Efficient Chosen-Ciphertext Secure Identity-Based Encryption with Wildcards

James Birkett[1], Alexander W. Dent[1], Gregory Neven[2], and Jacob C. N. Schuldt[1,3]

[1] Information Security Group,
Royal Holloway, University of London,
Egham, TW20 0EX, UK.
{j.m.birkett,a.dent}@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
[3] Institute of Industrial Science, University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan.
schuldt@iis.u-tokyo.ac

Abstract. We propose new instantiations of chosen-ciphertext secure identity-based encryption schemes with wildcards (WIBE). Our schemes outperform all existing alternatives in terms of efficiency as well as security. We achieve these results by extending the hybrid encryption (KEM–DEM) framework to the case of WIBE schemes. We propose and prove secure one generic construction in the random oracle model, and one 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 [19, 18, 10, 8] 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 private 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. This extension therefore provides a very intuitive interface for identity-based mailing lists.

Arbitrary-length plaintexts. 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

This paper will appear in J. Pieprzyk, H. Ghodosi, and E. Dawson. (Editors): ACISP 2007, volume 4586 of Lecture Notes in Computer Science, pages 274–292, 2007.

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 [11]. One would intuitively expect these results to extend to the case of WIBEs, but this was never formally shown to be the case.

Chosen-ciphertext security. The basic schemes of [1] are proved secure under an appropriate adaptation of indistinguishability (IND) under chosen-plaintext attack (CPA) [13], where the adversary is given access to a key derivation oracle and has to distinguish between encryptions of two messages of its choice. This security notion is often not considered sufficient for practise though. Rather, the community seems to have settled with the stronger notion of indistinguishability under chosen-ciphertext attack (CCA) [16] as the "right" security notion for practical use. The need for chosen-ciphertext security in practise was shown by Bleichenbacher's attack [21] on the SSL key establishment protocol, which was based on the (CPA-secure) RSA-PKCS#1 version 1 [22] encryption standard. The practical appreciation for the notion is exemplified by the adoption of the (CCA-secure) RSA-OAEP encryption scheme [23] in version 2 of the RSA-PKCS#1 standard.

A generic construction. Canetti *et al.* [9] proposed a generic construction of a CCA-secure hierarchical identity-based encryption (HIBE) scheme with up to L hierarchy levels from any $(L+1)$-level CPA-secure HIBE scheme and any one-time signature scheme. Abdalla *et al.* adapted their techniques to the WIBE setting, but their construction requires a $(2L+2)$-level CPA-secure WIBE scheme to obtain an L-level CCA-secure one. (The reason is that the construction of [9] prefixes a bit to identity strings indicating whether it is a real identity or a public key of the one-time signature scheme. In the case of WIBE schemes, these bits must be put on separate levels, because if not the simulator may need to make illegal key derivation queries to answer the adversary's decryption queries.)

Doubling the hierarchy depth has a dramatic impact on efficiency and security of the schemes. First, the efficiency of all known WIBE schemes (in terms of computation, key length, and ciphertext length) is linear in the hierarchy depth, so the switch to CCA-security essentially doubles most associated costs. Second, the security of all known WIBE schemes degrades exponentially with the maximal hierarchy depth L. If the value of L is doubled, then either the scheme is restricted to half the (already limited) number of "useful" hierarchy levels, or that the security parameter must be increased to restore security. The first measure seriously limits the functionality of the scheme, the second increases costs even further.

For example, the WIBE scheme from [1] based on Waters' HIBE scheme [20] loses a factor of $(2nq_{\mathrm{K}})^L$ in the reduction to the BDDH problem, where n is the bit length of an identity string at each level of the hierarchy and q_{K} is the number of adversarial key derivation queries. Assume for simplicity that the advantage of solving the BDDH problem in a group of order $p > 2^k$ is $2^{-k/2}$. If $n = 128$ and $q_{\mathrm{K}} = 2^{20}$, then to limit an adversary's advantage to 2^{-80} in a WIBE scheme with $L = 5$ levels, one should use a group order of at least $160 + 56L = 440$ bits. In the CCA-secure construction however, one needs a group order of $160+56(2L+2) = 832$ bits, almost doubling the size of the representation of a group element, and multiplying by eight the cost of most (cubic-time) algorithms! Furthermore, since there are twice as many levels, the ciphertext must contain twice as many group elements, so overall, ciphertexts are four times as large and the cost of encryption and decryption is multiplied by sixteen!

Our contributions. In this paper, we provide formal support for the use of hybrid encryption with WIBE schemes, and we present CCA-secure schemes that are more

Scheme	$\lvert mpk\rvert$	$\lvert d\rvert$	$\lvert C\rvert$	Encap	Decap	Security loss
$\mathcal{2\text{-}L}(\mathcal{BB})$	$4L+7$	$2L+1$	$3L+2$	$3L+2$	$2L+1$	q_{H}^{2L+2}
$\mathcal{OW}(\mathcal{BB})$	$2L+3$	$L+1$	$2L+2$	$2L+2$	$L+1$	q_{H}^{L}
$\mathcal{2\text{-}L}(\mathcal{BBG})$	$2L+6$	$2L$	$L+3$	$L+3$	2	q_{H}^{2L+2}
$\mathcal{OW}(\mathcal{BBG})$	$L+4$	$L+1$	$L+3$	$L+3$	2	q_{H}^{L}
$\mathcal{2\text{-}L}(\mathcal{W}a)$	$(n+3)L+3$	$2L+1$	$(n+2)L+2$	$(n+2)L+2$	$2L+1$	$(2nq_{\mathrm{K}})^{2L+2}$
no-$\mathcal{RO}$	$(n+1)L+3$	$L+1$	$(n+1)L+2$	$(n+1)L+2$	$L+3$	$L(20(n+1)q_{\mathrm{K}})^{L}$

Fig. 1. Efficiency comparison between our CCA-secure schemes and those of [1]. The $\mathcal{BB}$, $\mathcal{BBG}$ and $\mathcal{W}a$ schemes are the WIBE schemes based on [5, 7, 20] presented in [1]. The no-$\mathcal{RO}$ scheme is our direct construction without random oracles. The $2\text{-}\mathcal{L}(\cdot)$ transformation refers to the generic CCA-secure construction of [1]; the $\mathcal{OW}(\cdot)$ transformation is our random-oracle based construction. We compare the schemes in terms of master public key size ($\lvert mpk\rvert$), user secret key size ($\lvert d\rvert$), ciphertext size ($\lvert C\rvert$), key encapsulation time (Encap), key decapsulation time (Decap), and the factor lost in the security reduction to the underlying assumption. The given values refer to the number of group elements for $\lvert mpk\rvert$, $\lvert d\rvert$, $\lvert C\rvert$; to the number of exponentiations for Encap; and to the number of pairing computations for Decap. L is the maximal hierarchy depth and n is the bit length of (a collision-resistant hash of) an identity string. The values q_{H}, q_{K} and q_{D} refer to the number of queries of an adversary to the random oracle, key derivation oracle and decryption oracle, respectively.

efficient and secure than those obtained through the generic construction of [1]. We achieve these results by considering WIBE schemes as consisting of separate key and data encapsulation mechanisms (KEM–DEM) [11], leading to the definition of identity-based key encapsulation mechanisms with wildcards (WIB-KEM). Here, the WIB-KEM encrypts a random key under a (wildcarded) identity, while the DEM encrypts the actual data under this random key.

We first show that the combination of a CPA-secure (resp. CCA-secure) WIB-KEM with a CPA-secure (resp. CCA-secure) DEM indeed yields a CPA-secure (resp. CCA-secure) WIBE scheme. This result may be rather unsurprising, but needed proof: it 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 [11] and identity-based [3] cases. Furthermore, it should be noted that subtleties can arise in the proving of such results, for example in the case of certificateless KEMs [3].

Obviously, any secure WIBE scheme can be used to instantiate the WIB-KEM in the hybrid construction. (If the WIBE securely encrypts arbitrary messages, it also securely encrypts random keys.) This solves the problem of encrypting arbitrary-length messages, but still requires a CCA-secure WIBE scheme to achieve chosen-ciphertext security. As we argued above, all known instantiations of such schemes suffer from efficiency problems due to the doubling of the number of levels.

We therefore present a generic construction of L-level CCA-secure WIB-KEMs in the random oracle model [2] along the lines of Dent [12] from any L-level WIBE scheme that is one-way (OW) secure under chosen-plaintext attack. One-wayness is a much weaker security requirement than CCA-security, allowing much more efficient instantiations. In particular, one-wayness is implied by indistinguishability (for sufficiently large message spaces), so we can use any of the IND-CPA secure constructions of [1]. We also note that this construction can also be used to build CCA-secure HIBE schemes.

The resulting efficiency gains are summarised in Fig. 1. Abdalla *et al.* present two efficient schemes in the random oracle model based on the HIBE schemes of [5, 7]. One can see that our schemes perform significantly better in terms of key sizes, ciphertext length, and encapsulation and decapsulation times. When taking into account the security loss, one either has to conclude that our scheme supports twice the hierarchy depth, or that the inefficiency of the existing schemes in terms of memory size and computation time is blown up by a factor of at least two and eight, respectively.

Finally, we present a direct construction of a WIB-KEM scheme in the standard (i.e., non-random-oracle) model based on the HIB-KEM scheme by Kiltz and Galindo [15], which on its turn is based on Waters' HIBE scheme [20]. Note that the original version of the Kiltz-Galindo HIB-KEM scheme [14] is insecure, a fact which was noticed in [17], but the updated scheme in [15] does not suffer from the same weakness. We compare our scheme's efficiency to that of the only standard-model CCA-secure scheme in [1], namely the scheme obtained by applying their generic CCA transformation to the WIBE scheme based on Waters' HIBE. For fair comparison, we consider the optimised variant suggested in the full version of [1] that takes advantage of the fact that intermediate levels only contain one-bit identities. Our scheme is twice as efficient as the non-random-oracle scheme of [1] in terms of secret key size and pairing computations during decapsulation. The difference with regard to ciphertext size and encapsulation time is less pronounced, but this is disregarding the difference in security loss. As argued above, taking the security loss into account significantly blows up the costs of the scheme of [1]. For completeness, we should add that Fig. 1 hides the fact that our scheme relies on a hash function with a slightly stronger security assumption than the standard notion of second-preimage resistance.

2 Definitions

2.1 Notation

We first introduce some notation that we will use throughout the paper. We let $\{0,1\}^n$ denotes the set of bitstrings of length n, $\{0,1\}^{\leq n}$ denote the set of bitstrings of length at most n, and $\{0,1\}^*$ denote the set of bitstrings of arbitrary length. The notation $x \stackrel{\$}{\leftarrow} S$ denotes that x is assigned the value of an element selected uniformly at random from the set S. If A is an algorithm, then $x \leftarrow \mathsf{A}^{\mathcal{O}}(y, z)$ assigns to x the output of running A on inputs y and z, with access to oracle $\mathcal{O}$. A may be deterministic or probabilistic.

2.2 Syntax of WIBE Schemes, WIB-KEMs and DEMs

Syntax of WIBE schemes. 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 of depth L 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 $0 \le l \le L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^l$, all identities ID such that $ID \in_* P$, and all messages $m \in \{0,1\}^*$:

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

Syntax of 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 pairs (mpk, msk) output by Setup, all $0 \le l \le L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^l$, and all identities $ID \in_* P$,

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

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

Syntax of 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 a ciphertext C 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 $m \in \{0,1\}^*$, $\mathsf{Decrypt}(K, \mathsf{Encrypt}(K, m)) = m$.

2.3 Security Notions

Security games for WIBEs, WIB-KEMs and DEMs are presented in Figure 2. 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.

Security of WIBE schemes. We use the security definitions of indistinguishability under chosen-plaintext and chosen-ciphertext as per [1]. In both WIBE security games shown in Figure 2, $\mathcal{A}$ 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 (as shown in Figure 2) 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 (ID, C^*), 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 $\epsilon = \Pr[m' = m]$.

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. 2. Security games for WIBEs, WIB-KEMs and DEMs

Security of WIB-KEMs. In the IND-WID game for WIB-KEMs (also shown in Figure 2) $\mathcal{A}$ 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, $\mathcal{A}$ additionally has access to a decapsulation oracle, which on input (ID, C), 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 scheme (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 scheme (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 scheme and HIB-KEM are defined analogously.

Definition 3 *A WIBE scheme 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.*

When working in the random oracle model, we 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.

3 Security of the Hybrid Construction

Suppose we are given an IND-WID-CCA secure WIB-KEM scheme $\mathcal{WIB\text{-}KEM} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap})$ and an IND-CCA secure data encapsulation method $\mathcal{DEM} = (\mathsf{Encrypt}, \mathsf{Decrypt})$. 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 [11], we can combine them to form a WIBE scheme $\mathcal{WIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}', \mathsf{Decrypt}')$ 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, return $\perp$. Finally, compute $m \leftarrow \mathsf{Decrypt}(K, C_2)$, and return 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:*

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

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*.*

The theorem and proof are straightforward generalisations to the WIBE case of those in [11]. The proof is given in Appendix B Intuitively, the construction is secure as the KEM generates a one time symmetric key K, which "looks" random to the adversary, (i.e. is computationally indistinguishable from random) and this is enough for the DEM to be secure.

4 A Generic Construction 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 [12] 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 scheme $\mathcal{WIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ 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:

$$\Pr\left[\mathsf{Encrypt}(mpk, P, m; r) = C \ : \ r \xleftarrow{\$} \{0,1\}^\rho\right] \ \leq \ \gamma$$

where γ is a parameter of the scheme.

The only difficulty in applying the method of Dent [12] 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 originally encrypted. We assume that the set $W = \{i \in \mathbb{Z} : P_i = *\}$ is easily derived from the ciphertext. 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

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

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

- Setup and KeyDer are exactly as in $\mathcal{WIBE}$.
- $\mathsf{Encap}(mpk, P)$: Choose a random message $m \xleftarrow{\$} \mathcal{M}$. Compute $r \leftarrow H_1(m, P)$, $K \leftarrow H_2(m)$ 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 $r \leftarrow H_1(m, \mathsf{P}(ID, C))$ and check that $C = \mathsf{Encrypt}(mpk, \mathsf{P}(ID, C), m; r)$. If so, return $K \leftarrow H_2(m)$; 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, ϵ') adversary $\mathcal{B}$ against the OW-WID-CPA security of the WIBE, where:*

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

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

This proof of this theorem is a straightforward generalisation of the result of Dent [12]. The proof is given in Appendix C.

5 A Direct Construction 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 [15]. This construction is based on the Waters HIBE [20] and belongs to the Boneh-Boyen family of identity-based encryption schemes [6]. Before presenting our construction, we briefly recall the definitions for bilinear maps and second-preimage resistant hash functions:

Definition 7 (Bilinear map) *Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be multiplicative groups of prime order p. We say that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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 (BDDH problem) *We say that the BDDH problem in* $\mathbb{G}$ *is* (t, ϵ)*-hard if*

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

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

Definition 9 (Second-preimage resistant hash function) *A family* $F_{\{k \in \mathcal{K}\}} : \mathbb{G} \to \mathbb{Z}_p$ *of hash functions with key space* $\mathcal{K}$ *is called* (t, ϵ) *second-preimage resistant if all time* t *algorithms* $\mathcal{A}$ *have advantage at most* ϵ*, where the advantage of* $\mathcal{A}$ *is defined by:*

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

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 [15] in Figure 3. 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

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

We assume the function $h_1 : \mathbb{G} \to \mathbb{Z}_p^*$ is a second-preimage resistant hash function. The security of the Kiltz-Galindo scheme rests on the bilinear decisional Diffie-Hellman (BDDH) problem. Kiltz and Galindo proved the following security result of their scheme.

Theorem 10 *If there exists a* $(t, q_{\mathrm{K}}, q_{\mathrm{D}}, \epsilon)$ *attacker for the Kiltz-Galindo HIB-KEM in the IND-HID-CCA model, then there exists a* (t', ϵ') *algorithm which solves the BDDH problem in* $\mathbb{G}$ *and a* (t_h, ϵ_h) *attacker against the second pre-image resistance property of* h_1 *such that* $t' \leq t + O(\epsilon^{-2} \cdot \ln(\epsilon^{-1}))$*,* $t_h \leq O(t)$ *and*

$$\epsilon' \geq \frac{\epsilon - \epsilon_h}{(10(n+1)q)^L} - q/p\,,$$

where $q = q_{\mathrm{K}} + q_{\mathrm{D}}$ *and* p *is the order of* $\mathbb{G}$.

Note that the Kiltz-Galindo scheme generates keys which are elements of the group $\mathbb{G}_{\mathrm{T}}$, and we will follow this practise 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 [15] using the techniques of Abdalla *et al.* [1]. A naive implementation might try to construct an encapsulation algorithm as follows:

Algorithm Setup:
$v_1, v_2, v_3, \alpha \xleftarrow{\$} \mathbb{G}$; $z \leftarrow e(g, \alpha)$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1 \ldots L, j = 0 \ldots n$
$mpk \leftarrow (v_1, v_2, v_3, u_{1,0}, \ldots, u_{L,n}, z)$
$msk \leftarrow \alpha$
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
$C_i \leftarrow \left(u_{i,0} \prod_{j \in [ID_i]} u_{i,j}\right)^r$
$C_{l+1} \leftarrow (v_1^t v_2^l v_3)^r$
$K \leftarrow z^r$
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 $(g,\ C_0,\ v_1^t v_2^l v_3,\ C_{l+1})$
or $(g,\ C_0,\ u_{i,0} \prod_{j \in [ID_i]} u_{i,j},\ C_i)$,
for $i = 1 \ldots l$ 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. 3. The Kiltz-Galindo HIB-KEM scheme.

– $\mathsf{Encap}(mpk, P)$: 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 $1 \le i \le l$ compute C_i as

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

Finally, compute $t \leftarrow h_1(C_0)$, and $C_{l+1} \leftarrow (v_1^t v_2^l v_3)^r$.
The ciphertext $C = (C_0, \ldots, C_{l+1})$ is the encapsulation of key $K = z^r$.

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 increasing the scope of the hash function h_1. In the original proof of security, the hash function prevents an attacker from submitting a valid ciphertext C to the decapsulation oracle where C has the same decapsulation as C^* but $C_0 \neq C_0^*$. We extend this to prevent an attacker from submitting a valid ciphertext C to the decapsulation oracle where C has the same decapsulation but either $C_0 \neq C_0^*$ or C and C^* have wildcards in different positions. To do this we 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 $h_2(P_1) = h_2(P_2)$.

However, since an attacker can submit ciphertexts to the decapsulation oracle with patterns of his own choice, the increased scope of the hash function means that we have to rely on a slightly stronger assumption than standard second-preimage resistance. Informally, we will require the hash function to be second-preimage resistant, even when the attacker is allowed to choose the first L bits (corresponding to $h_2(P)$) of the challenge input for which he tries to find a collision. We formally define this property as follows:

Definition 11 (Extended second-preimage resistant hash function) *A family $F_{\{k \in \mathcal{K}\}} : \{0,1\}^{\leq L} \times \mathbb{G} \to \mathbb{Z}_p$ of hash functions with key space $\mathcal{K}$ is called (t, ϵ) extended second-preimage resistant if all time t algorithms $\mathcal{A}$ have advantage at most ϵ, where the advantage of $\mathcal{A}$ is defined by*

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

As in the description of the Kiltz-Galindo HIB-KEM, we will treat the family of hash functions as a fixed function to simplify the description of our scheme.

- Setup : Pick random elements $v_1, v_2, \alpha \xleftarrow{\$} \mathbb{G}$ 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 $0 \leq j \leq n$. The master public key is $mpk = (v_1, v_2, u_{1,0}, \ldots, u_{L,n}, z)$ 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 $d'_{l+1} \leftarrow g^{s_{l+1}}$. 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
$$C_i \leftarrow \begin{cases} \left(u_{i,0} \prod_{j \in [P_i]} u_{i,j}\right)^r & \text{if } P_i \neq * \\ \left(u_{i,0}^r, \ldots, u_{i,n}^r\right) & \text{if } P_i = * . \end{cases}$$
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 $t \leftarrow h_1(h_2(P), C_0)$, and $C_{l+1} \leftarrow (v_1^t v_2)^r$. The ciphertext $C = (C_0, \ldots, C_{l+1})$ 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 $(C_0, \ldots, C_{l+1})$. First compute $t \leftarrow h_1(h_2(P), C_0)$ 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
$$\begin{array}{l} (g \ , \ C_0 \ , \ v_1^t v_2 \ , \ C_{l+1}) \\ (g \ , \ C_0 \ , \ u_{i,0} \prod_{j \in [ID_i]} u_{i,j} \ , \ C_i) \quad \text{for } 1 \leq i \leq l, \ P_i \neq * \\ (g \ , \ C_0 \ , \ u_{i,j} \ , \ C_{i,j}) \quad \text{for } 1 \leq i \leq l, \ P_i = *, \ 0 \leq j \leq n \end{array}$$
are all Diffie-Hellman tuples. If not, return $\perp$. Rather than doing this test in the naive way by performing two pairing computations for each tuple, they can be aggregated in a single test as follows. Choose random exponents $r \xleftarrow{\$} \mathbb{Z}_p$, $r_i \xleftarrow{\$} \mathbb{Z}_p$ for $P_i \neq *$ and $r_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ for $P_i = *$, $0 \leq j \leq n$, compute
$$A \leftarrow (v_1^t v_2)^r \cdot \prod_{P_i \neq *} \left(u_{i,0} \prod_{j \in [ID_i]} u_{i,j}\right)^{r_i} \cdot \prod_{P_i = *} \prod_{j=0}^{n} u_{i,j}^{r_{i,j}}$$
$$B \leftarrow C_{l+1}^r \cdot \prod_{P_i \neq *} C_i^{r_i} \cdot \prod_{P_i = *} \prod_{j=0}^{n} C_{i,j}^{r_{i,j}}$$

and check whether $e(g, B) = e(C_0, A)$. If one or more of the tuples are not Diffie-Hellman tuples, this test fails with probability $1 - 1/p$. If it succeeds, decapsulate the key by first setting

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

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 $C = (C_0, \ldots, C_{l+1})$ 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 12 *If there exists a $(t, q_{\mathrm{K}}, q_{\mathrm{D}}, \epsilon)$ attacker for the Kiltz-Galindo WIB-KEM in the IND-WID-CCA model, then there exists a (t', ϵ') algorithm which solves the BDDH problem in $\mathbb{G}$ and a (t_h, ϵ_h) attacker against the extended second pre-image resistance property of h_1 such that $t' \leq t + O(\epsilon^{-2} \cdot \ln(\epsilon^{-1}))$, $t_h \leq O(t)$ and*

$$\epsilon' \geq \frac{\epsilon - \epsilon_h - q_D/p}{L(20(n+1)q_K)^L} \,,$$

where p is the order of $\mathbb{G}$.

The proof is given in Appendix A.

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 WIB-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.

We also note that the security proof for our construction can be completed, even if the used hash function is only assumed to be standard second-preimage resistant. However, this will add an additional security loss of $L2^L$ with respect to the hash function. Considering that security already degrades exponentially with L, this will not be a significant addition to the existing security loss and might be preferred instead of introducing a stronger assumption about the hash function.

6 Conclusion

We have proposed new chosen-ciphertext secure instantiations of WIBE schemes that improve on the existing schemes in both efficiency and security. To this end, we extended the KEM–DEM framework to the case of WIBE schemes. We proposed a generic construction in the random oracle model that transforms any one-way secure WIBE into a chosen-ciphertext secure WIB-KEM. We also proposed a direct construction of a WIB-KEM that is secure in the standard model. Our schemes overall gain at least a factor two in efficiency, especially when taking into account (as one should) the loose security bounds of all previously existing constructions.

Acknowledgements

The authors would like to thank Michel Abdalla, Dario Catalano, John Malone-Lee, Nigel Smart and Eike Kiltz for their comments on this paper. The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. The first author was also funded in part by the EPSRC. The third author is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO), and was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone and Ingo Wegener, editors, *ICALP 2006 (2)*, volume 4052 of *LNCS*, 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, 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, 1984.
5. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238, 2004.
6. 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, 2004.
7. 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, 2005.
8. 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, 2001.
9. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, 2004.
10. 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, 2001.
11. 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.

12. Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *Cryptography and Coding: 9th IMA International Conference*, volume 2898 of *LNCS*, pages 133–151, 2003.
13. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
14. Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP 2006*, volume 4058 of *LNCS*, pages 336–347, 2006.
15. Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. Unpublished manuscript, 2007.
16. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444, 1992.
17. Palash Sarkar and Sanjit Chatterjee. Transforming a CPA-secure HIBE protocol into a CCA-secure HIBE protocol without loss of security. Cryptology ePrint Archive, Report 2006/362, 2006. `http://eprint.iacr.org/`.
18. Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *Proc. of SCIS 2000*, Okinawa, Japan, January 2000.
19. 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, 1985.
20. Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, 2005.
21. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 1–12, Santa Barbara, CA, USA, August 23–27, 1998. Springer-Verlag, Berlin, Germany.
22. PKCS #1: RSA cryptography standard. RSA Data Security, Inc., June 1991.
23. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111, Perugia, Italy, May 9–12, 1994. Springer-Verlag, Berlin, Germany.
24. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004.

A Proof of security for Kiltz-Galindo WIB-KEM

Proof (Sketch). We combine the ideas of Abdalla *et al.* [1] and Kiltz-Galindo [15]. We will assume that starred variables correspond to the challenge ciphertext. For example, P^* is the challenge pattern. Consider a polynomial-time attacker $\mathcal{A}$. We begin the proof by changing the conditions in which $\mathcal{A}$ is said to win the game so that $\mathcal{A}$ wins the game if $b = b'$ and it never submitted a ciphertext C to the decapsulation oracle with $t = t^*$. Since $t = h_1(h_2(P), C_0)$ and the pair (C_0, P) uniquely defines the entire ciphertext, this collision can only occur if $\mathcal{A}$ submits the ciphertext C^* to the decapsulation oracle before the challenge phase (which can occur with probability at most q_D/p since r is chosen at random) or if there is an extended second pre-image collision in the hash function (which occurs with probability at most ϵ_h).

We now show that we can reduce the security of the scheme in this game to the DBDH problem. We begin by guessing the length of the challenge pattern and the position of the wildcards within the pattern. We guess this correctly with probability at least $1/(L2^L)$ and we abort if the attacker outputs a challenge pattern that differs from our guess or if the attacker makes an oracle query in the first stage that implies that our guess is incorrect. Let $W \subseteq \{1, 2, \ldots, L\}$ be the set of integers corresponding to the levels at which the wildcards appear in the challenge pattern.

The basic principle of the proof is to handle levels $i \notin W$ in exactly the same way as in the Kiltz-Galindo proof and to handle levels $i \in W$ in a naive way. We may extract private keys for identities in the same way as in the Waters HIBE. If we guess the position of the wildcards in the challenge pattern correctly, then this will mean we can extract private keys for all valid queries made by the attacker.

Note that since we have guessed the length and the location of the wildcards in the challenge pattern, we may immediately compute $h_2(P^*)$ even though we do not know the value of P^*.

Setup Our simulator takes as input a BDDH instance (g^a, g^b, g^c, Z). We will use g^c as C_0^* in the challenge ciphertext. Hence, we can immediately compute $t^* \leftarrow h_1(h_2(P^*), C_0^*)$. We use this to construct the public parameters for the encryption scheme as follows:

$$v_1 \leftarrow g^a \qquad d \xleftarrow{\$} \mathbb{Z}_p \qquad v_2 \leftarrow (g^a)^{-t^*} g^d \qquad z \leftarrow e(g^a, g^b) \qquad m \leftarrow 2q$$

Note that this implicitly defines $\alpha = g^{ab}$. For each level $i \notin W$ we compute

$$k_i \leftarrow \{1, \ldots, n\} \qquad x_{i,0}, x_{i,1}, \ldots, x_{i,n} \xleftarrow{\$} \mathbb{Z}_p \qquad y_{i,0}, y_{i,1}, \ldots, y_{i,n} \xleftarrow{\$} \{0, \ldots, m-1\}$$
$$u_{i,0} \leftarrow g^{x_{i,0}} v_1^{y_{i,0}-km} \qquad u_{i,j} \leftarrow g^{x_{i,j}} v_1^{y_{i,j}} \text{ for } 1 \le j \le n$$

For each level $i \in W$ we compute

$$x_{i,0}, x_{i,1}, \ldots, x_{i,n} \xleftarrow{\$} \mathbb{Z}_p \qquad u_{i,j} \leftarrow g^{x_{i,j}} \text{ for } 0 \le j \le n$$

We define the functions

$$F_i(ID_i) \leftarrow -mk_i + y_{i,0} + \sum_{j \in [ID_i]} y_{i,j}$$
$$J_i(ID_i) \leftarrow x_{i,0} + \sum_{j \in [ID_i]} x_{i,j}$$
$$K_i(ID_i) \leftarrow \begin{cases} 0 & \text{if } y_{i,0} + \sum_{j \in [ID_i]} y_{i,j} \equiv 0 \mod m \\ 1 & \text{otherwise} \end{cases}$$

Note that $F_i(ID_i) \equiv 0 \mod q$ if and only if $F_i(ID_i) = 0$, and so we have that $F_i(ID_i) \equiv 0 \mod q$ implies $K_i(ID_i) = 0$. Therefore, if $K_i(ID_i) = 1$ then $F_i(ID_i)$ can be inverted modulo q.

Key extraction oracle queries. Suppose an attacker makes a key extraction oracle query on the identity $ID = (ID_1, \ldots, ID_l)$. If this query is legal, then $ID \notin_* P^*$, which means that there must exists an integer i' such that $ID_{i'} \neq P_{i'} \neq *$. We demand that $K_{i'}(ID_{i'}) = 1$. This will occur with probability at least $1 - 1/m$. To extract the private key for ID we randomly choose $r_1, r_2, \ldots, r_l \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$d_0 \leftarrow v_1^{-\frac{J_{i'}(ID_{i'})}{F_{i'}(ID_{i'})}} \prod_{i=1}^{l} (u_{i,0} \prod_{j \in [ID_i]} u_{i,j})^{r_i}$$
$$d_{i'} \leftarrow v_1^{-\frac{1}{F_{i'}(ID_{i'})}} g^{r_{i'}} \qquad d_i \leftarrow g^{r_i} \text{ for all } i \neq i' .$$

A simple computation can verify that $(d_0, \ldots, d_l)$ is a valid private key for ID. The probability that such a private key can be computed for every key extraction oracle query is at least $(1 - 1/m)^{q_K} \ge 1 - q_K/m$. At this stage, the probability that the key extraction simulator fails may not be independent of the value of the message; hence, we use artificial aborts to ensure that we abort with the same probability regardless of the message value. By answering key extraction oracle queries in this way, we fail to accurately simulate the key extraction oracle with probability at most q_K/m.

Decryption oracle queries. Suppose an attacker makes a decryption oracle query for a ciphertext $C = (C_0, \ldots, C_{l+1})$ and an identity $ID = (ID_1, \ldots, ID_l)$. We first check that the ciphertext is consistent, i.e. that

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

are all Diffie-Hellman tuples, where $t = h_1(h_2(P), C_0)$ and P is the pattern under which the ciphertext was encrypted. If these tests fail, then the decryption oracle (correctly) outputs $\perp$. If the tests succeed and $t \neq t^*$ then we may decrypt the ciphertext by computing $K \leftarrow e\big(C_{l+1}/C_0^d\ ,\ g^b\big)^{1/(t-t^*)}$.

The challenge ciphertext. We assume that we correctly guessed the location of the wildcards in the challenge pattern $P^* = (P_1^*, \ldots, P_l^*)$. For every $i \notin W$ we require that $F_i(ID_i) = 0$. This will occur with probability at least $1/(nm)^L$ (as we require $K_i(ID_i) = 0$ and the correct value k_i to have been chosen). The challenge ciphertext is then built as follows. We set

$$K^* \leftarrow Z \qquad C_0^* \leftarrow g^c \qquad C_{l+1}^* \leftarrow (g^c)^d\,.$$

For each $i \notin W$, we set

$$C_i^* \leftarrow (g^c)^{J_i(ID_i)}\,.$$

For each $i \in W$, we set

$$C_{i,j}^* \leftarrow (g^c)^{x_{i,j}} \text{ for all } 1 \le j \le n\,.$$

It is clear to see that if the attacker can distinguish a valid key K from a randomly generated key K, then they will have distinguished a random value Z from the value $Z = e(g,g)^{abc}$. Hence, providing our simulation is correct, the simulator solves the BDDH problem whenever the attacker breaks the WIB-KEM. □

B Proof of security for the hybrid construction

We now restate the theorem of Section 3:

Theorem 13 *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:*

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

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

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

□

C Proof of security for the generic construction

We will prove this using a sequence of games in the manner of [24]. 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 [24].

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 of the game. Although the challenge pattern and challenge ciphertext are not yet defined in the first phase because the ciphertext is generated properly by the challenger at the start of the second phase, it is equivalent to consider the event that the challenger generates a ciphertext on which the adversary has already queried the decapsulation oracle.

Let E_1 be the event that the challenge ciphertext has already been queried to the decapsulation oracle by $\mathcal{A}$. 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 ϵ_2 of $\mathcal{A}$ in Game 2 satisfies

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

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 ϵ_3 of $\mathcal{A}$ in Game 3 satisfies

$$|\epsilon_3 - \epsilon_2| \leq \gamma q_D \, .$$

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. This is because in the random oracle model, $\mathcal{A}$ has no information about the encapsulated key K unless it queries the random oracle on m.

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 similarly to in Game 3 – passing key derivation queries to its own oracle and using H_1 and H_2-lists to answer decryption queries.

The slight difference is that it generates the hash values at random for itself, i.e it checks if there is a pair (x, h) in the H_i-list. If so, it returns h, otherwise it generates a random string h of the appropriate length (ρ for H_1 or λ for H_2), appends (x, h) into the H_i list and returns h. This accurately simulates the oracle.

Note that the simulation may add an entry to the H_2-list whenever $\mathcal{A}$ makes an H_2 oracle query, or when $\mathcal{A}$ makes a decapsulation query, since H_2 is used by the decapsulation oracle. 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) and checks if $\mathcal{A}_1$ queried the decapsulation oracle on (ID, C^*) for any identity $ID \in_* P^*$. If so, it outputs a random bit $b' \xleftarrow{\$} \{0,1\}$ and halts. If not, it then 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 such a query with probability at least ϵ_3 as before, and $\mathcal{B}$ wins if it then chooses the correct value of m from the list, which assuming the query was made occurs with probability $1/(q_{H_1} + q_{H_2} + q_D) = 1/(q_H + q_D)$, hence the advantage of $\mathcal{B}$ is

$$
\begin{aligned}
\epsilon' &= \epsilon_3/(q_H + q_D) \\
&\geq (\epsilon - \frac{q_D}{|\mathcal{M}|} - \gamma q_D)/(q_H + q_D)
\end{aligned}
$$

as required.

$\mathcal{B}$ 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. $\square$