To encrypt longer messages, one will have to resort to *hybrid* techniques: the sender uses the WIBE to encrypt a fresh symmetric key K and encrypts the actual message under the key K. The basic construction has been used within the cryptographic community for years, dating back to the work of Blum and Goldwasser in 1984 [4], but its security for the case of public-key encryption was not properly analysed until the work of Cramer and Shoup [10]. In their model, the encryption scheme consists of a separate key encapsulation mechanism (KEM), and data encapsulation mechanism (DEM).

In this paper, we propose a KEM-DEM model for WIBE schemes, and prove that the combination of a CPA/CCA secure KEM and a CPA/CCA secure DEM yields a CPA/CCA WIBE scheme. This result may be rather unsurprising, but its proof is necessary to validate the use of hybrid techniques for the case of WIBEs,

in the same way that it was necessary for the public-key [10] and identity-based [3] cases. Furthermore, it may be noted that subtleties can arise in the proving of such results, for example in the case of certificateless KEMs [3].

More interestingly, we present constructions for CCA-secure WIBEs. Security against adaptive chosen-ciphertext attacks [13] is generally considered to be the required security level for most practical uses. Previous constructions of L-level CCA-secure WIBEs [1] have either required the user to store 2^L private keys or have suffered from a security reduction in which the security of a level L WIBE was related to the security of a level $2L+2$ HIBE. This reduction comes at a huge cost. Since the security of all currently known WIBE schemes degrades exponentially with the number of levels, the resulting WIBE will necessarily be restricted to only half of the (already small) original hierarchy depth. One could consider replacing the one-time signature scheme with a MAC, as done for the case of (H)IBE schemes in [8], but even if one manages to overcome the subtleties in the proof the scheme still suffers from the very limited hierarchy depth.

Instead, we present direct CCA-secure constructions of WIB-KEMs, which in combination with a CCA-secure DEM immediately yield new CCA-secure WIBEs. We first present a generic construction in the random oracle model [2] along the lines of Dent [11] from any WIBE that is one-way secure under chosen-plaintext attack, a much weaker requirement yielding very efficient schemes. This construction could also be considered a potential method for producing secure HIBEs. Next, we present a construction in the standard model based on the variant of the Waters encryption scheme [17] suggested by Kiltz and Galindo [12]. This scheme falls within the Boneh-Boyen framework [5].

2 Definitions

2.1 Notation

We will use the following notation in this paper:

S^n	denotes the set of vectors $(s_1, \ldots, s_n)$, where $s_i \in S$. We will also use this notation for strings of length n.
S^*	denotes the set of arbitrary length vectors, i.e $\bigcup_{i \in \mathbb{Z}} S^i$.
$x \xleftarrow{\$} S$	x is an element selected uniformly at random from the set S.
$x \leftarrow \mathsf{A}(y, z)$	x is the output of running the algorithm A with two inputs, y and z. A may be deterministic or probabilistic.
$x \leftarrow \mathsf{A}(y, z; r)$	x is the output of running the probabilistic algorithm $\mathsf{A}(y, z)$, but with randomness given by r.
$x \leftarrow \mathsf{A}^{\mathcal{O}}(y, z)$	x is the output of $\mathsf{A}(x, y)$ when run with access to an oracle $\mathcal{O}$.

2.2 Syntax of WIBEs, WIB-KEMs and DEMs

WIBEs A pattern P is a tuple $(P_1, \ldots, P_l) \in (\{0,1\}^* \cup \{*\})^l$, for some $l \leq L$, where L is the maximum number of levels. An identity $ID = (ID_1, \ldots, ID_{l'})$ "matches" the pattern P if $l' \leq l$ and for all $1 \leq i \leq l'$, $ID_i = P_i$ or $P_i = *$. We write this as $ID \in_* P$.

A WIBE scheme consists of the following algorithms:

- Setup generates a master key pair (mpk, msk).
- $\mathsf{KeyDer}(d_{ID}, ID_{l+1})$ takes the secret key d_{ID} for $ID = (ID_1, \ldots, ID_l)$, generates a secret key $d_{ID'}$ for the identity $ID' = (ID_1, \ldots, ID_{l+1})$. The root user, who has identity $\varepsilon = ()$, uses $d_\varepsilon = msk$ as his private key. This will be used to derive keys for single level identities.

– $\mathsf{Encrypt}(mpk, P, m)$ encrypts a message $m \in \{0,1\}^*$ intended for all identities matching a pattern P, and returns a ciphertext C.
– $\mathsf{Decrypt}(d_{ID}, C)$ decrypts ciphertext C using the secret key d_{ID} for an identity $ID \in_* P$ and returns the corresponding message m. If the encryption is invalid, the $\mathsf{Decrypt}$ algorithm "rejects" by outputting $\perp$.

We will overload the notation for key derivation, writing $\mathsf{KeyDer}(msk, ID)$ to mean repeated application of the key derivation function in the obvious way.

Soundness requires that for all key pairs (mpk, msk) output by Setup, all $0 \leq l \leq L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^l$, all identities ID such that $ID \in_* P$, and all messages $m \in \{0,1\}^*$:

$$\Pr\left[\mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), \mathsf{Encrypt}(mpk, P, m)) = m\right] = 1\,.$$

WIB-KEMs We will now define an Identity-Based Key Encapsulation Mechanism with Wildcards (WIB-KEM). A WIB-KEM consists of the following algorithms:

– Setup and KeyDer algorithms are defined as in the WIBE case.
– $\mathsf{Encap}(mpk, P)$ takes the master public key mpk of the system and a pattern P, and returns (K, C), where $K \in \{0,1\}^\lambda$ is a one-time symmetric key and C is an encapsulation of the key K.
– $\mathsf{Decap}(mpk, d_{ID}, C)$ takes a private key d_{ID} for an identity $ID \in_* P$ and an encapsulation C, and returns the corresponding secret key K. If the encapsulation is invalid, the Decap algorithm "rejects" by outputting $\perp$.

A WIB-KEM must satisfy the following soundness property: for all key pairs (mpk, msk) output by Setup, all $0 \leq l \leq L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^l$, and all identities $ID \in_* P$,

$$\Pr\left[K' = K : (K, C) \leftarrow \mathsf{Encap}(mpk, P); K' \leftarrow \mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C)\right] = 1\,.$$

HIBEs and HIB-KEMs can be thought of as special cases WIBEs and WIB-KEMs restricted to patterns without wildcards.

DEMs A DEM consists of a pair of deterministic algorithms:

– $\mathsf{Encrypt}(K, m)$ takes a key $K \in \{0,1\}^\lambda$, and a message m of arbitrary length and outputs a ciphertext C.
– $\mathsf{Decrypt}(K, C)$ takes a key $K \in \{0,1\}^\lambda$ and outputs either the corresponding message m or the "reject" symbol $\perp$.

The DEM must satisfy the following soundness property: for all $K \in \{0,1\}^\lambda$, for all $m \in \{0,1\}^*$,

$$\Pr[\mathsf{Decrypt}(K, \mathsf{Encrypt}(K, m)) = m] = 1\,.$$

2.3 Security Models

Security games for WIBEs, WIB-KEMs and DEMs are presented in Figure 1. In all four games, s is some state information and $\mathcal{O}$ denotes the oracles the adversary has access to. In the OW-WID game, $\mathcal{M}$ denotes the message space of the WIBE. This will depend on the system parameters.

IND-WID security game for WIBEs:

1. $(mpk, msk) \leftarrow \mathsf{Setup}$
2. $(P^*, m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}}(mpk)$
3. $b \xleftarrow{\$} \{0,1\}$
4. $C^* \leftarrow \mathsf{Encrypt}(mpk, P^*, m_b)$
5. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(C^*, s)$

OW-WID security game for WIBEs:

1. $(mpk, msk) \leftarrow \mathsf{Setup}$
2. $(P^*, s) \leftarrow \mathcal{A}_1^{\mathcal{O}}(mpk)$
3. $m \xleftarrow{\$} \mathcal{M}$
4. $C^* \leftarrow \mathsf{Encrypt}(mpk, P^*, m)$
5. $m' \leftarrow \mathcal{A}_2^{\mathcal{O}}(C^*, s)$

IND-WID security game for WIB-KEMs:

1. $(mpk, msk) \leftarrow \mathsf{Setup}$
2. $(P^*, s) \leftarrow \mathcal{A}_1^{\mathcal{O}}(mpk)$
3. $(K_0, C^*) \leftarrow \mathsf{Encap}(mpk, P^*)$
4. $K_1 \xleftarrow{\$} \{0,1\}^\lambda$
5. $b \xleftarrow{\$} \{0,1\}$
6. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(K_b, C^*, s)$

IND security game for DEMs:

1. $(m_0, m_1, s) \leftarrow \mathcal{A}_1()$
2. $K \xleftarrow{\$} \{0,1\}^\lambda$
3. $b \xleftarrow{\$} \{0,1\}$
4. $C^* \leftarrow \mathsf{Encrypt}(K, m_b)$
5. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(C^*, s)$

Fig. 1. Security games for WIBEs, WIB-KEMs and DEMs

Security of WIBEs In both WIBE security games, $\mathcal{A}$ has access to a private key extraction oracle, which given an identity ID outputs $d_{ID} \leftarrow \mathsf{KeyDer}(msk, ID)$. In the CCA model only, $\mathcal{A}$ also has access to a decryption oracle, which on input (C, ID), returns $m \leftarrow \mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C)$.

The adversary wins the IND-WID game if $b' = b$ and it never queried the key derivation oracle on any identity matching the pattern P^*. Furthermore, in the CCA model, the adversary must never query the decryption oracle on (C^*, ID), for any ID matching the pattern P^*. We define the advantage of the adversary as $\epsilon = |2\Pr[b' = b] - 1|$.

The adversary wins the OW-WID-CPA game if $m' = m$ and it never queried the key derivation oracle on any identity matching the pattern P^*. We define the advantage of the adversary to be $\epsilon = \Pr[m' = m]$.

Security of WIB-KEMs In the IND-WID game for WIB-KEMs, $\mathcal{A}$ has access to a private key extraction oracle, which given an identity ID outputs $d_{ID} \leftarrow \mathsf{KeyDer}(msk, ID)$. In the CCA model only, $\mathcal{A}$ also has access to a decapsulation oracle, which on input (C, ID), returns $K \leftarrow \mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C)$.

Again, the adversary wins the IND-WID game if $b' = b$ and it never queried the key derivation oracle on any identity matching the pattern P^*. Furthermore, in the CCA model, the adversary must never query the decapsulation oracle on (C^*, ID), for any ID matching the pattern P^*. We define the advantage of the adversary as $\epsilon = |2\Pr[b' = b] - 1|$.

Security of DEMs In the IND-CPA game for DEMs, the adversary has access to no oracles. In the IND-CCA model, $\mathcal{A}_2$ may call a decryption oracle, which on input $C \neq C^*$ returns $m \leftarrow \mathsf{Decrypt}(K, C)$. Note that this oracle is only available in the second phase of the attack. The adversary wins if $b' = b$. We define the advantage of the adversary as $\epsilon = |2\Pr[b' = b] - 1|$.

Definition 1 *A WIBE (resp. WIB-KEM) is (t, q_K, ϵ) IND-WID-CPA secure if all time t adversaries making at most q_K queries to the key derivation oracle have advantage at most ϵ in winning the IND-WID-CPA game described above.*

Definition 2 *A WIBE (resp. WIB-KEM) is* (t, q_K, q_D, ϵ) *IND-WID-CCA secure if all time* t *adversaries making at most* q_K *queries to the key derivation oracle and at most* q_D *queries to the decryption (resp. decapsulation) oracle have advantage at most* ϵ *in winning the IND-WID-CCA game described above.*

The (t, q_K, ϵ) IND-HID-CPA and (t, q_K, q_D, ϵ) IND-HID-CCA security of a HIBE and HIB-KEM are defined analogously.

Definition 3 *A WIBE is* (t, q_K, ϵ) *OW-WID-CPA secure if all time* t *adversaries making at most* q_K *queries to the key derivation oracle have advantage at most* ϵ *in winning the OW-WID-CPA game described above.*

Definition 4 *A DEM is* (t, q_D, ϵ) *IND-CCA secure if all time* t *adversaries making at most* q_D *decryption queries in the the IND-CCA game described above has advantage at most* ϵ*.*

In the random oracle model, the adversary has access to one or more random oracles. When working in this model, we will add the number of queries made to the oracle as a parameter, so for example we would say a WIBE is $(t, q_K, q_D, q_H, \epsilon)$ IND-WID-CCA secure, where q_H is the total number of hash queries. The other definitions may be adapted in a similar manner.

2.4 Tiered HIBEs and WIBEs

In a HIBE or WIBE, we often consider an identity sequence $(ID_1, ID_2, \ldots, ID_l)$ derived from a sequence of text delimited by special delimiter characters. For example,

$$\texttt{alice@cs.univ.edu} \mapsto (\texttt{edu}, \texttt{univ}, \texttt{cs}, \texttt{alice})$$

The characters `.` and `@` allow us to split up the text string into the hierarchy. However, there is a slight problem with this e-mail example. The two delimiters have slightly different meanings. Anything to the left of the `@` delimit denotes the group to which the e-mail identity belongs and anything to the right of the `@` delimiter denotes the identity of the member of the group. This could be a problem. Consider the two e-mail addresses:

$$\texttt{leonhard.euler@maths.berlin.edu} \mapsto (\texttt{edu}, \texttt{berlin}, \texttt{maths}, \texttt{euler}, \texttt{leonhard})$$
$$\texttt{leonhard@euler.maths.berlin.edu} \mapsto (\texttt{edu}, \texttt{berlin}, \texttt{maths}, \texttt{euler}, \texttt{leonhard})$$

Despite being separate e-mail addresses, these they have the same identity sequence in the hierarchy. The problem is compounded when one considers identity-based encryption with wildcards. It is unclear whether a message encrypted using the pattern $(\texttt{edu}, \texttt{berlin}, \texttt{maths}, \texttt{euler}, *)$ should be decrypted by anyone with an e-mail address on the server `euler.maths.berlin.edu` or by anyone with the surname `euler` with an e-mail address on the server `maths.berlin.edu`.

We solve this problem by introducing the concept of a *tiered* HIBE or WIBE. In a k-tier HIBE, a text string is mapped into k sequences of identities. Hence, a normal HIBE can be considered a 1-tier HIBE. The above e-mail example can be considered a 2-tier HIBE/WIBE. The first tier is the name of the server and the second tier is the name of the user with an e-mail account on that server. Hence,

$$\texttt{leonhard.euler@maths.berlin.edu} \mapsto ((\texttt{edu}, \texttt{berlin}, \texttt{maths}), (\texttt{euler}, \texttt{leonhard}))$$
$$\texttt{leonhard@euler.maths.berlin.edu} \mapsto ((\texttt{edu}, \texttt{berlin}, \texttt{maths}, \texttt{euler}), (\texttt{leonhard}))$$

It is clear that the two different e-mail addresses lead to two different tiered sequences of identities. For a particular decryption key, we may distinguish between two types of tier:

– A delegation tier. If a user has a decryption key for the identity sequence $(ID_1, ID_2, \ldots, ID_l)$ at a delegation tier, then they may deduce the decryption key for the identity sequence $(ID_1, ID_2, \ldots, ID_l, ID_{l+1})$ at that tier, where the same identity sequences are used at all other tiers and ID_{l+1} is any bit-string.
– A non-delegation tier. If a user has a decryption key for the identity sequence $(ID_1, ID_2, \ldots, ID_l)$ at a non-delegation tier, then they may not deduce the decryption key for the identity sequence $(ID_1, ID_2, \ldots, ID_l, ID_{l+1})$ at that tier, for any value of ID_{l+1}.

Boneh, Boyen and Goh [6] call this limited delegation. Any HIBE can be turned into a k-tier HIBE where the first $k-1$ tiers are non-delegatable and the k-th tier is delegatable using a suitable encoding scheme. Consider the tiered identity sequence:

$$\begin{array}{c}((ID_{1,1}, ID_{1,2}, \ldots, ID_{1,l(1)}),\\ (ID_{2,1}, ID_{2,2}, \ldots, ID_{2,l(2)}),\\ \ldots,\\ (ID_{k,1}, ID_{k,2}, \ldots, ID_{k,l(k)}))\end{array}$$

This maps onto the normal HIBE identity

$$([1]||ID_{1,1}, [1]||ID_{1,2}, \ldots, [1]||ID_{1,l(1)}, [2]||ID_{2,1}, [2]||ID_{2,2}, \ldots, [k]||ID_{k,l(k)})$$

where $[i]$ is the binary representation of i in $\lceil \log k \rceil$-bits. The corresponding encoding for WIBEs is

$$([1], ID_{1,1}, [1], ID_{1,2}, \ldots, [1], ID_{1,l(1)}, [2], ID_{2,1}, [2], ID_{2,2}, \ldots, [k], ID_{k,l(k)})$$

We also note that any Boneh-Boyen-Goh style scheme [6], including the Waters HIBE scheme [17], the Kiltz-Galindo KEM [12] and the Waters WIBE [1], can be turned into a k-tiered HIBE or WIBE. In such schemes, the public parameters contain one or more group elements u for each possible level of the HIBE. In a tiered HIBE or WIBE, the public parameters contain one or more group elements u for each combination of level and tier. Key derivation, encryption and decryption proceed in the logical manner. Furthermore, for all these schemes it is possible to derive keys that are either delegatable or non-delegatable at each tier [6].

3 Security of the Hybrid Construction

Suppose we have an IND-WID-CCA secure WIB-KEM $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap})$ and an IND-CCA secure DEM $(\mathsf{Encrypt}, \mathsf{Decrypt})$. Let us also suppose that the length λ of keys generated by the WIB-KEM is the same as the length of keys used by the DEM. Then, following the method of [10], we can combine them to form a WIBE $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}', \mathsf{Decrypt}')$ as follows:

– $\mathsf{Encrypt}'(mpk, P, m)$: Compute $(K, C_1) \leftarrow \mathsf{Encap}(mpk, P)$, $C_2 \leftarrow \mathsf{Encrypt}(K, m)$. Return $C = (C_1, C_2)$.
– $\mathsf{Decrypt}'(d_{ID}, C)$: Parse C as (C_1, C_2). If the parsing fails, return $\perp$. Otherwise, compute $K \leftarrow \mathsf{Decap}(d_{ID}, C_1)$. If Decap rejects, output $\perp$. Finally, compute $m \leftarrow \mathsf{Decrypt}(K, C_2)$, and output m.

Theorem 5 *Suppose there is a (t, q_K, q_D, ϵ)-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against IND-WID-CCA security of the hybrid WIBE. Then there is a $(t_{\mathcal{B}}, q_K, q_D, \epsilon_{\mathcal{B}})$-adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-WID-CCA security of the WIB-KEM and a $(t_{\mathcal{C}}, q_D, \epsilon_{\mathcal{C}})$-adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ against the IND-CCA security of the DEM such that:*

$$t_{\mathcal{B}} \leq t + q_D t_{\mathsf{Dec}} + t_{\mathsf{Enc}}$$
$$t_{\mathcal{C}} \leq t + q_D(t_{\mathsf{Dec}} + t_{\mathsf{Decap}} + t_{\mathsf{KeyDer}}) + q_K t_{\mathsf{KeyDer}} + t_{\mathsf{Encap}} + t_{\mathsf{Setup}}$$
$$\epsilon = \epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}}$$

where t_{Enc} *is the time to run the DEM's* Encrypt *algorithm,* t_{Dec} *is the time to run the DEM's* Decrypt *algorithm,* t_{Setup} *is the time to run* Setup*,* t_{Decap} *is the time to run* Decap *and* t_{KeyDer} *is the time to run* KeyDer*.*

The theorem and proof are straightforward generalisations to the WIBE case of those in [10]. The proof is given in the full version of the paper (and in Appendix A).

4 Generic CCA Secure WIB-KEMs in the Random Oracle Model

One approach to building systems secure against adaptive chosen ciphertext attacks is to first construct a primitive that is secure against passive attacks, and use some generic transformation to produce a system secure against the stronger adaptive attacks. We will apply a method proposed by Dent in [11] which converts an OW-CPA secure probabilistic encryption scheme into an IND-CCA KEM. We will use the same idea to convert an OW-WID-CPA secure WIBE scheme into an IND-WID-CCA secure WIB-KEM. Suppose we have an OW-WID-CPA secure probabilistic WIBE:

$$\mathcal{WIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$$

with message space $\mathcal{M}$. We will write $\mathsf{Encrypt}(mpk, P^*, m; r)$ to mean running the encryption algorithm with inputs (mpk, P^*, m) using a ρ-bit string of randomness r. We require that for all master keys mpk generated by Setup, all patterns P, all messages $m \in \mathcal{M}$ and all ciphertexts C:

$$|\{r \in \{0,1\}^\rho : \mathsf{Encrypt}(mpk, P, m; r) = C\}| \leq \gamma$$

where γ is a parameter of the scheme.

The only difficulty in applying the method of Dent [11] is that we must re-encrypt the recovered message as an integrity check. In the WIBE setting, this means we must know the pattern under which the message was original encrypted. Hence, we require an efficient algorithm W which on input $C = \mathsf{Encrypt}(mpk, P, m)$ where $P = (P_1, \ldots, P_l)$ is a pattern, returns the set $W = \{i \in \mathbb{Z} : P_i = *\}$. This is certainly possible with the Waters and BBG based WIBEs presented in [1]. If a scheme does not already have this property, it could be modified so that the set W is included explicitly as a ciphertext component. W can then be used to give an algorithm P, which on input (ID, C), where C is a ciphertext and $ID = (ID_1, \ldots, ID_l)$ is an identity, returns the pattern $P = (P_1, \ldots, P_l)$ given by $P_i = *$ for $i \in \mathsf{W}(C)$ and $P_i = ID_i$ otherwise.

We will use $\mathcal{WIBE}$ to construct an IND-WID-CCA secure WIB-KEM

$$\mathcal{WIBE}\text{-}\mathcal{KEM} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap})$$

using two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^\rho$ and $H_2 : \{0,1\}^* \to \{0,1\}^\lambda$, where λ is the length of keys output by the WIB-KEM. The Encap and Decap algorithms are given by:

- $\mathsf{Encap}(mpk, P)$: Choose a random message $m \xleftarrow{\$} \mathcal{M}$. Compute $r \leftarrow H_1(m)$, $K \leftarrow H_2(m)$ and compute $C \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$. Return (K, C)

– $\mathsf{Decap}(d_{ID}, C)$: Compute $m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$. If $m = \perp$, return $\perp$. Compute $r \leftarrow H_1(m)$, and $K \leftarrow H_2(m)$ and check that $C = \mathsf{Encrypt}(mpk, \mathsf{P}(ID, C), m; r)$. If so, return K, otherwise return $\perp$.

Theorem 6 *Suppose there is a $(t, q_K, q_D, q_H, \epsilon)$ adversary $\mathcal{A}$ against the IND-WID-CCA security of the WIB-KEM in the random oracle model. Then there is a $(t', q_K, q_H, \epsilon')$ adversary $\mathcal{B}$ against the OW-WID-CPA security of the WIBE, where:*

$$\epsilon' = (\epsilon - q_D\Big(\frac{1}{|\mathcal{M}|} + \gamma\Big))/(q_D + q_1 + q_2)$$
$$t' \leq t + q_H t_H + q_D q_H t_{Enc}$$

where t_H is the time to look up a hash value from the list, and t_{Enc} is the time taken to do an encryption.

This proof of this theorem is a straightforward generalisation of the result of Dent [11]. The proof is given in the full version of the paper (and in Appendix B).

5 A CCA Secure WIB-KEM without Random Oracles

5.1 The Kiltz-Galindo HIB-KEM

We present a construction for a WIB-KEM based on the Kiltz-Galindo HIB-KEM [12]. This construction is based on the Waters HIBE [17] and belongs to the Boneh-Boyen family of identity-based encryption schemes [5]. The construction presented in this section uses bilinear maps and target collision resistant hash functions. We briefly recall the definitions here:

Definition 7 (Bilinear map) *Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_{\mathrm{T}}$ be multiplicative groups of prime order p. We say that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathrm{T}}$ is an admissible bilinear map if the following hold true:*

- *For all $a, b \in \mathbb{Z}_p$ we have $e(g^a, g^b) = e(g, g)^{ab}$.*
- *$e(g, g)$ is not the identity element of $\mathbb{G}_{\mathrm{T}}$.*
- *e is efficiently computable.*

Definition 8 (Target collision resistant hash function) *A family $F_{\{k \in \mathcal{K}\}} : \mathbb{G} \rightarrow \mathbb{Z}_p$ of hash functions with key space $\mathcal{K}$ is called (t, ϵ)-target collision resistant if all time t algorithms $\mathcal{A}$ have advantage at most ϵ, where the advantage of an algorithm is defined by*

$$\Pr[x \neq y \wedge F_k(x) = F_k(y) : k \xleftarrow{\$} \mathcal{K}; x \xleftarrow{\$} \mathbb{G}; y \leftarrow \mathcal{A}(k, x)]$$

In principle, a key k for the hash function should be included as part of the public parameters, but to simplify the description of the scheme, we will treat the family of hash functions as if it were a fixed function.

We recall the Kiltz-Galindo HIB-KEM [12] in Figure 2. Note that the identities at each level are assumed to be n bits long i.e., $ID_i \in \{0,1\}^n$, and we set

$$[ID_i] = \{1 \leq j \leq n : \text{ the } j\text{-th of } ID_i \text{ is one}\}.$$

Furthermore, we let the function $h_1 : \mathbb{G} \rightarrow \mathbb{Z}_p^*$ denote a target collision resistant hash function. The security of the Kiltz-Galindo scheme rests on the bilinear decisional Diffie-Hellman (BDDH) problem.

Definition 1 (BDDH problem). *We say that the BDDH problem in $\mathbb{G}$ is (t, ϵ)-hard if*

$$\Big|\Pr\Big[\,\mathcal{A}(g^a, g^b, g^c, e(g,g)^{abc}) = 1 \;:\; a, b, c \xleftarrow{\$} \mathbb{Z}_p\,\Big] - \Pr\Big[\,\mathcal{A}(g^a, g^b, g^c, e(g,g)^{d}) = 1 \;:\; a, b, c, d \xleftarrow{\$} \mathbb{Z}_p\,\Big]\Big| \;\leq\; \epsilon$$

for any algorithm $\mathcal{A}$ running in time at most t.

Kiltz and Galindo proved the following security result of their scheme.

Theorem 9 *If the BDDH problem in $\mathbb{G}$ is (t', ϵ')-hard and the hash function h is (t_h, ϵ_h) target collision resistant, then the Kiltz-Galindo HIB-KEM is $(t, q_{\mathrm{K}}, q_{\mathrm{D}}, \epsilon)$ IND-HID-CCA secure, where $t = t' - O(\epsilon^{-2} \cdot \ln(\epsilon^{-1}) + q)$, $\epsilon' = O((nq)^L \cdot (\epsilon + q/p)) + \epsilon_h$ and $q = q_{\mathrm{K}} + q_{\mathrm{D}}$.*

Algorithm Setup:
$v_1, v_2, \alpha \xleftarrow{\$} \mathbb{G}$; $z \leftarrow e(g, \alpha)$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1 \ldots L, j = 0 \ldots n$
$mpk \leftarrow (v_1, v_2, u_{1,0}, \ldots, u_{L,n}, z)$
$msk \leftarrow \alpha$
Return (mpk, msk)

Algorithm $\mathsf{KeyDer}(d_{(ID_1,\ldots,ID_l)}, ID_{l+1})$:
Parse $d_{(ID_1,\ldots,ID_l)}$ as $(d_0, \ldots, d_l)$
$s_{l+1} \xleftarrow{\$} \mathbb{Z}_p^*$; $d'_{l+1} \leftarrow g^{s_{l+1}}$
$d'_0 \leftarrow d_0 \cdot \left(u_{l+1,0} \prod_{j \in ID_{l+1}} u_{l+1,j}\right)^{s_{l+1}}$
Return $(d'_0, d_1, \ldots, d_l, d'_{l+1})$

Algorithm $\mathsf{Encap}(mpk, ID)$:
Parse ID as $(ID_1, \ldots, ID_l)$
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $C_0 \leftarrow g^r$; $t \leftarrow h_1(C_0)$
For $i = 1 \ldots l$ do
$C_i \leftarrow \left(u_{i,0} \prod_{j \in [ID_i]} u_{i,j}\right)^r$
$C_{l+1} \leftarrow (v_1^t v_2)^r$
$K \leftarrow z^r$
Return $(K, (C_0, \ldots, C_{l+1}))$

Algorithm $\mathsf{Decap}(d_{(ID_1,\ldots,ID_l)}, C)$:
Parse $d_{(ID_1,\ldots,ID_l)}$ as $(d_0, \ldots, d_l)$
Parse C as $(C_0, \ldots, C_{l+1})$
$t \leftarrow h_1(C_0)$
If any of $(g,\ C_0,\ v_1^t v_2,\ C_{l+1})$
or $(g,\ C_0,\ u_{i,0} \prod_{j \in [ID_i]} u_{i,j},\ C_i)$, $i = 1 \ldots l$
is not a DH tuple then $K \leftarrow \perp$
else $K \leftarrow e(C_0, d_0) / \prod_{i=1}^{l} e(C_i, d_i)$
Return K

Fig. 2. The Kiltz-Galindo HIB-KEM scheme.

Note that the Kiltz-Galindo scheme generates keys which are elements of the group $\mathbb{G}_{\mathrm{T}}$, and we will follow this practice in our construction of the WIB-KEM. However, our definition of a WIB-KEM requires that the keys it generates are bitstrings. This discrepancy can be overcome by hashing the group element used as the key using a smooth hash function. A hash function $h : \mathbb{G}_{\mathrm{T}} \to \{0,1\}^\lambda$ is ϵ-smooth if for all $K \in \{0,1\}^\lambda$ and for all $z \in \mathbb{G}_{\mathrm{T}}^*$, the probability

$$Pr[h(z^r) = K : r \xleftarrow{\$} \mathbb{Z}_p] = 1/2^\lambda + \epsilon$$

5.2 The Kiltz-Galindo WIB-KEM

We attempt to build a WIB-KEM using a similar approach to that of Kiltz-Galindo [12] using the techniques of Abdalla *et al.* [1]. A naive implementation might try to construct an encapsulation algorithm as follows:

Algorithm $\mathsf{Encap}(mpk, P)$:
  Parse P as $(P_1, \ldots, P_l)$
  $r \xleftarrow{\$} \mathbb{Z}_p^*$; $C_0 \leftarrow g^r$; $t \leftarrow h_1(C_0)$
  For $i = 1 \ldots l$ do
    if $P_i \neq *$, then
      $C_i \leftarrow \left(u_{i,0} \prod_{j \in [P_i]} u_{i,j}\right)^r$
    else
      $C_i \leftarrow (u_{i,0}^r, \ldots, u_{i,n}^r)$
  $C_{l+1} \leftarrow (v_1^t v_2)^r$
  $K \leftarrow z^r$
  Return $(K, (C_0, \ldots, C_{l+1}))$

However, such an implementation would be insecure in the IND-WID-CCA model. An attacker could output a challenge pattern $P^* = (*)$ and would receive a key K and an encapsulation (C_0, C_1, C_2) where $C_0 = g^{r^*}$ and $C_1 = (u_0^{r^*}, \ldots, u_n^{r^*})$. It would be simple for the attacker then to construct a valid encapsulation of the same key for a particular identity ID by setting $C_1' \leftarrow u_0^{r^*} \prod_{j \in [ID]} u_i^{r^*}$. Thus, submitting the identity ID and the ciphertext (C_0, C_1', C_2) to the decryption oracle will return the correct decapsulation of the challenge.

This attack demonstrates the importance of knowing the location of the wildcards that were used to create an encapsulation. We solve this problem by using a *two tier* version of the Kiltz-Galindo HID-KEM. The first tier is used to encode the identity of the recipient as usual. The second tier is used to specify the locations of the wildcards in an encapsulation. Our scheme makes use of a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathrm{T}}$ and a target collision resistant hash function, $h_1 : \mathbb{G} \rightarrow \mathbb{Z}_p$. We will also make use of a function h_2, which on input of a pattern $P = (P_1, \ldots, P_l)$, returns a bitstring $b_1 b_2 \ldots b_l$, where $b_i = 1$ if P_i is a wildcard, otherwise $b_i = 0$. Note that two patterns P_1, P_2 have wildcards in the same location if and only if $h_2(P_1) = h_2(P_2)$.

- Setup : Pick random elements $v_1, v_2, v_3, \alpha \xleftarrow{\$} \mathbb{Z}_p$ and compute $z \leftarrow e(\alpha, g)$ where g is the generator of $\mathbb{G}$. Furthermore, pick elements $u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $1 \leq i \leq L$ and $0 \leq j \leq n$, and elements $w_j \xleftarrow{\$} \mathbb{G}$ for $0 \leq j \leq L$. The master public key is $mpk = (v_1, v_2, v_3, u_{1,0}, \ldots, u_{L,n}, w_0, \ldots, w_L, z)$ and the master secret is $msk = \alpha$.
- $\mathsf{KeyDer}(msk, ID_1)$: Pick $s_1 \xleftarrow{\$} \mathbb{Z}_p$. Compute $d_0 \leftarrow \alpha(u_{1,0} \prod_{j \in [ID_1]} u_{1,j})^{s_1}$ and $d_1 \leftarrow g^{s_1}$. The private key for ID_1 is (d_0, d_1). This can be thought of as an example of the next algorithm where the decryption key for the null identity is $d_0 \leftarrow \alpha$.
- $\mathsf{KeyDer}(d_{ID}, ID_{l+1})$: Parse the private key d_{ID} for $ID = (ID_1, \ldots, ID_l)$ as $(d_0, \ldots, d_l)$. Pick $s_{l+1} \xleftarrow{\$} \mathbb{Z}_p$ and compute $d'_{l+1} \xleftarrow{\$} g^{s_{l+1}}$. Lastly, compute

$$d'_0 \leftarrow d_0 \cdot \left(u_{l+1,0} \prod_{j \in [ID_{l+1}]} u_{l+1,j}\right)^{s_{l+1}}.$$

  The private key for $ID' = (ID_1, \ldots, ID_l, ID_{l+1})$ is $d_{ID'} = (d'_0, d_1, \ldots, d_l, d'_{l+1})$.
- $\mathsf{Encap}(mpk, P)$: Parse the pattern P as $(P_1, \ldots, P_l) \in (\{0,1\}^n \cup \{*\})^l$. Pick $r \xleftarrow{\$} \mathbb{Z}_p^*$, set $C_0 \leftarrow g^r$, and for $1 \leq i \leq l$ compute C_i as

$$C_i \leftarrow \begin{cases} \left(u_{i,0} \prod_{j \in [P_i]} u_{i,j}\right)^r & \text{if } P_i \neq * \\ (u_{i,0}^r, \ldots, u_{i,n}^r) & \text{if } P_i = * \end{cases}.$$

  Furthermore, set

$$C_{l+1} \leftarrow \left(w_0 \prod_{j \in [h_2(P)]} w_j\right)^r$$

If $P_i = *$ we will use the notation $C_{i,j}$ to mean the j^{th} component of C_i i.e. $u_{i,j}^r$. Finally, compute $t \leftarrow h_1(C_0)$, and $C_{l+2} \leftarrow (v_1^t v_2)^r$. The ciphertext $C = (C_0, \ldots, C_{l+2})$ is the encapsulation of key $K = z^r$.

– $\mathsf{Decap}(d_{ID}, C)$: Parse d_{ID} as $(d_0, \ldots, d_{l'})$ and C as $(C_0, \ldots, C_{l+2})$. First compute $t \leftarrow h_1(C_0)$ and $h_2(P)$, where P is the pattern under which C was encrypted. Note that $h_2(P)$ is implicitly given by C, even though P is not. Test whether

$$\begin{aligned}
&(g\ ,\ C_0\ ,\ v_1^t v_2\ ,\ C_{l+2}) \\
&(g\ ,\ C_0\ , w_0 \textstyle\prod_{j\in[h_2(P)]} w_j\ ,\ C_{l+1}) \\
&(g\ ,\ C_0\ ,\ u_{i,0} \textstyle\prod_{j\in[ID_i]} u_{i,j}\ ,\ C_i) \quad \text{for } 1 \le i \le l,\ P_i \neq * \\
&(g\ ,\ C_0\ ,\ u_{i,j}\ ,\ C_{i,j}) \quad \text{for } 1 \le i \le l,\ P_i = *,\ 0 \le j \le n
\end{aligned} \tag{1}$$

are all Diffie-Hellman tuples. If not, return $\perp$. Otherwise, decapsulate key by first setting

$$C_i' \leftarrow \begin{cases} C_i & \text{if } P_i \neq * \\ C_{i,0} \prod_{j\in[ID_i]} C_{i,j} & \text{if } P_i = * \end{cases} \quad \text{for } 1 \le i \le l'$$

and then computing $K \leftarrow\ e(C_0, d_0) / \prod_{i=1}^{l'} e(C_i', d_i)$.

Soundness. Given a correctly formed encapsulation $C = (C_0, \ldots, C_{l+2})$ of a key $K = z^r$ for a pattern P, it can be verified that decapsulation of C with a private key $d_{ID} = (d_0, \ldots, d_{l'})$ for $ID \in_* P$ yields the correct key since

$$\begin{aligned}
\frac{e(C_0, d_0)}{\prod_{i=1}^{l'} e(C_i', d_i)} &= \frac{e\big(g^r, \alpha \prod_{i=1}^{l'} \big(u_{i,0} \prod_{j\in[ID_i]} u_{i,j}\big)^{s_i}\big)}{\prod_{i=1}^{l'} e\big(\big(u_{i,0} \prod_{j\in[ID_i]} u_{i,j}\big)^r, g^{s_i}\big)} \\
&= \frac{e(g^r, \alpha) \prod_{i=1}^{l'} e\big(g^r, \big(u_{i,0} \prod_{j\in[ID_i]} u_{i,j}\big)^{s_i}\big)}{\prod_{i=1}^{l'} e\big(\big(u_{i,0} \prod_{j\in[ID_i]} u_{i,j}\big)^r, g^{s_i}\big)} \\
&= e(g, \alpha)^r \\
&= z^r
\end{aligned}$$

Thus the scheme is sound.

Theorem 10 *If the Kiltz-Galindo HIB-KEM is $(t, q_{\mathrm{K}}, q_{\mathrm{D}}, \epsilon)$ IND-HID-CCA secure then the WIB-KEM scheme described above is $(t', q_{\mathrm{K}}, q_{\mathrm{D}}, \epsilon')$ IND-WID-CCA secure, where $t' = t - O\big((nL + Lq_{\mathrm{K}} + q_{\mathrm{D}})t_{\mathsf{exp}} + nLq_{\mathrm{D}}t_{\mathsf{mul}}\big)$, $\epsilon' \ge \epsilon/2^L$ and t_{exp} and t_{mul} are the time it takes to compute an exponentiation and a multiplication in $\mathbb{G}$, respectively.*

Proof. The proof is by contradiction. Suppose that there is a $(t', q_{\mathrm{K}}, q_{\mathrm{D}}, \epsilon')$-adversary $\mathcal{A}$ against the IND-WID-CCA security of the scheme. We will use this adversary to construct a $(t, q_{\mathrm{K}}, q_{\mathrm{D}}, \epsilon)$-adversary $\mathcal{B}$ for the Kiltz-Galindo HIB-KEM. In the construction of $\mathcal{B}$, we will use similar ideas to those presented in [1].

Let $mpk = (v_1, v_2, u_{1,0}, \ldots, u_{L,n}, z)$ be the master public key given to $\mathcal{B}$ by a HIB-KEM challenger. Firstly, $\mathcal{B}$ will attempt to guess the positions of the wildcards in the challenge pattern P^* that $\mathcal{A}$ will produce, and randomly picks a set of integers $W \subseteq \{1, \ldots, L\}$, corresponding to the levels at which the wildcards appear. Based on this guess, $\mathcal{B}$ defines a "projection" that maps identities from the WIB-KEM to identities in the HIB-KEM. This projection will enable $\mathcal{B}$ to use its own oracles when responding to queries made by $\mathcal{A}$. Define the function $\pi(i) : \{1, \ldots, L\} \to \{0, \ldots, L\}$ as

$$\pi(i) = \begin{cases} 0 & \text{if } i \in W \\ i - |\{j\ :\ j \le i \text{ and } j \in W|\} & \text{if } i \notin W \end{cases}$$

Given an identity $ID' = (ID'_1, \ldots, ID'_l)$ in the WIB-KEM, $\mathcal{B}$ will "project" ID' to an identity ID in the HIB-KEM by setting $ID_{\pi(i)} \leftarrow ID'_i$ whenever $\pi(i) \neq 0$. This corresponds to removing all components ID'_i having $\pi(i) = 0$ (i.e. components at level i where $i \in W$). We will use the notation $ID = (ID'_i)_{\pi(i)\neq 0}$ to describe that ID is obtained from projecting ID' to the HIB-KEM. Besides constructing a projection, $\mathcal{B}$ computes a master public key $mpk' = (v_1, v_2, v_3, u'_{1,0}, \ldots, u'_{L,n}, z)$ for the WIB-KEM by reusing the values v_1, v_2 and z from mpk, setting $v_3 \leftarrow g^{\beta_v}$ where $\beta_v \xleftarrow{\$} \mathbb{Z}_p$ and for all $1 \leq i \leq L$ and $0 \leq j \leq n$ setting $u'_{i,j} \leftarrow g^{\beta_{i,j}}$ where $\beta_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ if $i \in W$ and $u'_{i,j} \leftarrow u_{i,j}$ otherwise. Finally, $\mathcal{B}$ runs $\mathcal{A}$ with mpk' as input.

Key derivation queries To make the notation easier to follow, we define the functions $H_i(ID_i) = u_{i,0} \prod_{j\in[ID_i]} u_{i,j}$ and $H'_i(ID'_i) = u'_{i,0} \prod_{j\in[ID'_i]} u'_{i,j}$ for all $1 \leq i \leq L$. Now, $\mathcal{B}$ responds to the queries as follows: On input $ID' = (ID'_1, \ldots, ID'_{l'})$, $\mathcal{B}$ constructs $ID = (ID'_i)_{\pi(i)\neq 0}$ as described above. Then $\mathcal{B}$ queries its HIB-KEM key derivation oracles with input ID to get a private key $d = (d_0, \ldots, d_l)$. From this key, a private key $d' = (d'_0, \ldots, d'_{l'})$ for ID' is computed as

$$d'_0 \leftarrow d_0 \prod_{\{i\,:\,i\in W\}} H'_i(ID'_i)^{r_i}\ , \quad d'_i \leftarrow \begin{cases} g^{r_i} & \text{if } i \in W \\ d_{\pi(i)} & \text{if } i \notin W \end{cases}$$

where $r_i \xleftarrow{\$} \mathbb{Z}_p$. It is easy to see that this will yield a correctly formed private key since $d'_i = g^{r_i}$ for all $1 \leq i \leq l'$ and

$$\begin{aligned} d'_0 &= \alpha \prod_{\{i\,:\,i\notin W\}} H_{\pi(i)}(ID'_i)^{r_i} \prod_{\{i\,:\,i\in W\}} H'_i(ID'_i)^{r_i} \\ &= \alpha \prod_{\{i\,:\,i\notin W\}} \Big(u_{\pi(i),0} \prod_{j\in[ID'_i]} u_{\pi(i),j}\Big)^{r_i} \prod_{\{i\,:\,i\in W\}} H'_i(ID'_i)^{r_i} \\ &= \alpha \prod_{\{i\,:\,i\notin W\}} \Big(u'_{i,0} \prod_{j\in[ID'_i]} u'_{i,j}\Big)^{r_i} \prod_{\{i\,:\,i\in W\}} H'_i(ID'_i)^{r_i} \\ &= \alpha \prod_{i=1}^{l'} H'_i(ID'_i)^{r_i} \end{aligned}$$

Decapsulation queries On input (ID', C'), where $C' = (C'_0, \ldots, C'_{l'+1})$ is an encapsulation for a pattern P, $\mathcal{B}$ checks that C' is a valid encapsulation as done in the Decap algorithm, shown in equation 1. If the encapsulation is invalid, it returns $\perp$, otherwise $\mathcal{B}$ constructs an encapsulation $C = (C_0, \ldots, C_{l+1})$ of the same key in the HIB-KEM. This is done by considering each element in C', discarding C'_i if $\pi(i) = 0$, and setting

$$C_0 \leftarrow C'_0,\ \ C_{\pi(i)} \leftarrow \begin{cases} C'_{i,0} \prod_{j\in[ID'_i]} C'_{i,j} & \text{if } P_i = * \\ C'_i & \text{if } P_i \neq * \end{cases} \quad \text{and} \quad C_{l+1} \leftarrow \frac{C'_{l'+1}}{(C_0)^{\beta_v h_2(P)}}\ .$$

The length of C is $l + 1 = l' + 1 - |W|$. Note that the position of the wildcards can easily be determined from C', and this is all the information about P that is needed. Let z^r be the key encapsulated by C'. C is known to be a valid encapsulation of z^r for the identity $ID = (ID')_{\pi(i)\neq 0}$, since $C_0 = C'_0 = g^r$,

$$C_{l+1} = \frac{\big(v_1^{h_1(C_0)} v_3^{h_2(S)} v_2\big)^r}{(g^r)^{\beta_v h_2(S)}} = \frac{\big(v_1^{h_1(C_0)} v_3^{h_2(S)} v_2\big)^r}{v_3^{r h_2(S)}} = \big(v_1^{h_1(C_0)} v_2\big)^r \quad \text{and}$$

$$C_{\pi(i)} = \Big(u'_{i,0} \prod_{j\in[ID'_i]} u'_{i,j}\Big)^r = \Big(u_{\pi(i),0} \prod_{j\in ID_{\pi(i)}} u_{\pi(i),j}\Big)^r = H\big(ID_{\pi(i)}\big)^r.$$

Now $\mathcal{B}$ simply submits (ID, C) to its own HIB-KEM decryption oracle to obtain z^r, which it then forwards to $\mathcal{A}$.

At the end of the first phase of the simulation, $\mathcal{A}$ returns a challenge pattern P^*. If W, chosen by $\mathcal{B}$ in the beginning of the simulation, does not describe the position

of the wildcards in P^*, $\mathcal{B}$ aborts. Otherwise, the projection of P^* will correspond to an identity $ID^* = (P_i^*)_{\pi(i)\neq 0}$ in the HIB-KEM. $\mathcal{B}$ returns this identity ID^* as its own challenge identity in the HIB-KEM security game. $\mathcal{B}$'s challenger will respond with (K_b, C^*) where b is a secret random bit chosen by the challenger, C^* is the encapsulation of K_0, and K_1 is a random key. Let $K_0 = z^r$ for a value r known only by $\mathcal{B}$'s challenger. From $C^* = (C_0^*, \ldots, C_{l+1}^*)$, $\mathcal{B}$ constructs an encapsulation of z^r for P^* in the WIB-KEM by setting

$$C_0'^* \leftarrow C_0^*, \;\; C_i'^* \leftarrow \begin{cases} C_{\pi(i)}^* & \text{if } i \notin W \\ \left((C_0^*)^{\beta_{i,0}}, \ldots, (C_0^*)^{\beta_{i,n}}\right) & \text{if } i \in W \end{cases} \quad \text{and} \;\; C_{l'+1}'^* \leftarrow C_{l+1}^* (C_0^*)^{\beta_v h_2(P)}.$$

Since $C_0^* = g^r$, $(C_0^*)^{\beta_v} = v_3^r$ and $(C_0^*)^{\beta_{i,j}} = (u'_{i,j})^r$, it is easy to see that C'^* is a valid encapsulation of z^r for the pattern P^* in the WIB-KEM. After computing C'^*, $\mathcal{B}$ submits (K_b, C'^*) to $\mathcal{A}$ as a response to the challenge pattern P^*.

During the second phase of the simulation, $\mathcal{A}$ can make key derivation queries for identities $ID \notin_* P^*$ and decapsulation queries for any pair (ID', C') other than (ID, C^*) for some $ID \in_* P^*$. $\mathcal{B}$ will respond to these queries as in the first phase. At the end of the second phase, $\mathcal{A}$ will return a bit b' which is a guess on b in the challenge (K_b, C'^*). $\mathcal{B}$ simply forwards b' to the HIB-KEM challenger as its own guess on b.

Note that if $\mathcal{A}$ makes a decapsulation query (ID', C') which is mapped onto the challenge identity and encapsulation (ID^*, C^*) in the HIB-KEM, then it could potentially correspond to a query which is legal in the WIB-KEM but not in the HIB-KEM. However, since the WIB-KEM encapsulation is valid and $\mathcal{B}$ checks that $C'_{l'+1}$ correctly encodes the location of the wildcards, we know that C' must have wildcards in the same locations as C^*, and hence they are equal. Thus, such a query is also illegal in the WIB-KEM.

This completes the description of the simulation. It remains to analyse the success probability of $\mathcal{B}$. Let E_1 be the event that $\mathcal{B}$ aborts because of an incorrect guess of W, and let $S_{\mathcal{A}}$ be the event that $\mathcal{A}$ correctly guesses the value of b. We have that

$$\begin{aligned} \Pr[\mathcal{B} \text{ succeeds}] &= \Pr[\neg E_1 \wedge S_{\mathcal{A}}] \\ &= \Pr[\neg E_1] \Pr[S_{\mathcal{A}} | \neg E_1] \end{aligned}$$

Since W is chosen independently and at random, we have $\Pr[\neg E_1] = 1/2^L$. Lastly, since $\mathcal{B}$ provides a perfect simulation if it does not abort, we have that $\Pr[S_{\mathcal{A}}|\neg E_1] = \epsilon'$ and hence

$$\Pr[\mathcal{B} \text{ succeeds}] = \epsilon \geq \frac{\epsilon'}{2^L}.$$

Thus, the theorem follows. □

Note that, as is the case for all known HIBE and WIBE schemes, the security of our WIB-KEM degrades exponentially with the maximal hierarchy depth L. The scheme can therefore only be used for relatively small (logarithmic) values of L. We leave the construction of a WIBE-KEM with polynomial efficiency and security in all parameters as an open problem. Any solution to this problem would directly imply a WIBE and a HIBE scheme with polynomial security as well, the latter of which has been an open problem for quite a while now.

6 Conclusion

We have extended the concept of WIBE to Key Encapsulation, and used this idea to provide methods to construct CCA secure WIBE schemes, in both the random oracle and standard models, which are much more efficient than the earlier constructions.

References

1. Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In *ICALP (2)*, pages 300–311, 2006.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.
3. Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005. http://eprint.iacr.org/.
4. Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 289–299, Santa Barbara, CA, USA, August 19–23, 1984. Springer-Verlag, Berlin, Germany.
5. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer-Verlag, Berlin, Germany.
6. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.
7. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.
8. Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103, San Francisco, CA, USA, February 14–18, 2005. Springer-Verlag, Berlin, Germany.
9. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer-Verlag, Berlin, Germany.
10. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33:167–226, 2004.
11. Alexander W. Dent. A designer's guide to KEMs. In K. Paterson, editor, *Cryptography and Coding: 9th IMA International Conference*, volume 2898 of *LNCS*, pages 133–151. Springer-Verlag, Berlin, Germany, 2003.
12. Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In *ACISP*, pages 336–347, 2006.
13. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.
14. Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.
15. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer-Verlag, Berlin, Germany.
16. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004.
17. Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

A Proof of security for the hybrid construction

We first restate the theorem of Section 3:

Theorem 11 *Suppose there is a (t, q_K, q_D, ϵ)-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against IND-WID-CCA security of the hybrid WIBE. Then there is a $(t_{\mathcal{B}}, q_K, q_D, \epsilon_{\mathcal{B}})$-adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-WID-CCA security of the WIB-KEM and a $(t_{\mathcal{C}}, q_D, \epsilon_{\mathcal{C}})$-adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ against the IND-CCA security of the DEM such that:*

$$
\begin{aligned}
t_{\mathcal{B}} &\leq t + q_D t_{\mathsf{Dec}} + t_{\mathsf{Enc}} \\
t_{\mathcal{C}} &\leq t + q_D(t_{\mathsf{Dec}} + t_{\mathsf{Decap}} + t_{\mathsf{KeyDer}}) + q_K t_{\mathsf{KeyDer}} + t_{\mathsf{Encap}} + t_{\mathsf{Setup}} \\
\epsilon &= \epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}}
\end{aligned}
$$

where t_{Enc} is the time to run the DEM's Encrypt *algorithm, t_{Dec} is the time to run the DEM's* Decrypt *algorithm, t_{Setup} is the time to run* Setup*, t_{Decap} is the time to run* Decap *and t_{KeyDer} is the time to run* KeyDer*.*

Proof. The proof is structured as a sequence of games. Let Game 1 be the original game played by $\mathcal{A}$ against the WIBE.

If we write the operations of the hybrid scheme out in full, the game is as follows:

1. $(mpk, msk) \leftarrow \mathsf{Setup}$
2. $(P^*, m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}}(mpk)$
3. $b \xleftarrow{\$} \{0,1\}$
4. $(K^*, C_1^*) \leftarrow \mathsf{Encap}(mpk, P^*)$
5. $C_2^* \leftarrow \mathsf{Encrypt}(K^*, m_b)$
6. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}((C_1^*, C_2^*), s)$

In the above, $\mathcal{O}$ represents the oracles that $\mathcal{A}$ is given access to. Since we are working in the CCA model, these are the key derivation oracle, which on input ID returns $\mathsf{KeyDer}(msk, ID)$, and the decryption oracle, which on input $(ID, (C_1, C_2))$ returns

$$\mathsf{Decrypt}(\mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C_1), C_2)\,.$$

$\mathcal{A}$ wins if both $b' = b$, and it never requested the decryption key for any identity ID matching the pattern P^* or queried the decryption oracle on $(ID, (C_1^*, C_2^*))$. Let S_1 be the event that $\mathcal{A}$ wins Game 1.

We now define a modified game, Game 2, as follows:

1. $(mpk, msk) \leftarrow \mathsf{Setup}$
2. $(P^*, m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}}(mpk)$
3. $b \xleftarrow{\$} \{0,1\}$
4. $(K, C_1^*) \leftarrow \mathsf{Encap}(mpk, P^*)$
5. $K^* \xleftarrow{\$} \{0,1\}^{\lambda}$
6. $C_2^* \leftarrow \mathsf{Encrypt}(K^*, m_b)$
7. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}((C_1^*, C_2^*), s)$

The decryption oracle is modified so that in the second phase, after the challenge ciphertext (C_1^*, C_2^*) has been issued, if it is queried on $(ID, (C_1^*, C_2))$, where $C_2 \neq C_2^*$, and ID is any identity matching the pattern P^*, then it simply returns $\mathsf{Decrypt}(K^*, C_2)$. Let S_2 be the event that $\mathcal{A}$ wins Game 2.

We now describe the $(t_{\mathcal{B}}, q_K, q_D, \epsilon_{\mathcal{B}})$-adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-WID-CCA security of the WIB-KEM. $\mathcal{B}_1$ takes a master public key mpk and runs $\mathcal{A}_1(mpk)$ which outputs (P^*, m_0, m_1, s). It sets $s' = (P^*, m_0, m_1, s)$ and outputs (P^*, s').

$\mathcal{B}_2$ receives $(K^*, C_1^*, (P^*, m_0, m_1, s))$ as input and then chooses a random bit $d \xleftarrow{\$} \{0,1\}$. It then computes $C_2^* \leftarrow \mathsf{Encrypt}(K^*, m_d)$ and runs $\mathcal{A}_2((C_1^*, C_2^*), s)$, which outputs a bit d'. If $d' = d$, $\mathcal{B}_2$ outputs 0, otherwise it outputs 1.

Key Derivation Queries: To respond to $\mathcal{A}$'s key derivation queries, $\mathcal{B}$ simply forwards the query to its own key derivation oracle.
Decryption Queries If $\mathcal{A}$ makes a decryption query on $(ID, (C_1, C_2))$, $\mathcal{B}$ queries its decapsulation oracle on (ID, C_1) and obtains a key K. If $K = \perp$, $\mathcal{B}$ returns $\perp$, otherwise it returns $m \leftarrow \mathsf{Decrypt}(K, C_2)$. In the second phase, $\mathcal{B}_2$ responds as before, except if it is queried on $(ID, (C_1^*, C_2))$ for any $ID \in_* P^*$ and $C_2 \neq C_2^*$, it returns $\mathsf{Decrypt}(K^*, C_2)$.

It is clear that if $\mathcal{B}$'s challenger chooses bit $b = 0$, then the key K^* is the correct key encapsulated in C_1^*, so $\mathcal{A}$'s view of the game is exactly as in Game 1. This implies that

$$\Pr[S_1] = \Pr[d' = d | b = 0] = \Pr[b' = 0 | b = 0] \,.$$

Similarly, if the challenger chooses bit $b = 1$, then the key K^* is chosen at random, so $\mathcal{A}$'s view of the game is exactly as in Game 2. So

$$\Pr[S_2] = \Pr[d' = d | b = 1] = \Pr[b' = 0 | b = 1]$$

Combining these results and noting that

$$\epsilon_{\mathcal{B}} = |2 \Pr[b' = b] - 1| = |\Pr[b' = 0 | b = 0] - \Pr[b' = 0 | b = 1]$$

we get that

$$|\Pr[S_2] - \Pr[S_1]| = \epsilon_{\mathcal{B}} \,.$$

Running Time $\mathcal{B}$ runs $\mathcal{A}$, performs one DEM decryption per decryption query that $\mathcal{A}$ makes, and performs one DEM encryption for the challenge so $\mathcal{B}$ runs in time

$$t' \leq t + q_D t_{\mathsf{Dec}} + t_{\mathsf{Enc}} \,.$$

Finally, there is a $(t_{\mathcal{C}}, q_D, \epsilon_{\mathcal{C}})$-adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ against the IND-CCA security of the DEM such that

$$|\Pr[S_2]| = \epsilon_{\mathcal{C}} \,.$$

$\mathcal{C}_1$ generates $(mpk, msk) \leftarrow \mathsf{Setup}$. It then runs $\mathcal{A}_1(mpk)$ which outputs a tuple (P^*, m_0, m_1, s). It sets $s' = (P^*, mpk, msk, s)$ and outputs (m_0, m_1, s'). $\mathcal{C}_2$ receives (C_2^*, s') from the challenger, parses s' as (P^*, mpk, msk, s) and computes $(K, C_1^*) \leftarrow \mathsf{Encap}(mpk, P^*)$. Finally, it runs $\mathcal{A}_2((C_1^*, C_2^*), s)$ which outputs b', and $\mathcal{C}_2$ outputs b'.

Key Derivation Queries To respond to $\mathcal{A}$'s key derivation queries, $\mathcal{C}$ simply uses the KeyDer algorithm and the master secret key which it knows.
Decryption Queries If $\mathcal{A}$ makes a decryption query on $(ID, (C_1, C_2))$ for some $C_1 \neq C_2^*$, $\mathcal{B}$ computes $\mathsf{Decrypt}(\mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C_1), C_2)$. In the second phase, it responds to queries where $C_1 = C_1^*$ by passing C_2 to it's own decryption oracle and returning the result.

$\mathcal{A}$'s view of this simulation is identical to Game 2, since the key used by the IND-CCA challenger is randomly chosen and unrelated to the encapsulation C_1^*, so

$$|\Pr[S_2] - \frac{1}{2}| = |\Pr[b' = b] - \frac{1}{2}| = \epsilon_{\mathcal{C}} \,.$$

Running Time $\mathcal{C}$ runs Setup, runs $\mathcal{A}$, performs one KEM encapsulation in the challenge phase and performs q_K key derivation operations, and q_D decryptions, decapsulations and key derivations. So $\mathcal{C}$ runs in time

$$t_{\mathcal{C}} \leq t + q_D(t_{\mathsf{Dec}} + t_{\mathsf{Decap}} + t_{\mathsf{KeyDer}}) + q_K t_{\mathsf{KeyDer}} + t_{\mathsf{Encap}} + t_{\mathsf{Setup}} \,.$$

Combining these results, we get

$$\epsilon = \epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} .$$

□

B Proof of security for the generic construction

We will prove this using a sequence of games in the manner of [16]. In particular, we will need the following lemma:

Lemma 1 (Difference Lemma). *Let A, B and F be events, and suppose that $A \wedge \neg F$ is equivalent to $B \wedge \neg F$. Then* $\Pr[A] - \Pr[B] \leq \Pr[F]$.

The lemma is proved in [16].

Proof (Proof of Theorem 6).

Let Game 1 be the original attack game against the WIB-KEM. We define a modified game, Game 2, which is the same as Game 1, but in Game 2, the adversary may not query the Decap oracle on (ID, C^*) for any identity $ID \in_* P^*$ at any time. In the second phase this is already forbidden, but we must consider the possibility that it makes such a query in the first phase.

Let E_1 be the event that $\mathcal{A}$ queries the decapsulation oracle on the challenge encapsulation in the first phase. Then $\Pr[E_1] \leq q_D/|\mathcal{M}|$. This follows since each message has exactly one valid encapsulation for a given pattern, and in the first phase the adversary has no information about the challenge encapsulation.

By the difference lemma, the advantage of $\mathcal{A}$ in Game 2 is at least

$$\epsilon_2 \geq \epsilon - \frac{q_D}{|\mathcal{M}|} .$$

We now define a modified game, Game 3, which is the same as Game 2, but we respond to the oracle queries as follows:

- Hash queries: The H_1 and H_2 oracles are simulated by making use of two lists, H_1-list and H_2-list, which are initially empty. To respond to the adversary's query $H_i(x)$, we first check if there is a pair (x, h) in the H_i-list. If so, we return h, otherwise we query $h \leftarrow H_i(x)$ to the real oracle, append (x, h) to the H_i-list and return h.
- KeyDer queries: These are handled as in the original game.
- Decap queries: To respond to a decapsulation query on (ID, C), we look for a pair (m, h) in the H_1-list which satisfies $\mathsf{Encrypt}(mpk, \mathsf{P}(ID, C), m; h) = C$. If one exists, we compute $K \leftarrow H_2(m)$ using the method described above and return K. Otherwise we return $\perp$.

Game 3 proceeds exactly as Game 2 unless the following happens: Let E_2 be the event that $\mathcal{A}$ makes a decapsulation query on (ID, C) such that $m = \mathsf{Decrypt}(d_{ID}, C)$ and $C = \mathsf{Encrypt}(mpk, \mathsf{P}(ID, C), m; H_1(m))$, but $\mathcal{A}$ has not yet queried H_1 on m. $\Pr[E_2] \leq q_D\gamma$ by definition of γ. Thus the advantage of $\mathcal{A}$ in Game 2 is at least

$$\epsilon_3 = \epsilon_2 - \gamma q_D .$$

Let E_3 be the event that $\mathcal{A}$ queries H_1 or H_2 on $m = \mathsf{Decrypt}(d_{ID}, C^*)$, for some $ID \in P^*$. Since $\mathcal{A}$ has advantage ϵ_3, it must make this query with probability at least ϵ_3.

We now construct an OW-WID-CPA adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ using $\mathcal{A}$ as an oracle. $\mathcal{B}$ handles all oracle queries as in Game 3 – passing key derivation queries to

its own oracle and using H_1 and H_2-lists to answer decryption queries. Note that the simulation may add an entry to the H_2-list whenever $\mathcal{A}$ makes an H_2 oracle query. Hence, the size of the H_2-list is bounded by $q_{H_2} + q_D$. The simulation only adds an entry to the H_1-list when a H_1 oracle query is made; hence, the size of the H_1 list is bounded by q_{H_1}.

$\mathcal{B}_1$ simply takes a master public key mpk, and runs $\mathcal{A}_1(mpk)$. $\mathcal{A}$ returns (P^*, s), which $\mathcal{B}$ simply returns to its own challenger.

$\mathcal{B}_2$ takes (C^*, s), generates a random bitstring $K^* \xleftarrow{\$} \{0,1\}^\lambda$ and runs $\mathcal{A}_2$ on $((K^*, C^*), s)$. When $\mathcal{A}_2$ terminates, $\mathcal{B}$ chooses a random message m from either the H_1-list or H_2-list and returns this as its guess for the challenge message.

$\mathcal{B}$ simulates the environment of Game 3 exactly, at least until $\mathcal{A}$ queries H_1 or H_2 on $m = \mathsf{Decrypt}(d_{ID}, C^*)$. Since $\mathcal{A}$ cannot detect this difference without making one of these oracle queries, it must still make one with probability ϵ_3 as before, and $\mathcal{B}$ wins if it then chooses the correct value of m from the list.

$\mathcal{B}$ must perform one hash list lookup for each hash query made by $\mathcal{A}$, while for each decryption query, it must perform at most q_H encryptions to find the corresponding entry in the hash list, so it's running time is $t' + q_H t_H + q_D q_H t_{Enc}$ as claimed.

The claim now follows. □