J. Cryptol.
DOI: 10.1007/s00145-010-9060-3

Journal of
CRYPTOLOGY

Wildcarded Identity-Based Encryption*

Michel Abdalla
Ecole Normale Supérieure, LIENS-CNRS-INRIA, Paris, France
michel.abdalla@ens.fr
url: http://www.di.ens.fr/users/mabdalla

James Birkett
Information Security Institute, Queensland University of Technology, Brisbane, Australia
james.birkett@qut.edu.au

Dario Catalano
Dipartimento di Matematica e Informatica, Università di Catania, Catania, Italy
catalano@dmi.unict.it
url: http://www.ippari.unict.it/~catalano

Alexander W. Dent
Information Security Group, Royal Holloway, University of London, Egham, Surrey, UK
a.dent@rhul.ac.uk
url: http://www.isg.rhul.ac.uk/~alex

John Malone-Lee
EMB Consultancy, Epsom, Surrey, UK
jmalonelee@gmail.com

Gregory Neven
IBM Zurich Research Laboratory, Ruschlikon, Switzerland
gregory@neven.org
url: http://www.neven.org
and
Department of Electrical Engineering, Katholieke Universiteit Leuven, Heverlee, Belgium

Jacob C. N. Schuldt
Institute of Industrial Science, University of Tokyo, Tokyo, Japan
schuldt@iis.u-tokyo.ac.jp

Nigel P. Smart
Department of Computer Science, University of Bristol, Bristol, UK
nigel@cs.bris.ac.uk
url: http://www.cs.bris.ac.uk/~nigel

Communicated by Dan Boneh

* This paper is a combined and extended version of two conference papers [1,5].

Published online: 12 February 2010

Received 21 January 2009 and revised 14 January 2010

Abstract. In this paper, we introduce a new primitive called identity-based encryption with wildcards, or WIBE for short. It allows a sender to encrypt messages to a whole range of receivers whose identities match a certain pattern. This pattern is defined through a sequence of fixed strings and wildcards, where any string can take the place of a wildcard in a matching identity. Our primitive can be applied to provide an intuitive way to send encrypted email to groups of users in a corporate hierarchy. We propose a full security notion and give efficient implementations meeting this notion under different pairing-related assumptions, both in the random oracle model and in the standard model.

Key words. Identity-based encryption, Wildcard, Pairings.

1. Introduction

The concept of identity-based cryptography was introduced by Shamir as early as in 1984 [25], and the same paper proposed an identity-based signature scheme. However, it took nearly 20 years for an efficient identity-based encryption (IBE) scheme to be proposed. In 2000 and 2001, respectively, Sakai, Ohgishi and Kasahara [24] and Boneh and Franklin [8] proposed IBE schemes based on elliptic curve pairings. Also, in 2001 Cocks proposed a system based on the quadratic residuosity problem [13].

One of the main application areas proposed for IBE is that of email encryption. In this scenario, given an email address, one can encrypt a message to the owner of the email address without needing to obtain an authentic copy of the owner's public key first. In order to decrypt the email, the recipient must authenticate itself to a trusted authority who generates a private key corresponding to the email address used to encrypt the message.

1.1. *Identity-Based Encryption with Wildcards*

Our work is motivated by the fact that many email addresses correspond to groups of users rather than single individuals. Consider the scenario where there is some kind of organisational hierarchy. Take as an example an organisation called ECRYPT which is divided into virtual labs, say AZTEC and STVL. In addition, these virtual labs are further subdivided into working groups WG1, WG2 and WG3. Finally, each working group may consist of many individual members. There are several extensions of the IBE primitive to such a hierarchical setting (HIBE) [16,18]. The idea is that each level can issue keys to users on the level below. For example, the owner of the ECRYPT key can issue decryption keys for ECRYPT.AZTEC and ECRYPT.STVL.

Suppose that we wish to send an email to all the members of the AZTEC.WG1 working group, which includes the personal addresses

- ECRYPT.AZTEC.WG1.Nigel,
- ECRYPT.AZTEC.WG1.Dario,
- ECRYPT.AZTEC.WG1.John.

Given a standard HIBE, one would have to encrypt the message to each user individually. To address this limitation, we introduce the concept of *identity-based encryption*

with wildcards (WIBE). The way in which decryption keys are issued is exactly as in a standard HIBE scheme; what differs is encryption. Our primitive allows the encrypter to replace any component of the recipient identity with a *wildcard* so that any identity matching the *pattern* can decrypt. Denoting wildcards by `*`, in the example above the encrypter would use the identity

– ECRYPT.AZTEC.WG1.*

to encrypt to all members of the AZTEC.WG1 group.

It is often suggested that identity strings should be appended with the date so as to add timeliness to the message, and so try to mitigate the problems associated with key revocation. Using our technique we can now encrypt to a group of users, with a particular date, by encrypting to an identity of the form

– ECRYPT.AZTEC.WG1.*.22Oct2006

for example. Thus any individual in the group

– ECRYPT.AZTEC.WG1

with a decryption key for 22nd October 2006 will be able to decrypt.

As another example, take a hierarchy of email addresses at academic institutions of the form

– `name@department.university.edu`,

i.e. the email address of John Smith working at the computer science department of Some State University would be `johnsmith@cs.ssu.edu`. Using our primitive, one can send encrypted email to everyone in the computer science department at Some State University by encrypting to identity `*@cs.ssu.edu`, to everyone at SSU by encrypting to `*@*.ssu.edu`, to all computer scientists at any institution by encrypting to `*@cs.*.edu`, or to all system administrators in the university by encrypting to `sysadmin@*.ssu.edu`.

1.2. *Our Contributions*

In this paper, we introduce the primitive of identity-based encryption with wildcards, or a WIBE for short. We define appropriate security notions under chosen-plaintext and chosen-ciphertext attack, and present the first instantiations of this primitive. In more detail, we present the syntax and security notions in Sect. 3. To illustrate the relationship between WIBEs and other identity-based primitives, we show how WIBE schemes can be built from HIBE schemes and from fuzzy identity-based encryption schemes.

As is the case for most public-key and identity-based encryption schemes, the non-hybrid WIBE schemes 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 [6], but its security for the case of public-key encryption was not properly analysed until the work of Cramer and Shoup [14]. One would intuitively expect these results to extend to the case of WIBEs, which is indeed the case. We present the syntax for a WIB-KEM

in Sect. 4, along with the composition theorem which proves that the combination of a secure WIB-KEM and a secure DEM results in a secure WIBE scheme.

We also give several constructions for a WIBE scheme, classified according to their security guarantees. We first present the Boneh–Boyen WIBE (BB-WIBE—see Sect. 5.1) and the Boneh–Boyen–Goh WIBE (BBG-WIBE—see Sect. 5.2). These schemes are IND-CPA secure in the selective identity model and do not require random oracles to be proven secure, although we do require random oracles in order to prove their security in the full (non-selective-identity) model (see Sect. 5.4). We also present the Waters WIBE scheme (see Sect. 5.3) which is secure in the non-selective-identity IND-CPA setting without random oracles.

The range of IND-CPA WIBE schemes available makes selection difficult. The Waters WIBE scheme has the best security guarantees, but the worst performance. In particular, the number of elements in the master public key depends upon the maximum length of an identity, which is typically of the order of 160 bits. Hence, even with a small number of levels, the size of the master public key can be prohibitive. Both the BB-WIBE scheme and the BBG-WIBE scheme have better performance characteristics, but their security (in the non-selective-identity model) depends on random oracles. Furthermore, the BBG-WIBE scheme reduces to the less-studied L-BDHI assumption, but has the best performance characteristics.

The construction of IND-CCA secure WIBE schemes is more difficult. We present two generic transformations from an IND-CPA scheme into an IND-CCA scheme. The first transformation is based on the Canetti–Halevi–Katz transform (see Sect. 6.1) which builds an L-level IND-WID-CCA secure WIBE from an $(L + 1)$-level IND-WID-CPA WIBE. The disadvantage of our construction compared to the original CHK transform is that our construction always encrypts messages under patterns of length $L + 1$. This often increases the space and time complexity of the scheme in practical situations (as the worst performance characteristic are often obtained for "full-length" patterns). The approach we present in this paper is different from the approach given in the ePrint version of [1], which requires using $2L + 2$ levels as opposed to $L + 1$. We thank the anonymous referee for helping guide us to this improvement.

Our second transform is based on Dent's construction of a KEM (see Sect. 6.2). This converts a weakly secure (one-way) WIBE scheme into an IND-CCA secure WIB-KEM, but requires the random oracle model in order to prove its security. We note that one-way security is implied by IND-CPA security (for sufficiently large messages spaces). Consequently, we can use any of the IND-CPA constructions given in Sect. 5 to build an IND-CCA secure scheme.

In [5], we also presented a WIB-KEM in the standard model based on the Kiltz–Galindo HIB-KEM from [20]. Due to our improved CPA to CCA transform described above, this is no longer as efficient as the transformed Waters WIBE, hence we do not consider the Kiltz–Galindo WIB-KEM in this paper.

An overview of all the schemes we present is given in Table 1 and Table 2.

2. A Recap on Various Primitives

In this section, we recall basic notation and known results on different primitives that we will be using throughout this paper. In particular, we will recall several constructions of

Table 1. Efficiency comparison between our CPA-secure schemes. We compare the generic scheme of Sect. 3.3, the Waters-WIBE scheme of Sect. 5.3, the BB-WIBE scheme of Sect. 5.1, the BBG-WIBE scheme of Sect. 5.2, and the Waters-WIBE scheme of Sect. 5.3. The schemes are compared in terms of master number of elements in the public key ($|mpk|$), number of elements in the user secret key ($|d|$), number of element in the ciphertext ($|C|$), number of pairing operations required for decryption (Decrypt), the security assumption under which the scheme is proved secure, and whether this proof is in the random oracle model or not. The generic construction does not introduce any random oracles, but if the security proof of the HIBE scheme is in the random oracle model, then the WIBE obviously inherits this property. L is the maximal hierarchy depth and n is the bit length of an identity string. Figures are worst-case values, usually occurring for identities at level L with all-wildcard ciphertexts.

Scheme	$\|mpk\|$	$\|d\|$	$\|C\|$	Decrypt	Assumption	RO
Generic	$\|mpk_{HIBE}\|$	$2^L \cdot \|d_{HIBE}\|$	$\|C_{HIBE}\|$	$\mathsf{Decrypt}_{HIBE}$	IND-HID-CPA HIBE	No
BB-WIBE	$2L+3$	$L+1$	$2L+2$	$L+1$	BDDH	Yes
BBG-WIBE	$L+4$	$L+2$	$L+3$	2	L-BDHI	Yes
Waters-WIBE	$(n+1)L+3$	$L+1$	$(n+1)L+2$	$L+1$	BDDH	No

Table 2. Efficiency comparison between our CCA-secure schemes. The BB-WIBE scheme is the IND-WID-CPA scheme given in 5.1; the BBG-WIBE scheme is the IND-WID-CPA scheme given in 5.2; the Waters scheme is the IND-WID-CPA scheme given in 5.3. The $OML(\cdot)$ transformation refers to the (one more level) generic CCA-secure construction of a CCA-secure WIBE from a CPA-secure WIBE presented in Sect. 6.1. The $OW(\cdot)$ transformation is our random-oracle based construction of a WIB-KEM scheme from a CPA-secure WIBE presented in Sect. 6.2. We compare the schemes in terms of number of elements in the master public key ($|mpk|$), number of elements in the user secret key ($|d|$), number of elements in the ciphertext ($|C|$), number of exponentiations required for key encapsulation (Encap), number of pairings required for key decapsulation (Decap), and the dominant factor lost in the security reduction to the underlying assumption. L is the maximal hierarchy depth and n is the bit length of an identity string. The values q_H and q_K refer to the number of queries made by an adversary to the random oracle and key derivation oracle, respectively.

Scheme	$\|mpk\|$	$\|d\|$	$\|C\|$	Encap	Decap	Security loss
OML(BB-WIBE)	$2L+5$	$L+1$	$2L+3$	$2L+4$	$L+2$	q_H^{L+1}
OW(BB-WIBE)	$2L+3$	$L+1$	$2L+2$	$2L+2$	$L+1$	q_H^{L}
OML(BBG-WIBE)	$L+4$	$L+1$	$L+3$	$L+3$	2	q_H^{L+1}
OW(BBG-WIBE)	$L+4$	$L+1$	$L+3$	$L+3$	2	q_H^{L}
OML(Waters)	$(n+1)(L+1)+3$	$L+1$	$(n+1)L+3$	$(n+1)L+3$	$L+2$	$(2nq_K)^{L+1}$

Hierarchical Identity-Based Encryption schemes (HIBEs) upon which out Wildcarded Identity-Based Encryption schemes (WIBEs) are based.

2.1. *Basic Notation*

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers. Let ε be the empty string. If $n \in \mathbb{N}$, then $\{0,1\}^n$ denotes the set of n-bit strings and $\{0,1\}^*$ is the set of all finite bit strings. If $s = (s_1, \ldots, s_n)$ is an ordered sequence of n elements of some set and $0 \leq \ell \leq n$, then $s_{\leq \ell}$ is the ordered sequence consisting of the first ℓ elements of s, i.e. $s_{\leq \ell} = (s_1, \ldots, s_\ell)$. Furthermore, if ID is an n-bit string, then we set

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

If S is a finite set, then $y \xleftarrow{\$} S$ denotes the assignment to y of a randomly chosen element of the set S. If $\mathcal{A}$ is a deterministic algorithm, then $y \leftarrow \mathcal{A}(x)$ denotes the assignment to y of the output $\mathcal{A}$ when run on the input x. If $\mathcal{A}$ is a randomised algorithm, then $y \xleftarrow{\$} \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$ on the input x when the algorithm is run with fresh random coins.

2.2. *Hash Functions*

A hash function is a family of maps $F_k : \mathcal{IP} \to \mathcal{OP}$ index by a keyspace $\mathcal{K}$ in which the output space $\mathcal{OP}$ is finite. The input space $\mathcal{IP}$ may be finite or infinite. Additionally, the key space may be empty or non-empty. There are many security properties that can be ascribed to a hash function. We will only need to consider one security property at this time (although we will introduce further security notions in later sections and may model these hash functions as random oracles).

Definition 1. A (t, ϵ)-adversary $\mathcal{A}$ against the second pre-image resistance property of a family of hash functions $F_k : \mathcal{IP} \to \mathcal{OP}$ with a finite input space $\mathcal{IP}$ is an algorithm that runs in time at most t and has advantage at least ϵ, where the adversary's advantage is defined to be:

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

2.3. *One-Time Signature Schemes*

In order to amplify the security of a HIBE/WIBE (from IND-CPA security to IND-CCA security), we will make use of a one-time signature scheme. A one-time signature scheme is a triple of algorithms $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$. The key generation algorithm SigGen outputs signing and verification keys (sk, vk) for the signature scheme. The signing algorithm takes as input a signing key sk and a message $m \in \{0,1\}^*$, and outputs a signature $\sigma \in \{0,1\}^*$. The verification algorithm takes as input a verification key vk, a message $m \in \{0,1\}^*$ and a signature $\sigma \in \{0,1\}^*$, and outputs either $\top$ (indicating a valid signature) or $\bot$ (indicating an invalid signature). For correctness, we require that for all key pairs (sk, vk), messages $m \in \{0,1\}^*$, and signatures $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$, we have that $\mathsf{Verify}(vk, m, \sigma) = \top$ with probability one.

The security notion for a one-time unforgeable signature scheme is captured by the following game played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger:

1. The challenger generates a key pair $(sk^*, vk^*) \xleftarrow{\$} \mathsf{SigGen}$.
2. The adversary runs $\mathcal{A}_1$ on input vk^*. The adversary outputs a message m^* and some state information *state*.
3. The challenger computes $\sigma^* \xleftarrow{\$} \mathsf{Sign}(sk^*, m^*)$.
4. The adversary runs $\mathcal{A}_2$ on σ^* and *state*. The adversary outputs a message signature pair (m, σ).

The adversary wins the game if $\mathsf{Verify}(vk^*, m, \sigma) = \top$ and $(m, \sigma) \neq (m^*, \sigma^*)$. The adversary's advantage is defined to be $\Pr[\,\mathcal{A}\text{ wins}\,]$.

Definition 2. A (t, ϵ)-adversary against the one-time unforgeability of the signature scheme is an algorithm that runs in time t and has advantage at least ϵ in winning the above game.

2.4. *Bilinear Maps and Related Assumptions*

Let $\mathbb{G}, \mathbb{G}_T$ be multiplicative groups of prime order p with an admissible map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. By admissible we mean that the map is bilinear, non-degenerate and efficiently computable. Bilinearity means that for all $a, b \in \mathbb{Z}_p$ and all $g \in \mathbb{G}$ we have $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$. By non-degenerate we mean that $\hat{e}(g, g) = 1$ if and only if $g = 1$.

In such a setting, we can define a number of computational problems. The first we shall be interested in is called the bilinear decisional Diffie–Hellman (BDDH) problem [19]: given a tuple (g, g^a, g^b, g^c, T), the problem is to decide whether $T = \hat{e}(g, g)^{abc}$ or whether it is a random element of $\mathbb{G}_T$. More formally, we define the following game between an adversary $\mathcal{A}$ and a challenger. The challenger first chooses a random generator $g \xleftarrow{\$} \mathbb{G}^*$, random integers $a, b, c \xleftarrow{\$} \mathbb{Z}_p$, a random element $T \xleftarrow{\$} \mathbb{G}_T$, and a random bit $\beta \xleftarrow{\$} \{0, 1\}$. If $\beta = 1$ it feeds $\mathcal{A}$ the tuple $(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc})$ as input; if $\beta = 0$ it feeds $\mathcal{A}$ the tuple (g, g^a, g^b, g^c, T) as input. The adversary $\mathcal{A}$ must then output its guess β' for β. The adversary has advantage ϵ in solving the BDDH problem if

$$\left| \Pr\left[\mathcal{A}\left(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}\right) = 1\right] - \Pr\left[\mathcal{A}\left(g, g^a, g^b, g^c, T\right) = 1\right]\right| \geq \epsilon,$$

where the probabilities are over the random choice of g, a, b, c, T and over the random coins of $\mathcal{A}$.

Definition 3. A (t, ϵ)-adversary $\mathcal{A}$ against the BDDH problem is an algorithm that runs in time at most t and has advantage at least ϵ.

We note that throughout this paper we will assume that the time t of an adversary includes its code size, in order to exclude trivial "lookup" adversaries.

A second problem we will use in our constructions is the ℓ-bilinear Diffie–Hellman Inversion (ℓ-BDHI) problem [7,21]. The problem is to compute $\hat{e}(g, g)^{1/\alpha}$ for random $g \xleftarrow{\$} \mathbb{G}^*$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$ given $g, g^\alpha, \ldots, g^{(\alpha^\ell)}$. The decisional variant of this problem is to distinguish $\hat{e}(g, g)^{1/\alpha}$ from a random element of $\mathbb{G}_T$. We say that adversary $\mathcal{A}$ has advantage ϵ in solving the decisional ℓ-BDHI problem if

$$\left| \Pr\left[\mathcal{A}\left(g, g^\alpha, \ldots, g^{(\alpha^\ell)}, \hat{e}(g, g)^{1/\alpha}\right) = 1\right] - \Pr\left[\mathcal{A}\left(g, g^\alpha, \ldots, g^{(\alpha^\ell)}, T\right) = 1\right]\right| \geq \epsilon,$$

where the probability is over the random choice of $g \xleftarrow{\$} \mathbb{G}^*$, $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $T \xleftarrow{\$} \mathbb{G}_T$, and the coins of $\mathcal{A}$.

Definition 4. A (t, ϵ)-adversary against the decisional ℓ-BDHI problem is an algorithm that runs in time at most t and has advantage at least ϵ in the above game.

We note that the BDDH problem is a weaker assumption than the ℓ-BDHI assumption. Hence, all other things being equal schemes which are based on the BDDH assumption are to be preferred to ones based on the ℓ-BDHI assumption. However, our

most efficient constructions are based on the ℓ-BDHI assumption as opposed to the BDDH assumption. As the two assumptions are very different in nature, it is hard to compare precisely various schemes; indeed, the comparison would depend on the readers view with respect to the interpretation of exact security results and the view of the relative hardness of the two underlying problems.

2.5. *Hierarchical Identity-Based Encryption*

An identity-based encryption (IBE) scheme is a tuple of algorithms (Setup, KeyDer, Encrypt, Decrypt) providing the following functionality. The trusted authority runs Setup to generate a master key pair (mpk, msk). It publishes the master public key mpk and keeps the master secret key msk private. When a user with identity ID wishes to become part of the system, the trusted authority generates a decryption key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$, and sends this key over a secure and authenticated channel to the user. To send an encrypted message m to the user with identity ID, the sender computes the ciphertext $C \xleftarrow{\$} \mathsf{Encrypt}(mpk, ID, m)$, which can be decrypted by the user as $m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$. We refer to [8] for details on the security definitions for IBE schemes.

In this paper, we are more interested in the concept of Hierarchical Identity-Based Encryption (HIBE) [16,18]. In a HIBE scheme, users are organised in a tree of depth L, with the root being the master trusted authority. The identity of a user at level $0 \leq \ell \leq L$ in the tree is given by a vector $ID = (ID_1, \ldots, ID_\ell) \in (\{0,1\}^*)^\ell$. A HIBE scheme is a tuple of algorithms (Setup, KeyDer, Encrypt, Decrypt) providing the same functionality as in an IBE scheme, except that a user $ID = (ID_1, \ldots, ID_\ell)$ at level ℓ can use its own secret key d_{ID} to generate a secret key for any of its children $ID' = (ID_1, \ldots, ID_\ell, ID_{\ell+1})$ via $d_{ID'} \xleftarrow{\$} \mathsf{KeyDer}(d_{ID}, ID_{\ell+1})$. Note that by iteratively applying the KeyDer algorithm, user ID can derive secret keys for any of its descendants $ID' = (ID_1, \ldots, ID_{\ell+\delta})$, $\delta \geq 0$. We will occasionally use the overloaded notation

$$d_{ID'} \xleftarrow{\$} \mathsf{KeyDer}\big(d_{ID}, (ID_{\ell+1}, \ldots, ID_{\ell+\delta})\big)$$

to denote this process. The secret key of the root identity at level 0 is $d_\varepsilon \leftarrow msk$. Encryption and decryption are the same as for IBE, but with vectors of bit strings as identities instead of ordinary bit strings.

The security of a HIBE scheme is defined through the following IND-HID-CPA game, played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary outputs two equal-length messages (m_0, m_1) and a challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell*}^*)$, along with some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the ciphertext $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, ID^*, m_\beta)$.

4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information *state*. The adversary is given access to a key derivation oracle as before. The adversary outputs a bit β'.

In most cases, we will suppress the *state* information passed between adversary algorithms and simply assume that all necessary details are passed from one algorithm to the next. The adversary wins the game if $\beta = \beta'$ and it never queries the key derivation oracle with any ancestor identity of ID^*, i.e. any identity $ID = (ID_1^*, \ldots, ID_\ell^*)$ where $\ell \leq \ell^*$. The adversary's advantage is defined to be equal to $|2 \cdot \Pr[\,\mathcal{A} \text{ wins}\,] - 1|$.

Definition 5. A (t, q_K, ϵ)-adversary against the IND-HID-CPA security of a HIBE scheme is an algorithm that runs in time at most t, makes at most q_K queries to the key derivation oracle, and has advantage at least ϵ in winning the IND-HID-CPA game described above.

The IND-HID-CCA security game is identical to the IND-HID-CPA security game with the exception that in the IND-HID-CCA security game the adversary additionally has access to a decryption oracle that, on input of a ciphertext C and an identity ID, returns the decryption $m \leftarrow \mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C)$. The adversary wins the game if $\beta = \beta'$, it never queries the key derivation oracle with any ancestor identity of ID^*, and it never queries the decryption oracle with the pair (C^*, ID^*) after the challenge ciphertext is computed.

Definition 6. A (t, q_K, q_D, ϵ)-adversary against the IND-HID-CCA security of the HIBE scheme is an algorithm that runs in time at most t, makes at most q_K queries to the key derivation oracle, makes at most q_D queries to the decryption oracle, and has advantage at least ϵ in winning the IND-HID-CCA game described above.

In a *selective-identity* (sID) attack [7], the adversary has to output the challenge identity ID^* at the very beginning of the game, before even seeing the master public key. In other words, the adversary is considered to be a triple $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_0$ simply outputs the challenge identity (and some state information to be passed to $\mathcal{A}_1$). The definitions for IND-HID-CPA and IND-HID-CCA security are otherwise identical to those above. In the random oracle model [2], all algorithms, as well as the adversary, have access to a random oracle mapping arbitrary bit strings onto a range that possibly depends on the master public key. All above security definitions then take an extra parameter q_H denoting the adversary's maximum number of queries to the random oracle.

We now recap on the main efficient HIBE constructions in the literature, namely the HIBE schemes of Waters (W-HIBE), Boneh–Boyen (BB-HIBE), and Boneh–Boyen–Goh (BBG-HIBE).

2.6. *The Boneh–Boyen HIBE*

In this section, we present a variant of the HIBE scheme by Boneh and Boyen [7]. In this scheme, we assume that identities are vectors of elements of $\mathbb{Z}_p$—if necessary this can be achieved by applying a collision-resistant hash function $h : \{0,1\}^* \to \mathbb{Z}_p$ to binary identities before applying the scheme. The scheme is described in Fig. 1. The

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$, $j = 0, 1$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,1})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot \left(u_{\ell+1,0} \cdot u_{\ell+1,1}^{ID_{\ell+1}}\right)^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, ID, m):
Parse ID as $(ID_1, \ldots, ID_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1, \ldots, \ell$ do
$C_{2,i} \leftarrow \left(u_{i,0} \cdot u_{i,1}^{ID_i}\right)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_l)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Fig. 1. The Boneh–Boyen HIBE scheme.

main difference between the original HIBE scheme of [7] and our variant above is that our scheme uses a different value $u_{i,1}$ for each level, while the original scheme uses the same value u_1 for all levels. Adding wildcard functionality to the original scheme would require us to include u_1^r in the ciphertext, but this ruins security as it can be used to change the identity for which a ciphertext is encrypted.

For completeness, we prove the security of this new HIBE scheme, despite its similarities to scheme of Boneh and Boyen [7].

Theorem 1. *If there exists a (t, q_K, ϵ)-adversary against the IND-sHID-CPA security of the BB-HIBE (with hierarchy depth L) then there exists a (t', ϵ')-adversary against the BDDH problem in $\mathbb{G}$, where $\epsilon' \geq \epsilon - q_K/p$ and $t' \leq t + O(L \cdot q_K \cdot t_{\text{exp}})$ and t_{exp} is the maximum time for an exponentiation in $\mathbb{G}$ and p is the order of $\mathbb{G}$.*

Proof. The present proof follows very closely the proof of security for the original scheme in [7]. As before, we assume that there exist an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ that breaks the IND-sID-CPA-security of the BB-HIBE scheme and then we show how to efficiently build another adversary $\mathcal{B}$ that, using $\mathcal{A}$ as a subroutine, manages to solve the BDDH problem in $\mathbb{G}$.

Algorithm $\mathcal{B}$ first receives as input a random tuple $(g, A = g^a, B = g^b, C = g^c, Z)$ and its goal is to determine whether $Z = \hat{e}(g, g)^{abc}$ or $\hat{e}(g, g)^z$ for a random element z in $\mathbb{Z}_p$. Algorithm $\mathcal{B}$ should output 1 if $Z = \hat{e}(g, g)^{abc}$ and 0 otherwise. Algorithm $\mathcal{B}$ works as follows.

Initialisation. Algorithm $\mathcal{B}$ starts by running algorithm $\mathcal{A}_0$, which responds with the challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$ where $0 \leq \ell^* \leq L$. If $\ell^* = L$ then $\mathcal{B}$ sets $\tilde{ID^*} \leftarrow ID^*$. Otherwise, $\mathcal{B}$ randomly generates $ID_{\ell^*+1}^*, \ldots, ID_L^* \xleftarrow{\$} \mathbb{Z}_p$ and sets $\tilde{ID^*} \leftarrow (ID_1^*, \ldots, ID_L^*)$.

Setup. To generate the systems parameters, $\mathcal{B}$ first sets $g_1 \leftarrow g$, $h_1 \leftarrow A$, and $g_2 \leftarrow B$. Algorithm $\mathcal{B}$ then chooses $\alpha_{1,0}, \ldots, \alpha_{L,0}, \alpha_{1,1}, \ldots, \alpha_{L,1} \xleftarrow{\$} \mathbb{Z}_p^*$ at random and sets $u_{i,0} \leftarrow g_1^{\alpha_{i,0}} \cdot h_1^{-ID_i^* \alpha_{i,1}}$ and $u_{i,1} \leftarrow h_1^{\alpha_{i,1}}$ for $i = 1, \ldots, L$. $\mathcal{B}$ defines the master public key to be $mpk \leftarrow (g_1, h_1, g_2, u_{1,0}, \ldots, u_{L,0}, u_{1,1}, \ldots, u_{L,1})$. Note that the corresponding master secret key $msk = g_2^a$ is unknown to $\mathcal{B}$.

Phase 1. $\mathcal{B}$ runs $\mathcal{A}_1$ on input mpk. If $\mathcal{A}_1$ makes a key derivation oracle query on $ID = (ID_1, \ldots, ID_\ell)$, where $ID_i \in \mathbb{Z}_p$ and $\ell \leq L$ then ID cannot be a prefix of ID^*. Hence, if ID is a prefix of $\tilde{ID}^*$ then $\mathcal{A}$ aborts; we let E be the event that this occurs. Otherwise, let j be the smallest index such that $ID_j \neq \tilde{ID}_j^*$. To reply to this query, $\mathcal{B}$ first computes the key for identity $ID' = (ID_1, \ldots, ID_j)$ and then derive the key for ID using the key derivation algorithm. To derive the key for identity ID', $\mathcal{B}$ chooses the values $r_1, \ldots, r_j \xleftarrow{\$} \mathbb{Z}_p$ at random and sets $d_{ID'} = (a_0, a_1, \ldots, a_j)$ where

$$a_0 \leftarrow g_2^{\frac{-\alpha_{j,0}}{\alpha_{j,1}(ID_j - ID_j^*)}} \cdot \prod_{i=1}^{j} \left(u_{i,0} \cdot u_{i,1}^{ID_i}\right)^{r_i},$$

$$a_i \leftarrow g_1^{r_i} \quad \text{for } i = 1, \ldots, j-1,$$

$$a_j \leftarrow g_2^{\frac{-1}{\alpha_{j,1}(ID_j - ID_j^*)}} \cdot g_1^{r_j}.$$

Algorithm $\mathcal{A}_1$ terminates and outputs two equal-length messages (m_0, m_1).

Challenge. Algorithm $\mathcal{B}$ then chooses a random bit $\beta \xleftarrow{\$} \{0, 1\}$ and sends $C^* = (C, C^{\alpha_{1,0}}, \ldots, C^{\alpha_{\ell^*,0}}, m_\beta \cdot Z)$ to $\mathcal{A}$ as the challenge ciphertext. Since $u_{i,0} \cdot u_{i,1}^{ID_i^*} = g_1^{\alpha_{i,0}}$ for all i, we have that

$$C^* = \left(g_1^c, \left(u_{1,0} \cdot u_{1,1}^{ID_i^*}\right)^c, \ldots, \left(u_{\ell^*,0} \cdot u_{\ell^*,1}^{ID_{\ell^*}^*}\right)^c, m_\beta \cdot Z\right).$$

As a result, when $Z = \hat{e}(g, g)^{abc} = \hat{e}(h_1, g_2)^c$, C^* is a valid encryption of message m_β for the challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$. On the other hand, when $Z = \hat{e}(g, g)^z$ for a random value $z \xleftarrow{\$} \mathbb{Z}_p$, then the challenge ciphertext is independent of β from the view point of the adversary.

Phase 2. $\mathcal{B}$ runs $\mathcal{A}_2$ on the challenge ciphertext C^*. If $\mathcal{A}_2$ makes any key derivation oracle queries, then they are answered as in Phase 1. $\mathcal{A}_2$ terminates and outputs a bit β'.

Output. If $\beta = \beta'$ then $\mathcal{B}$ outputs 1, guessing that $Z = \hat{e}(g, g)^{abc}$, otherwise $\mathcal{B}$ outputs 1.

Suppose E does not occur. Clearly, when $Z = \hat{e}(g, g)^{abc}$, the view of $\mathcal{A}$ is identical to its view in a real attack and, thus, the probability that $b = b'$ is exactly the probability that $\mathcal{A}$ wins the IND-sHID-CPA game. On the other hand, when Z is a random group element in $\mathbb{G}_T$, then the probability that $b = b'$ is exactly $1/2$. Hence, if E does not occur then $\mathcal{A}$ wins with probability ϵ. If E does occur, then the simulator fails; however, for E to occur then $\mathcal{A}$ must submit a key extraction query for an identity ID where ID^*

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_i \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$
$mpk \leftarrow (g_1, g_2, h_1, u_0, \ldots, u_L)$
$d_0 \leftarrow h_2$
For $i = 1, \ldots, L+1$ do
  $d_i \leftarrow 1$
$msk \leftarrow (d_0, d_1, \ldots, d_L, d_{L+1})$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_L, d_{L+1})$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot d_{\ell+1}^{ID_{\ell+1}} \cdot \left(u_0 \prod_{i=1}^{\ell} u_i^{ID_i}\right)^{r_{\ell+1}}$
For $i = \ell+2, \ldots, L$ do
  $d'_i \leftarrow d_i \cdot u_i^{r_{\ell+1}}$
$d'_{L+1} \leftarrow d_{L+1} \cdot g_1^{r_{\ell+1}}$
Return $(d'_0, d'_{\ell+2}, \ldots, d'_L, d'_{L+1})$

Algorithm Encrypt(mpk, ID, m):
Parse ID as $(ID_1, \ldots, ID_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
$C_2 \leftarrow \left(u_0 \prod_{i=1}^{\ell} u_i^{ID_i}\right)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return (C_1, C_2, C_3)

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_{L+1})$
Parse C as (C_1, C_2, C_3)
$m' \leftarrow C_3 \cdot \frac{\hat{e}(C_2, d_{L+1})}{\hat{e}(C_1, d_0)}$
Return m'

Fig. 2. The Boneh–Boyen–Goh HIBE scheme.

is a prefix of ID and ID is a prefix of $\tilde{ID}^*$. This implies that $ID_{\ell^*+1} = ID^*_{\ell^*+1}$ but, since $ID^*_{\ell^*+1}$ is chosen at random and hidden from the execution of the attacker $\mathcal{A}$, we have that $\Pr[E] \leq q_K/p$. From the above, the result announced in Theorem 1 follows immediately. □

2.7. *The Boneh–Boyen–Goh Scheme*

In this section, we present the HIBE scheme due to Boneh, Boyen and Goh [10], referred to as the BBG-HIBE scheme here. Again, we assume that identities are vectors of elements of $\mathbb{Z}_p$. The scheme is described in Fig. 2.

The following theorem about the security of the scheme was proved in (the full version of) [10].

Theorem 2. *If there exists a (t, q_K, ϵ)-adversary against the IND-sHID-CPA security of the BBG-HIBE (with hierarchy depth L) then there exists a (t', ϵ')-adversary against the L-BDHI problem in $\mathbb{G}$, where $\epsilon' \geq \epsilon$ and $t' \leq t + O(L \cdot q_K \cdot t_{\text{exp}})$ and t_{exp} is the time for an exponentiation in $\mathbb{G}$.*

2.8. *The Waters Scheme*

Waters [27] argued that his IBE scheme can easily be modified into an L-level HIBE scheme as per [7]. Here we explicitly present this construction, that we refer to as the Waters-HIBE scheme. The scheme makes use of n-bit identities and is described in Fig. 3. The scheme makes use of group elements $(u_{1,0}, \ldots, u_{L,n})$ which are available as part of the scheme's public parameters. These group elements define a series of hash

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^\alpha$; $h_2 \leftarrow g_2^\alpha$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$; $j = 0, \ldots, n$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,n})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot F_{\ell+1}(ID_{\ell+1})^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, ID, m):
Parse ID as $(ID_1, \ldots, ID_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1, \ldots, \ell$ do
$C_{2,i} \leftarrow F_i(ID_i)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Fig. 3. The Waters HIBE scheme.

functions $(F_1, \ldots, F_L)$ where

$$F_i(ID_i) = u_{i,0} \prod_{j \in [ID_i]} u_{i,j}.$$

Waters [27] informally states that the above HIBE scheme is IND-HID-CPA secure under the BDDH assumption, in the sense that if there exists a (t, q_K, ϵ)-adversary against the HIBE, then there exists an algorithm solving the BDDH problem with advantage $\epsilon' = O((n \cdot q_K)^L \epsilon)$. We shall assume in what follows that the Waters HIBE scheme is indeed IND-HID-CPA secure. However, the reader should be aware that any security results we state for schemes derived from the Water HIBE scheme are conjectural relative to the above assumption.

2.9. *Hierarchical Identity-Based Key Encapsulation*

One efficient paradigm for producing HIBE schemes is to the hybrid KEM-DEM construction. In the public key setting, this was first formally investigated by Cramer and Shoup [14] and extended to the identity-based setting by Bentahar et al. [3]. A hybrid construction consists of an asymmetric KEM and a symmetric DEM.

A hierarchical identity-based KEM (HIB-KEM) consists of four algorithms (Setup, KeyDer, Encap, Decap). The setup algorithm Setup and key derivation algorithm KeyDer have the same syntax as for a HIBE scheme. The encapsulation algorithm Encap takes as input a master public key mpk and an identity $ID = (ID_1, \ldots, ID_\ell)$ with $0 \leq \ell \leq L$; it outputs a symmetric key $K \in \{0,1\}^\lambda$ and an encapsulation C. The decapsulation algorithm Decap takes as input a private key d_{ID} and an encapsulation C, and outputs either a symmetric key $K \in \{0,1\}^\lambda$ or the error symbol $\perp$.

The security models for a HIB-KEM is similar to those of a HIBE scheme. The IND-HID-CCA game for a HIB-KEM, played between an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger, is defined as follows:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary is also given access to a decryption oracle that will, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$ and a ciphertext C, return $\mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C)$. The adversary outputs a challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$ and some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$, computes the encapsulation $(C^*, K_0) \xleftarrow{\$} \mathsf{Encap}(mpk, ID^*)$ and chooses a random key $K_1 \xleftarrow{\$} \{0,1\}^\lambda$.
4. The adversary runs $\mathcal{A}_2$ on the input (C^*, K_β) and the state information *state*. The adversary is given access to a key derivation oracle and decryption oracle as before. The adversary outputs a bit β'.

The adversary wins the game if $\beta = \beta'$, it never queries the key derivation oracle with any ancestor identity of ID^*, and if it does not query the decryption oracle on the pair (ID^*, C^*) after it receives the challenge ciphertext. As usual, the adversary's advantage is defined to be equal to $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition 7. A (t, q_K, q_D, ϵ)-adversary against the IND-HID-CCA security of the HIB-KEM is an algorithm that runs in time at most t, makes at most q_K queries to the key derivation oracle, makes at most q_D queries to the decryption oracle, and has advantage at least ϵ in winning the IND-HID-CCA game described above.

Again, if the random oracle model [2] is used in the analysis of a scheme, then the above security definitions take an extra parameter q_H as input. This parameter denotes the adversary's maximum number of queries to the random oracle.

A DEM is a pair of deterministic algorithms $(\mathsf{Enc}, \mathsf{Dec})$. The encryption algorithm Enc takes as input a symmetric key $K \in \{0,1\}^\lambda$ and a message m of arbitrary length, and outputs a ciphertext $C \leftarrow \mathsf{Dec}(K, C)$. The decryption algorithm Dec takes as input a symmetric key $K \in \{0,1\}^\lambda$ and a ciphertext C, and returns either a message m or the error symbol $\perp$. The DEM must satisfy the following soundness property: for all $K \in \{0,1\}^\lambda$ and for all $m \in \{0,1\}^*$, we have that $\mathsf{Dec}(K, \mathsf{Enc}(K, m)) = m$.

The only security model which will concern us for DEMs is the (one-time) IND-CCA security game, which is played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a key $K \xleftarrow{\$} \{0,1\}^\lambda$.
2. The adversary runs $\mathcal{A}_1$. The adversary outputs two equal-length messages (m_0, m_1) and some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the ciphertext $C^* \leftarrow \mathsf{Enc}(K, m_\beta)$.
4. The adversary runs $\mathcal{A}_2$ on input C^* and the state information *state*. The adversary may query a decryption oracle which will, on input of a ciphertext $C \neq C^*$, return $\mathsf{Dec}(K, C)$. The adversary outputs a bit β'.

The adversary wins if $\beta = \beta'$ and its advantage is defined to be $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition 8. A (t, q_D, ϵ)-adversary against the (one-time) IND-CCA security of the DEM is an algorithm that runs in time at most t, makes at most q_D decryption oracle queries, and has advantage at least ϵ in winning the IND-CCA game described above.

A HIB-KEM and a DEM can be "glued" together to form a complete HIBE scheme. Further details can be found in [3].

2.10. *The Canetti–Halevi–Katz Transform*

We shall, in one of our constructions of a CCA WIBE scheme, make use of the techniques behind the Canetti–Halevi–Katz transform [11]. To aid the reader we recap on this here. This is a transform to turn a weakly secure (IND-sID-CPA) IBE scheme into a fully secure (IND-CCA) public key encryption scheme. We let $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ denote the key-generation, extraction, encryption, and decryption algorithms of the IBE scheme, and $(\mathsf{Setup}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ denote the key-generation, encryption, and decryption algorithms of the derived public key scheme. The transform also makes use of a one-time signature scheme, defined by a tuple of algorithms $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$.

The algorithm Setup' is defined to be equal to Setup, i.e. public/private key of the PKE scheme is the master public/private keys, (mpk, msk), of the IBE scheme. Algorithm $\mathsf{Encrypt}'$ is defined as follows: First, a key-pair (sk, vk) for the one-time signature scheme is created by calling SigGen; then the message is encrypted via $\mathsf{Encrypt}(mpk, vk, m)$ with respect to the "identity" vk to produce c. The resulting ciphertext c is then signed with sk to produce $\sigma = \mathsf{Sign}(sk, c)$. The tuple (vk, c, σ) is the ciphertext for our PKE.

To decrypt the recipient first verifies σ is a valid signature on c with respect to the verification key vk, by calling $\mathsf{Verify}(vk, c, \sigma)$. If it is then the function KeyDer is called with respect to the "identity" vk, using private key of the PKE (i.e. msk). Then the ciphertext can be decrypted using the algorithm $\mathsf{Decrypt}$.

3. Wildcard Identity-Based Encryption

3.1. *Syntax*

Identity-based encryption with wildcards (WIBE) schemes are essentially a generalisation of HIBE schemes where at the time of encryption, the sender can decide to make the ciphertext decryptable by a whole range of users whose identities match a certain pattern. Such a pattern is described by a vector $P = (P_1, \ldots, P_\ell) \in (\{0,1\}^* \cup \{*\})^\ell$, where $*$ is a special wildcard symbol. We say that identity $ID = (ID_1, \ldots, ID_{\ell'})$ *matches* P, denoted $ID \in_* P$, if and only if $\ell' \leq \ell$ and for all $i = 1, \ldots, \ell'$ we have that $ID_i = P_i$ or $P_i = *$. Note that under this definition, any ancestor of a matching identity is also a matching identity. This is reasonable for our purposes because any ancestor can derive the secret key of a matching descendant identity anyway.

If $P = (P_1, \ldots, P_\ell)$ is a pattern, then we define $W(P)$ to be the set of wildcard positions in P, i.e.

$$W(P) = \{1 \leq i \leq \ell \,:\, P_i = *\}.$$

Formally, a WIBE scheme is a tuple of algorithms (Setup, KeyDer, Encrypt, Decrypt) providing the following functionality. The Setup and KeyDer algorithms behave exactly as those of a HIBE scheme. To create a ciphertext of a message $m \in \{0,1\}^*$ intended for all identities matching pattern P, the sender computes $C \xleftarrow{\$} \mathsf{Encrypt}(mpk, P, m)$. Any of the intended recipients $ID \in_* P$ can decrypt the ciphertext using its own decryption key as $m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$.

Note that we implicitly assume that the pattern P used to encrypt the message is included within the ciphertext. This is because any parent of the pattern should be able to decrypt the message, and hence the parent will need to be able to fill in the non-wildcarded entries in the pattern for decryption. For example, suppose the pattern is $P = (ID_1, *, ID_3)$ and that the decryptor has identity $ID = (ID_1, ID_2)$. Then by our definition of a matching pattern we have $ID \in_* P$, and so the decryptor will need to be informed of ID_3 so as to be able to decrypt the ciphertext. Note that an anonymous version of the definitions can be presented, but we do not consider this further in this paper for simplicity.

Correctness requires that for all key pairs (mpk, msk) output by Setup, all messages $m \in \{0,1\}^*$, all $0 \le \ell \le L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^\ell$, and all identities $ID \in_* P$, we have

$$\mathsf{Decrypt}\big(\mathsf{KeyDer}(msk, ID), \mathsf{Encrypt}(mpk, P, m)\big) = m$$

with probability one.

3.2. *Security Notions*

We define the security of WIBE schemes analogously to that of HIBE schemes, but with the adversary choosing a challenge pattern instead of an identity to which the challenge ciphertext will be encrypted. To exclude trivial attacks, the adversary is not able to query the key derivation oracle on any identity that matches the challenge pattern, nor is it able to query the decryption oracle on the challenge ciphertext in combination with any identity matching the challenge pattern.

More formally, the IND-WID-CPA security model is defined through the following game, played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary outputs two equal-length messages (m_0, m_1) and a challenge pattern P^*, along with some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the ciphertext $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, P^*, m_\beta)$.
4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information *state*. The adversary is given access to a key derivation oracle as before. The adversary outputs a bit β'.

The adversary wins the game if $\beta = \beta'$ and it never queries the decryption oracle on any identity ID which matches the pattern P^*, i.e. any identity $ID \in_* P^*$. The adversary's advantage is defined as $|2 \cdot \Pr[\,\mathcal{A} \text{ wins}\,] - 1|$.

Definition 9. A (t, q_K, ϵ)-adversary against the IND-WID-CPA security of the WIBE scheme is an algorithm that runs in time at most t, makes at most q_K key derivation oracle queries, and has advantage at least ϵ in the IND-WID-CPA game described above.

In the IND-WID-CCA, the security model is identical to the IND-WID-CPA security model with the exception that the adversary has access to a decryption oracle, which will, on input of an identity ID and a ciphertext C, return $\mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C)$. The adversary wins the game if $\beta = \beta'$, it never queries the decryption oracle on any identity $ID \in_* P^*$, and the adversary does not query the decryption oracle the combination of any identity $ID \in_* P^*$ and the ciphertext C^*. The adversary's advantage is defined as $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition 10. A (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of the WIBE scheme is an algorithm that runs in time at most t, makes at most q_K key derivation oracle queries, makes at most q_D decryption oracle queries, and has advantage at least ϵ in the IND-WID-CCA game described above.

As for the case of HIBEs, we also define a weaker selective-identity (sWID) security notion, in which the adversary commits to the challenge pattern at the beginning of the game, before the master public key is made available. The notions of IND-sWID-CPA and IND-sWID-CCA security are defined analogously to the above. In the random oracle model, the additional parameter q_H denotes the adversary's maximum number of queries to the random oracle, or the total number of queries to all random oracles when it has access to multiple ones.

If the WIBE scheme has a finite message space $\mathcal{M}$, then we may also define a one-way notion for encryption security (OW-WID-CPA). This is formally defined via the following game, played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on input mpk. The adversary is given access to a key derivation oracle as in the IND-WID-CPA game. The adversary outputs a challenge pattern P^* and some state information $state$.
3. The challenger generates $m \xleftarrow{\$} \mathcal{M}$ and computes the ciphertext $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, P^*, m)$.
4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information $state$. The adversary is given access to a key derivation oracle as before. It output a message m'.

The adversary wins the game if $m = m'$ and the adversary never queries the key derivation oracle on an identity $ID \in_* P^*$. The adversary's advantage is defined to be $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition 11. A (t, q_K, ϵ)-adversary against the OW-WID-CPA security of the WIBE scheme is an algorithm that runs in time at most t, makes at most q_K key derivation oracle queries, and has advantage at least ϵ in winning the OW-WID-CPA game described above.

3.3. *Constructing a WIBE from a HIBE*

In order to clarify the relationship between HIBEs and WIBEs, we first point out a generic construction of a WIBE scheme from any HIBE scheme. However, this WIBE scheme has a secret key size that is exponential in the depth of the hierarchy tree. Let "$*$" denote a dedicated bitstring that cannot occur as a user identity. Then the secret key of a user with identity $(ID_1, \ldots, ID_\ell)$ in the WIBE scheme contains the 2^ℓ HIBE secret keys of all patterns matching this identity. For example, the secret key of identity (ID_1, ID_2) contains four HIBE secret keys, namely those corresponding to identities

$$(ID_1, ID_2), (\text{“}*\text{”}, ID_2), (ID_1, \text{“}*\text{”}), (\text{“}*\text{”}, \text{“}*\text{”}).$$

To encrypt to a pattern $(P_1, \ldots, P_\ell)$, one uses the HIBE scheme to encrypt to the identity obtained by replacing each wildcard in the pattern with the "$*$" string, i.e. the identity $(ID_1, \ldots, ID_\ell)$ where $ID_i = \text{“}*\text{”}$ if $P_i = *$ and $ID_i = P_i$ otherwise. The final WIBE ciphertext consists of the pattern and the HIBE ciphertext. Decryption is done by selecting the appropriate secret key from the list and using the decryption algorithm of the HIBE scheme.

Theorem 3. *If there exists a (t, q_K, ϵ) attacker against the IND-WID-CPA security of the WIBE scheme (with hierarchy depth L) then there exists a $(t', 2^L q_K, \epsilon)$-adversary against the IND-HID-CPA security of the corresponding HIBE scheme, where $t' \leq t + 2^L q_K t_K$ and t_K is the time taken to compute a key derivation query. If there exists a (t, q_K, q_D, ϵ) attacker against the IND-WID-CCA security of the WIBE scheme (with hierarchy depth L) then there exists a $(t', 2^L q_K, q_D, \epsilon)$-adversary against the IND-HID-CCA security of the corresponding HIBE scheme, where $t' \leq t + 2^L q_K t_K + q_D t_D$, t_K is the time taken to compute a key derivation query, and t_D is the time taken to compute a decryption query.*

Notice that the appearance of the term 2^L in the security reduction means that this construction is only guaranteed to be secure when the number of levels grows poly-logarithmically in the secure parameter. This restriction occurs in the security analysis of all the HIBE schemes that we consider.

The efficiency of the WIBE scheme obtained with this construction is roughly the same as that of the underlying HIBE scheme, but with the major disadvantage that the size of the secret key is 2^ℓ times that of a secret key in the underlying HIBE scheme. This is highly undesirable for many applications, especially since the secret key may very well be kept on an expensive secure storage device. It is interesting to investigate whether WIBE schemes exist with overhead polynomial in all parameters. We answer this question in the affirmative here by presenting direct schemes with secret key size linear in ℓ. Unfortunately, for all of our schemes, this reduction in key size comes at the cost of linear-size ciphertexts, while the generic scheme can achieve constant-size ciphertexts when underlain by a HIBE with constant ciphertext size, e.g. that of [10].

3.4. *The Relationship Between WIBEs and Generalized Identity-Based Encryption, Fuzzy Identity-Based Encryption, and Attribute Based Encryption*

As we have seen WIBEs are closely related to HIBEs. They are also related to a concept called Generalised Identity-Based Encryption (GIBE) [9]. In a GIBE one has a set of

policies P and a set of roles R. The roles are partially ordered so that a "higher" role can delegate its abilities to a "lower" role. Whether a party can decrypt a ciphertext depends on whether a predicate defined on the set $P \times R$ evaluates to true. In particular, a ciphertext is encrypted to a policy $\pi \in P$, and it can be decrypted by a role ρ if and only if the predicate evaluated on (π, ρ) evaluates to true. It is easy to see that the roles in a GIBE correspond to the identities in a WIBE, whilst the policies correspond to the wildcarded patterns. Hence, a WIBE is a specific example of a GIBE. However, the expressive nature of a GIBE being greater than that of a WIBE comes at a cost, in that one can construct WIBE schemes which are more secure than the equivalent GIBE.

Another related primitive is fuzzy identity-based encryption (FIBE) [23], which allows a ciphertext encrypted to identity ID to be decrypted by any identity ID' that is "close" to ID according to some metric. In the schemes of [23], an identity is a subset containing n elements from a finite universe. Two identities ID and ID' are considered "close" if $|ID \cap ID'| \geq d$ for some parameter d. A FIBE with $n = 2L$ and $d = L$ can be used to construct a WIBE scheme (without hierarchical key derivation) by letting the decryption key for identity $(ID_1, \ldots, ID_\ell)$ correspond to the decryption key for the set

$$\{1\|ID_1, \ldots, \ell\|ID_\ell, (\ell+1)\|\varepsilon, \ldots, L\|\varepsilon, 1\|\text{“}*\text{”}, \ldots, L\|\text{“}*\text{”}\}.$$

Suppose that "$\perp$" is a unique string which cannot occur as a user identity and distinct from "$*$". One can encrypt to pattern $P = (P_1, \ldots, P_\ell)$ by encrypting to the set

$$\{1\|P'_1, \ldots, \ell\|P'_\ell,\ (\ell+1)\|\varepsilon, \ldots, L\|\varepsilon, 1\|\text{“}\perp\text{”}, \ldots, L\|\text{“}\perp\text{”}\},$$

where the $P'_i \leftarrow P_i$ if $i \notin W(P)$ and $P'_i \leftarrow$ "$*$" if $i \in W(P)$. The dummy symbols "$\perp$" are only used to ensure that the size of the encryption set is exactly $2L$ (as required by the definition of the FIBE scheme). We stress that this construction does not give a full WIBE scheme as it does not permit hierarchical key derivation. This also implies that a "parent" identity cannot decrypt message sent to its "children" identities as it cannot derive the key for the child.

Fuzzy-IBE, GIBEs, and WIBEs are themselves examples of a policy-based encryption mechanisms. In such systems, access to encrypted data is provided as long as the recipient has a key (or set of keys) which correspond to some policy. The power of identity-based mechanisms to enable policy-based access control to encrypted data was realised very early on in the history of pairing-based IBE [26]. In recent years, this idea has been formalised under the heading of Attribute Based Encryption.

In Attribute Based Encryption [23], or, more correctly, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [4,17], a recipient is issued keys corresponding to a number of credentials. An encryptor will encrypt a message under a policy, i.e. a set of credentials which are required by any user who wishes to obtain access to the message. Any recipient which has credential key which meet the policy statement has access to the encrypted data. The defining characteristic of CP-ABE is that the policies are embedded in the ciphertexts.

In the context of WIBEs, the policy is that the user should have a key (credential) which matches the pattern. For a pattern such as $(ID_1, *, ID_3)$ this can be interpreted as having a credential for an identity with ID_1 in the first position and an identity with ID_3 in the third position. However, a CP-ABE scheme would offer separate credentials

(keys) for each position, whereas a WIBE compresses all of these credentials in a single key. Hence, ABE is clearly a more powerful concept than a WIBE, as it allows more expressive policies, but WIBE schemes are often simpler to construct.

4. Identity-Based Key Encapsulation with Wildcards

We can also define a notion of Identity-Based Key Encapsulation Mechanism with Wildcards (WIB-KEM). A WIB-KEM consists of the following four algorithms (Setup, KeyDer, Encap, Decap). The algorithms Setup and KeyDer are defined as in the WIBE case. The encapsulation algorithm Encap takes the master public key mpk of the system and a pattern P, and returns (C, K), where $K \in \{0,1\}^\lambda$ is a symmetric key and C is an encapsulation of the key K. Again we assume that the encapsulation includes a public encoding of the pattern P under which the message has been encrypted. Finally, the decapsulation algorithm $\mathsf{Decap}(mpk, d_{ID}, C)$ takes a private key d_{ID} and an encapsulation C, and returns either a secret key K or the error symbol $\perp$.

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

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

The IND-WID game for WIB-KEMs is similar to both the IND-WIB game for WIBEs and the IND-HIB game for HIB-KEMs. The IND-WIB-CCA game is played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary is also given access to a decryption oracle that will, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$ and a ciphertext C, return $\mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C)$. The adversary outputs a challenge pattern P^* and some state information $state$.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$, computes the encapsulation $(C^*, K_0) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$ and chooses a random key $K_1 \xleftarrow{\$} \{0,1\}^\lambda$.
4. The adversary runs $\mathcal{A}_2$ on the input (C^*, K_β) and the state information $state$. The adversary is given access to a key derivation oracle and decryption oracle as before. The adversary outputs a bit β'.

The adversary wins the game if $\beta = \beta'$, it never queries the key derivation oracle on any identity $ID \in_* P^*$, and if it does not query the decryption oracle on the pair (ID, C^*) for some $ID \in_* P^*$ after it receives the challenge ciphertext. As usual, the adversary's advantage is defined to be equal to $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Another common form for writing the advantage of an IND-WID-CCA adversary for a WIB-KEM is given by the following simple lemma.

Lemma 1. *If $\mathcal{A}$ is a (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of the WIB-KEM and β, β' are as in the IND-WID-CCA security game, then*

$$\epsilon = \left|\Pr\left[\beta' = 1 \mid \beta = 1\right] - \Pr\left[\beta' = 1 \mid \beta = 0\right]\right|.$$

Definition 12. A (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of a HIB-KEM is an algorithm that runs in time t, makes at most q_K queries to the key derivation oracle, makes at most q_D queries to the decryption oracle, and has advantage at least ϵ in winning the IND-WID-CCA game described above.

We may combine a WIB-KEM (Setup, KeyDer, Encap, Decap) with a DEM (Enc, Dec) (see Sect. 2.9) to form a complete WIBE scheme (Setup, KeyDer, Encrypt, Decrypt), where the encryption and decryption algorithms are as follows:

- Encrypt(mpk, P^*, m):
  1. Compute $(C_1, K) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$.
  2. Compute $C_2 \leftarrow \mathsf{Enc}(K, m)$.
  3. Output the ciphertext $C = (C_1, C_2)$.
- Decrypt(d_{ID}, C):
  1. Parse C as (C_1, C_2).
  2. Compute $K \xleftarrow{\$} \mathsf{Decap}(d_{ID}, C_1)$. If $K = \perp$ then output $\perp$.
  3. Compute $m \leftarrow \mathsf{Dec}(K, C_2)$.
  4. Output m.

Theorem 4. *If there exists 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 $(t_{\mathcal{B}'}, q_D, \epsilon_{\mathcal{B}'})$-adversary $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ against the IND-CCA security of the DEM such that*:

$$\begin{aligned} t_{\mathcal{B}} &\leq t + q_D t_{\mathsf{Dec}} + t_{\mathsf{Enc}}, \\ t_{\mathcal{B}'} &\leq t + q_D(t_{\mathsf{Dec}} + t_{\mathsf{Decap}} + t_{\mathsf{KeyDer}}) + q_K t_{\mathsf{KeyDer}} \\ &\quad + t_{\mathsf{Encap}} + t_{\mathsf{Setup}}, \\ \epsilon &\leq 2\epsilon_{\mathcal{B}'} + \epsilon_{\mathcal{B}}, \end{aligned}$$

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

Proof. This proof mirrors the proofs of Cramer and Shoup [14] and Bentahar et al. [3]. We prove this result in two stages. First, we change the nature of the security game. Let Game 1 be the normal IND-WID-CCA game for the WIBE scheme. Let Game 2 be the slight adaptation of the IND-WID-CCA game:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary is also given access to a decryption oracle that, on input of an identity ID and a ciphertext C, returns $\mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C)$. The adversary outputs two messages (m_0, m_1) of equal length and a challenge pattern P^*, along with some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and a key $K^* \xleftarrow{\$} \{0,1\}^\lambda$, then computes the ciphertext $(C_1^*, K) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$ and $C_2 \leftarrow \mathsf{Enc}(K^*, m_\beta)$. The challenge ciphertext is $C^* \leftarrow (C_1^*, C_2^*)$.
4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information *state*. The adversary is given access to a key derivation oracle as before. The adversary is also given to a decryption oracle that, on input of an identity ID and a ciphertext $C = (C_1, C_2)$, returns

$$\begin{cases} \mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C) & \text{if } ID \notin_\star P^* \text{ or } C_1 \neq C_1^*, \\ \mathsf{Dec}(K^*, C_2) & \text{if } ID \in_\star P^* \text{ and } C_1 = C_1^*. \end{cases}$$

   The adversary outputs a bit β'.

Note that the only two differences between the game and the IND-WID-CCA game are that a random key is used to compute the challenge ciphertext and to decrypt certain ciphertexts after the challenge ciphertext is issued.

We show that any change in the actions of $\mathcal{A}$ between Game 1 and Game 2 give rise to an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-WID-CCA security of the WIB-KEM. We describe the algorithm $\mathcal{B}_1$ below:

1. $\mathcal{B}_1$ takes as input the master public key mpk.
2. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation oracle query, then $\mathcal{B}_1$ forwards this query to its own oracle and returns the result. If $\mathcal{A}_1$ makes a decryption oracle query on an identity ID and a ciphertext (C_1, C_2), the $\mathcal{B}_1$ forwards C_1 to its decapsulation oracle and receives a key K in return. $\mathcal{B}_1$ returns $\mathsf{Dec}(K, C_2)$ to $\mathcal{A}$. $\mathcal{A}_1$ outputs a challenge pattern P^* and two equal-length messages (m_0, m_1).
3. $\mathcal{B}_1$ outputs the challenge pattern P^*.

The challenger then computes a challenge encapsulation (C_1^*, K^*) where K^* is either the decapsulation of C_1^* or a random key. The algorithm $\mathcal{B}_2$ runs as follows:

1. $\mathcal{B}_2$ takes as input the challenge encapsulation (C_1^*, K^*). $\mathcal{B}_2$ chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the remainder of the challenge ciphertext $C_2^* \leftarrow \mathsf{Enc}(K^*, m_\beta)$.
2. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the challenge ciphertext $C^* = (C_1^*, C_2^*)$. If $\mathcal{A}_2$ makes a key derivation oracle query, then $\mathcal{B}_2$ forwards this query to its own oracle and returns the result. If $\mathcal{A}_2$ makes a decryption oracle query on an identity $ID \in_\star P^*$ and a ciphertext (C_1^*, C_2), then $\mathcal{B}_2$ returns $\mathsf{Dec}(K^*, C_2)$ to $\mathcal{A}_2$. Otherwise, if $\mathcal{A}_2$ makes a decryption oracle query on an identity ID and a ciphertext (C_1, C_2), then $\mathcal{B}_2$

answers the query as before, by querying its own oracle to find the decapsulation of C_1 and decrypting C_2 itself. $\mathcal{A}_2$ outputs a bit β'.
3. If $\beta = \beta'$ then $\mathcal{B}_2$ outputs 1; otherwise $\mathcal{B}_2$ outputs 0.

If K^* is the decapsulation of C_1^* then $\mathcal{B}$ simulates Game 1 for $\mathcal{A}$; whereas if K^* is a random key then $\mathcal{B}$ simulates Game 2 for $\mathcal{A}$. Thus we have,

$$\left|\Pr[\mathcal{A} \text{ wins in Game 1}] - \Pr[\mathcal{A} \text{ wins in Game 2}]\right| = \epsilon_{\mathcal{B}}$$

by virtue of Lemma 1.

However, the security of Game 2 depends only on the (one-time) IND-CCA security of the DEM. We give an algorithm $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ reduces the security of the WIBE in Game 2 to the security of the DEM. We describe the algorithm $\mathcal{B}'_1$ below:

1. $\mathcal{B}'_1$ computes $(mpk, msk) \leftarrow \mathsf{Setup}$.
2. $\mathcal{B}'_1$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation or decryption oracle query, then $\mathcal{B}'_1$ computes the correct answer using its knowledge of the master private key msk. $\mathcal{A}_1$ outputs a challenge pattern P^* and two equal-length messages (m_0, m_1).
3. $\mathcal{B}'_1$ outputs the messages (m_0, m_1).

The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the challenge encryption $C_2^* \xleftarrow{\$} \mathsf{Enc}(K^*, m_\beta)$ using a randomly chosen (and hidden) key $K^* \xleftarrow{\$} \{0,1\}^\lambda$. The algorithm $\mathcal{B}'_2$ runs as follows:

1. $\mathcal{B}'_2$ takes C_2^* as input. $\mathcal{B}'_2$ computes the encapsulation $(C_1^*, K) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$ and sets the challenge ciphertext $C^* \leftarrow (C_1^*, C_2^*)$.
2. $\mathcal{B}'_2$ runs $\mathcal{A}_2$ on the input C^*. If $\mathcal{A}_2$ makes a key derivation oracle query, then $\mathcal{B}'_2$ answers it correctly using its knowledge of the master private key msk. If $\mathcal{A}_2$ makes a decryption oracle query on an identity $ID \in_* P^*$ and a ciphertext (C_1^*, C_2) then $\mathcal{B}'_2$ computes the correct answer by querying its own decryption on C_2 and returning the result. Otherwise, if $\mathcal{A}_2$ makes a decryption oracle query on an identity ID and a ciphertext C, then $\mathcal{B}'_2$ computes the correct answer using its knowledge of the master private key msk. $\mathcal{A}_2$ outputs a bit β'.
3. $\mathcal{B}'_2$ outputs the bit β'.

$\mathcal{B}'$ correctly simulates Game 2 for $\mathcal{A}$. Furthermore, $\mathcal{A}$ wins in Game 2 if and only if $\mathcal{B}$ wins the IND-CCA game for a DEM. Hence,

$$\left|2 \cdot \Pr[\mathcal{A} \text{ wins Game 2}] - 1\right| = \epsilon_{\mathcal{B}'}$$

and so we have that

$$\begin{aligned}
\epsilon &= \left|2 \cdot \Pr[\mathcal{A} \text{ wins Game 1}] - 1\right| \\
&\leq 2 \cdot \left|\Pr[\mathcal{A} \text{ wins in Game 1}] - \Pr[\mathcal{A} \text{ wins in Game 2}]\right| \\
&\quad + \left|2 \cdot \Pr[\mathcal{A} \text{ wins in Game 2}] - 1\right| \\
&= 2\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{B}'}.
\end{aligned}$$

□

5. IND-WID-CPA Secure WIBEs

In this section, we propose several WIBE schemes which are IND-WID-CPA secure, based on three existing HIBE schemes from the Boneh–Boyen family (BB-HIBE, BBG-HIBE, Waters-HIBE). These three direct constructions all utilise a similar technique of modifying a HIBE's ciphertext generation to include some extra data related to each wildcard. The security proof then reduces the security of the resulting WIBE to that of the underlying HIBE. These schemes are all proven secure using the same "projection" technique and so we only prove the security of one scheme (Waters-WIBE) relative to the security of the underlying HIBE (in this case Waters-HIBE). Note, in that due to our earlier comment on the lack of a full security proof for the Waters-HIBE, we obtain a full security theorem only for the cases of the BB- and BBG-based WIBE's.

Each of these three schemes is proven secure, relative to the underlying HIBE, in the standard model; however, two of these schemes are only proven secure in the IND-sWID-CPA model. We therefore give a generic transformation from an IND-sWID-CPA secure scheme to an IND-WID-CPA secure scheme which uses the random oracle model.

5.1. *The Boneh–Boyen WIBE*

Our first construction is based on the slight variant of the BB-HIBE [7] which we prove secure in Sect. 2.6. As with the BB-HIBE scheme, the BB-WIBE makes use of identities which are vectors of elements of $\mathbb{Z}_p$. The scheme is described in Fig. 4. Note that the decryption algorithm can determine if $i \in W(P)$ by checking whether $C_{2,i}$ contains one group element or two.

The BB-WIBE can actually be seen as a close relative of the Waters-WIBE scheme (see Sect. 5.3) with the hash function $F_i(ID_i)$ being defined as

$$F_i(ID_i) = u_{i,0} \cdot u_{i,1}^{ID_i}.$$

Its security properties are different though since the BB-WIBE scheme can be proved secure in the selective-identity model only. We reduce its security to that of the BB-HIBE scheme, which in its turn is proved IND-sHID-CPA secure under the BDDH assumption in Sect. 2.6. The proof of the theorem below is analogous to that of Theorem 7, and hence omitted. One important difference with Theorem 7 is that the reduction from the BB-HIBE scheme is tight: because we prove security in the selective-identity model, we do not lose a factor 2^L due to having to guess the challenge pattern upfront.

Theorem 5. *If there exists a (t, q_K, ϵ)-adversary against the IND-sWID-CPA security of a BB-WIBE (with hierarchy depth L) then there exists a (t', q'_K, ϵ')-adversary against the IND-sHID-CPA security of the BB-HIBE, where*

$$t' \leq t + 2L(1 + q_K) \cdot t_{\text{exp}}, \qquad q'_K \leq q_K \quad \text{and} \quad \epsilon' \geq \epsilon,$$

where t_{exp} is the time required to compute an exponentiation in $\mathbb{G}$.

In terms of efficiency, the BB-WIBE scheme easily outperforms the Waters-WIBE scheme: the master public key contains $2L + 3$ group elements. Encryption to a recipient

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^\alpha$; $h_2 \leftarrow g_2^\alpha$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$, $j = 0, 1$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,1})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot \left(u_{\ell+1,0} \cdot u_{\ell+1,1}^{ID_{\ell+1}}\right)^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, P, m):
Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1, \ldots, \ell$ do
If $i \notin W(P)$ then $C_{2,i} \leftarrow \left(u_{i,0} \cdot u_{i,1}^{P_i}\right)^r$
If $i \in W(P)$ then $C_{2,i} \leftarrow (u_{i,0}^r, u_{i,1}^r)$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_l)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
For $i = 1, \ldots, \ell$ do
If $i \notin W(P)$ then $C'_{2,i} \leftarrow C_{2,i}$
If $i \in W(P)$ then
Parse $C_{2,i}$ as (v_1, v_2)
$C'_{2,i} \leftarrow v_1 \cdot v_2^{ID_i}$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C'_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Fig. 4. The Boneh–Boyen WIBE scheme.

pattern of length ℓ and w wildcards involves $\ell + w + 2$ (multi-)exponentiations and produces ciphertexts containing $\ell + w + 2$ group elements, or $2L + 2$ group elements in the worst case that $\ell = w = L$. Decryption requires the computation of $\ell + 1$ pairings, just like the Waters-WIBE scheme. However, this scheme is outperformed by the BBG-WIBE.

5.2. *The Boneh–Boyen–Goh WIBE*

Our second construction is based on the BBG-HIBE [10] (see Sect. 2.7). The BBG-HIBE scheme has the advantage of constant-sized ciphertexts. Our BBG-WIBE scheme does not have this advantage, but does have the advantage that a pattern with w wildcards leads to a ciphertext with $w + 3$ elements and is secure under the same decisional L-BDHI problem as the BBG-HIBE. Again, identities are considered to be vectors of elements of $\mathbb{Z}_p$ and the scheme is given in Fig. 5.

The BBG-WIBE scheme is significantly more efficient than the Waters-WIBE and BB-WIBE schemes in terms of decryption, and also offers more efficient encryption and shorter ciphertexts when the recipient pattern contains few wildcards. More precisely, the master public key contains $L + 4$ group elements. Encryption to a recipient pattern of length ℓ with w wildcards involves $w + 3$ (multi-)exponentiations and $w + 3$ group elements in the ciphertext, or $L + 3$ of these in the worst case that $\ell = w = L$. Decryption requires the computation of two pairings, as opposed to $\ell + 1$ of these for the Waters-WIBE and BB-WIBE schemes.

Again, the proof of the following theorem is analogous to that of Theorem 7, and hence omitted.

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_i \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$
$mpk \leftarrow (g_1, g_2, h_1, u_0, \ldots, u_L)$
$d_0 \leftarrow h_2$
For $i = 1, \ldots, L+1$ do
  $d_i \leftarrow 1$
$msk \leftarrow (d_0, d_1, \ldots, d_L, d_{L+1})$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_L, d_{L+1})$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot d_{\ell+1}^{ID_{\ell+1}} \cdot \left(u_0 \prod_{i=1}^{\ell} u_i^{ID_i}\right)^{r_{\ell+1}}$
For $i = \ell+2, \ldots, L$ do
  $d'_i \leftarrow d_i \cdot u_i^{r_{\ell+1}}$
$d'_{L+1} \leftarrow d_{L+1} \cdot g_1^{r_{\ell+1}}$
Return $(d'_0, d'_{\ell+2}, \ldots, d'_L, d'_{L+1})$

Algorithm Encrypt(mpk, P, m):
Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
$C_2 \leftarrow \left(u_0 \prod_{i=1, i \notin W(P)}^{\ell} u_i^{P_i}\right)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
$C_4 \leftarrow (u_i^r)_{i \in W(P)}$
Return (P, C_1, C_2, C_3, C_4)

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_{L+1})$
Parse C as (P, C_1, C_2, C_3, C_4)
Parse C_4 as $(v_i)_{i \in W(P)}$
$C'_2 \leftarrow C_2 \prod_{i=1, i \in W(P)}^{\ell} v_i^{ID_i}$
$m' \leftarrow C_3 \cdot \frac{\hat{e}(C'_2, d_{L+1})}{\hat{e}(C_1, d_0)}$
Return m'

Fig. 5. The Boneh–Boyen–Goh WIBE scheme.

Theorem 6. *If there is a (t, q_K, ϵ)-adversary against the IND-sWID-CPA security of the BBG-WIBE (with hierarchy depth L) then there exists a (t', q'_K, ϵ')-adversary against the IND-sHID-CPA security of the BBG-HIBE where*

$$t' \leq t - L(1 + 2q_K) \cdot t_{\text{exp}}, \qquad q'_K \leq q_K, \quad \text{and} \quad \epsilon' \geq \epsilon,$$

where t_{exp} is the time it takes to perform an exponentiation in $\mathbb{G}$.

5.3. *The Waters WIBE*

Our third construction is based on the Waters-HIBE [27] (see Sect. 2.8). As in the HIBE scheme, the WIBE makes use of identities which are n-bit strings and a series of hash functions $(F_1, \ldots, F_L)$ where

$$F_i(ID_i) = u_{i,0} \prod_{j \in [ID_i]} u_{i,j}.$$

The scheme is described in Fig. 6.

In terms of efficiency, the Waters-WIBE compares unfavourably with the BB-WIBE and BBG-WIBE (but (conjecturally) provides stronger security guarantees in the standard model). The master public key of the Waters-WIBE scheme contains $(n+1)L + 3$ group elements. Encrypting to a pattern of length ℓ containing w wildcards comes at the cost of $\ell + nw + 2$ exponentiations and $\ell + nw + 2$ group elements in the ciphertext; in the worst case of $\ell = w = L$ this means $(n+1)L + 2$ exponentiations and group elements. (The pairing $\hat{e}(h_1, g_2)$ can be precomputed.) Decryption requires the computation of $\ell + 1$ pairings.

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^\alpha$; $h_2 \leftarrow g_2^\alpha$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$; $j = 0, \ldots, n$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,n})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer($d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1}$):
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot F_{\ell+1}(ID_{\ell+1})^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, P, m):
Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1 \ldots \ell$ do
If $i \notin W(P)$ then $C_{2,i} \leftarrow F_i(ID_i)^r$
If $i \in W(P)$ then $C_{2,i} \leftarrow (u_{i,0}^r, \ldots, u_{i,n}^r)$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm Decrypt($d_{(ID_1,\ldots,ID_\ell)}, C$):
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
For $i = 1, \ldots, \ell$ do
If $i \notin W(P)$ then $C'_{2,i} \leftarrow C_{2,i}$
If $i \in W(P)$ then
Parse $C_{2,i}$ as $(v_0, \ldots, v_n)$
$C'_{2,i} \leftarrow v_0 \prod_{i \in [ID_i]} v_i$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C'_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Fig. 6. The Waters WIBE scheme.

In terms of efficiency, the Waters-WIBE scheme performs well enough to be considered for use in practice, but definitely leaves room for improvement. The main problem is the dependency of the scheme on n, the bit length of identity strings. In practice, one would typically use the output of a collision-resistant hash function as identity strings, so that $n = 160$ for a reasonable level of security. We note that the techniques of [12,22] could be applied to trade a factor d in efficiency against the loss of a factor of 2^{Ld} in the tightness of the reduction.

We now prove the security of the Waters-WIBE, relative to the security of the Waters-HIBE. This proof provides a template for the proofs of the security theorems for the BB and BBG WIBE's mentioned above. We reduce the security of the Waters-WIBE to the security of the Waters-HIBE. The security of the latter scheme, as has already been mentioned, is believed to reduce to the security of the BDDH problem (see Sect. 2.8).

Theorem 7. *If there exists a (t, q_K, ϵ)-adversary against the IND-WID-CPA security of the Waters-WIBE scheme (with hierarchy depth L) then there exists a (t', q'_K, ϵ')-adversary against the IND-HID-CPA security of the HIBE scheme, where*

$$t' \leq t + Ln(1 + q_K) \cdot t_{\text{exp}}, \qquad q'_K \leq q_K \quad \text{and} \quad \epsilon' \geq \epsilon / 2^L,$$

and t_{exp} is the time it takes to perform an exponentiation in $\mathbb{G}$.

Proof. Suppose there exists a (t, q_K, ϵ)-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the IND-WID-CPA security of the Waters-WIBE scheme. We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-HID-CPA security of the Waters-HIBE.

The intuitive idea behind the proof is that $\mathcal{B}$ guesses the levels in which the challenge pattern contains wildcards. Any query that $\mathcal{A}$ makes is passed by $\mathcal{B}$ to its own oracles after stripping out the levels corresponding to wildcards in the challenge pattern. To this end, we construction a "projection" map $\pi : \{1, \ldots, L\} \to \{1, \ldots, L\}$. Suppose that $\bar{P}^* \in \{\varepsilon, *\}^L$ is $\mathcal{B}$'s guess for the wildcard positions in the challenge pattern. Define $\bar{P}^*_{\leq i}$ to be equal to the first i components of $\bar{P}^*$ and define π as

$$\pi(i) = \begin{cases} 0 & \text{if } i \in W(\bar{P}), \\ i - \left|W(\bar{P}^*_{\leq i})\right| & \text{if } i \notin W(\bar{P}). \end{cases}$$

$\mathcal{B}$ is an adversary against the Waters-HIBE scheme. We denote parameters associated with the HIBE scheme using tildes. The algorithm $\mathcal{B}_1$ runs as follows:

1. $\mathcal{B}_1$ takes as input the master public key of the HIBE scheme $\tilde{mpk} = (\tilde{g}_1, \tilde{g}_2, \tilde{h}_1, \tilde{u}_{1,0}, \ldots, \tilde{u}_{L,n})$.
2. $\mathcal{B}_1$ computes $\bar{P} = (\bar{P}_1, \ldots, \bar{P}_L) \xleftarrow{\$} \{\varepsilon, *\}^L$.
3. $\mathcal{B}_1$ computes the master public key $mpk = (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,n})$ as follows:

$$\begin{aligned} &g_1 \leftarrow \tilde{g}_1 \qquad g_2 \leftarrow \tilde{g}_2 \qquad h_1 \leftarrow \tilde{h}_1, \\ &u_{i,j} \leftarrow \tilde{u}_{\pi(i),j} \quad \text{if } i \notin W(\bar{P}) \text{ and } j = 1, \ldots, n, \\ &u_{i,j} \leftarrow g_1^{\alpha_{i,j}} \quad \text{if } i \in W(\bar{P}),\ j = 1, \ldots, n \text{ and } \alpha_{i,j} \xleftarrow{\$} \mathbb{Z}_p. \end{aligned}$$

4. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation oracle on input $ID = (ID_1, \ldots, ID_\ell)$ then $\mathcal{B}_1$ constructs an identity $\tilde{ID} = (\tilde{ID}_1, \ldots, \tilde{ID}_{\tilde{\ell}})$ by setting $\tilde{ID}_{\pi(i)} \leftarrow ID_i$ for each $i \in W(\bar{P}^*_{\leq \ell})$. $\mathcal{B}_1$ queries its key derivation oracle on $\tilde{ID}$ and receives $(\tilde{d}_0, \ldots, \tilde{d}_{\tilde{\ell}})$. $\mathcal{B}_1$ reconstructs the decryption key $d_{ID} = (d_0, \ldots, d_\ell)$ for ID as:

$$\begin{aligned} &d_0 \leftarrow \tilde{d}_0 \textstyle\prod_{i \in W(\bar{P}^*_{\leq \ell})} \left(u_{i,0} \prod_{j \in [ID_i]} u_{i,j}\right)^{r_i} && \text{for } r_i \xleftarrow{\$} \mathbb{Z}_p, \\ &d_i \leftarrow d_{\pi(i)} && \text{if } i \notin W(\bar{P}^*_{\leq \ell}), \\ &d_i \leftarrow g_1^{r_i} && \text{if } i \in W(\bar{P}^*_{\leq \ell}). \end{aligned}$$

   $\mathcal{B}_1$ returns the key d_{ID} to $\mathcal{A}_1$. $\mathcal{A}_1$ outputs two equal-length messages (m_0, m_1) and a challenge pattern $P^* = (P^*_1, \ldots, P^*_{\ell^*})$.
5. If $\bar{P}^*_{\leq \ell^*}$ and P^* do not have wildcards in exactly the same positions, then $\mathcal{B}_1$ aborts. Otherwise, $\mathcal{B}_1$ computes a challenge identity $\tilde{ID}^* = (\tilde{ID}^*_1, \ldots, \tilde{ID}^*_{\tilde{\ell}^*})$ by setting $\tilde{ID}^*_i \leftarrow P^*_i$ for all $i \notin W(P^*)$. $\mathcal{B}_1$ outputs the challenge identity $\tilde{ID}^*$ and the two messages (m_0, m_1).

The challenger will now encrypt m_β under the identity $\tilde{ID}^*$ using the Waters-HIBE (for $\beta \xleftarrow{\$} \{0, 1\}$). This results in a ciphertext $\tilde{C}^* = (\tilde{C}^*_1, \tilde{C}^*_{2,1}, \ldots, \tilde{C}^*_{2,\tilde{\ell}^*}, \tilde{C}^*_3)$ which is input to the algorithm $\mathcal{B}_2$ described below:

1. $\mathcal{B}_2$ computes a challenge WIBE ciphertext $C^* = (P^*, C_1^*, C_{2,1}^*, \ldots, C_{2,L}^*, C_3^*)$ as follows:

$$\begin{aligned} C_1^* &\leftarrow \tilde{C}_1^*, \\ C_{2,i}^* &\leftarrow \tilde{C}_{2,\pi(i)}^* && \text{if } i \notin W(P^*), \\ C_{2,i}^* &\leftarrow (C_1^{*\alpha_{i,0}}, \ldots, C_1^{*\alpha_{i,n}}) && \text{for } i \in W(P^*), \\ C_3^* &\leftarrow \tilde{C}_3^*. \end{aligned}$$

2. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input C^*. If $\mathcal{A}_2$ makes a key derivation oracle query, then $\mathcal{B}_2$ answer its queries as before. $\mathcal{A}_2$ outputs a guess β'.
3. $\mathcal{B}_2$ outputs β'.

We make several observations about the adversary $\mathcal{B}$. First, note that $\mathcal{B}$ cannot correctly guess the bit β' unless it correctly guesses the locations of the wildcards in the challenge pattern. This happens with probability at least $1/2^L$. Second, we observe that if $\mathcal{B}$ correctly guesses the position of the wildcards in the challenge ciphertext, then $\mathcal{B}$ correctly simulates the key derivation oracle and challenge ciphertext for $\mathcal{A}$. Furthermore, if $\mathcal{B}$ correctly guesses the position of the wildcards in the challenge ciphertext, then any legal key derivation oracle query that $\mathcal{A}$ makes results in a legal key derivation oracle query made by $\mathcal{B}$. This is because for any identity $ID \not\in_\star P^*$ there must exist an index i such that $P_i^* \neq \star$ and $ID_i \neq P_i^*$. Hence, the "projected" identity $\tilde{ID}$ has $\tilde{ID}_{\pi(i)} = ID_i \neq P_i^* = \tilde{ID}_{\pi(i)}^*$. Hence, if $\mathcal{B}$ correctly guesses the position of the wildcards in the challenge ciphertext, then $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins. This leads to the results of the theorem. □

Note that the proof above loses a factor of 2^L in the security reduction. This limits the secure use of the scheme in practice to very small (logarithmic) hierarchy depths, but this was already the case for the Waters-HIBE scheme, which loses a factor $(nq_K)^L$ in its reduction to the BDDH problem. Alternatively, if we only consider patterns with a single sequence of consecutive wildcards, for example $(ID_1, \star, \star, \star, ID_5)$ or $(ID_1, \star, \star)$, then we only lose a factor of L^2 when reducing to the Waters-HIBE scheme. If we consider the selective-identity notion, there is no need to guess the challenge pattern, so we do not lose any tightness with respect to the Waters-HIBE scheme. In addition, the Waters-HIBE scheme would itself also have a tight security reduction to the BDDH problem in the selective-identity notion.

5.4. *Converting Selective-Identity Security to Full Security*

As observed by Boneh and Boyen [7] for the case of IBE schemes and by Boneh, Boyen and Goh [10] for the case of HIBE schemes, any HIBE scheme that is selective-identity (IND-sHID) secure can be transformed into a HIBE scheme that is fully (IND-HID) secure in the random oracle model. The transformation only works for small hierarchy depths though, since the proof loses a factor $O(q_H^L)$ in reduction tightness. We show here that the same transformation works for the case of WIBE schemes at a similar cost of a factor $O(q_H^L)$ in reduction.

Let $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a WIBE scheme with maximum hierarchy depth L. We construct a WIBE scheme $\Pi' = (\mathsf{Setup}, \mathsf{KeyDer}', \mathsf{Encrypt}', \mathsf{Decrypt}')$

where KeyDer', $\mathsf{Encrypt}'$, and $\mathsf{Decrypt}'$ are identical to KeyDer, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ with the exception that the identity/pattern is input to a hash function before it is input to the relevant algorithm. A pattern $P = (P_1, \ldots, P_\ell)$ is transformed into a pattern $P' = (P'_1, \ldots, P'_\ell)$ where

$$P'_i \leftarrow \begin{cases} H_i(P_i) & \text{if } P_i \neq \star, \\ \star & \text{otherwise,} \end{cases}$$

where $H_i : \{0,1\}^* \rightarrow \mathcal{ID}$ (for $1 \leq i \leq L$) are independent hash functions (modelled as distinct random oracles) and $\mathcal{ID}$ is an appropriately sized subset of the allowable identities for the original WIBE scheme.[1]

Theorem 8. *In the random oracle model, suppose that there exists a (t, q_K, q_H, ϵ)-adversary against the IND-WID-CPA security of Π' (with hierarchy depth L) then there exists a (t', q_K, ϵ')-adversary against the IND-sWID-CPA security of Π, where $t' \leq t$ and*

$$\epsilon' \geq \frac{\epsilon}{(L+1)(q_H + q_K L + 1)^L} - \frac{(q_H + q_K L + 1)^2}{|\mathcal{ID}|}.$$

Proof. Suppose there exists a (t, q_K, q_H, ϵ)-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the IND-WID-CPA security of Π'. We construct an IND-sWID-CPA adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2)$ against Π that uses $\mathcal{A}$ as a subroutine. The algorithm $\mathcal{B}_0$ runs as follows:

1. $\mathcal{B}_0$ chooses $\hat{\ell}^* \xleftarrow{\$} \{0, 1, \ldots, L\}$ and $\hat{ctr} \xleftarrow{\$} \{0, 1, \ldots, q_H + q_K L + 1\}$. $\mathcal{B}_0$ computes the challenge pattern $\hat{P}^* \leftarrow (\hat{P}^*_1, \ldots, \hat{P}^*_{\hat{\ell}^*})$ where

$$\hat{P}^*_i \leftarrow \begin{cases} \star & \text{if } \hat{ctr} = 0, \\ ID & \text{if } \hat{ctr} \neq 0 \text{ where } ID \xleftarrow{\$} \mathcal{ID}. \end{cases}$$

   $\mathcal{B}_0$ outputs $\hat{P}^*$.

The challenger now issues the master public key mpk to the adversary. Algorithm $\mathcal{B}_1$ run as follows:

1. $\mathcal{B}_1$ receives the master public key mpk.
2. $\mathcal{B}_1$ initialises a set of lists T_i to answer the random oracle queries for the hash function H_i. These lists are initially empty. For each list, $\mathcal{B}_1$ initialises a counter $ctr_i \leftarrow 1$.
3. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. $\mathcal{B}_1$ answers $\mathcal{A}_1$'s oracle queries as follows:
   - Suppose $\mathcal{A}_1$ queries the random oracle H_i on input ID. If $T_i[ID]$ is defined, then $\mathcal{B}_1$ returns $T_i[ID]$. Otherwise, if $ctr_i = \hat{ctr}_i$, then $\mathcal{B}_1$ sets $T_i[ID] \leftarrow \hat{P}^*_i$, else $\mathcal{B}_1$ sets $T_i[ID] \xleftarrow{\$} \mathcal{ID}$. In either case, $\mathcal{B}_1$ increments ctr_i by one and returns $T_i[ID]$.

[1] These L independent random oracles $(H_1, \ldots, H_L)$ can easily be constructed from a single random oracle H, e.g. by setting $H_i(\cdot) = H([i] \| \cdot)$ where $[i]$ is a fixed-length representation of the integer i.

– Suppose $\mathcal{A}_1$ queries the key derivation oracle on $ID = (ID_1, \ldots, ID_\ell)$. $\mathcal{B}_1$ computes the hashed identity $ID' = (ID'_1, \ldots, ID'_\ell)$ where $ID'_i \leftarrow H_i(ID_i)$ using the random oracle algorithm defined above. $\mathcal{B}_1$ queries its own key derivation oracle on the input ID' and returns to the result to $\mathcal{A}_1$.

$\mathcal{A}_1$ terminates by outputting a challenge pattern $P^* = (P^*_1, \ldots, P^*_{\ell^*})$ and two equal-length messages (m_0, m_1).

4. If $\ell^* \neq \hat{\ell}^*$, if there exists $i \in W(\hat{P}^*)$ such that $P^*_i \neq *$, or if there exists $1 \leq i \leq \ell^*$ such that $i \notin W(\hat{P}^*)$ and $H_i(P^*_i) \neq \hat{P}^*_i$, then $\mathcal{B}_1$ aborts.
5. $\mathcal{B}_1$ outputs the two messages (m_0, m_1).

The challenger computes the challenge ciphertext C^* (which is the encryption of m_β for some randomly chosen $\beta \xleftarrow{\$} \{0, 1\}$). This value is input to algorithm $\mathcal{B}_2$ which runs as follows:

1. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input C^*. If $\mathcal{A}_2$ makes any oracle query, then they are answered as above. $\mathcal{A}_2$ outputs a bit β'.
2. $\mathcal{B}_2$ outputs β'.

$\mathcal{B}$ wins the IND-sWID-CPA game if (1) $\mathcal{A}$ wins the IND-WID-CPA game; (2) $\mathcal{B}$ does not abort because the challenge pattern it outputs is incorrect; (3) $\mathcal{A}$ does not force $\mathcal{B}$ to make an illegal key derivation oracle query. The idea is that the counters $\hat{ctr}_i$ are $\mathcal{B}$'s guess as to which oracle query will define the challenge patterns (where a counter values of $\hat{ctr}_i = 0$ means that position is a wildcard). We require that for each of the hash oracles provides no collisions—i.e. for each $ID \neq ID'$ we have $H_i(ID) \neq H_i(ID')$. Since such a collision could only occur by accident, the probability is bounded by $(q_H + q_K L + 1)^2/|\mathcal{ID}|$ as there exists at most $q_H + q_K L + 1$ entries in all the lists. We exclude the possibility this occurs by losing an additive factor of $(q_H + q_K L)^2/|\mathcal{ID}|$ in the security reduction.

Furthermore, we require that the algorithm $\mathcal{B}$ correctly identifies the pattern that $\mathcal{A}$ outputs. Since the values are chosen at random, we have that $\hat{\ell}^* = \ell^*$ with probability $1/(L+1)$ and that the $\hat{ctr}_i$ value will be correct with probability $1/(q_H + q_K L + 1)$. If $\mathcal{B}$ correctly guesses these values and there are no hash collisions, then $\mathcal{A}$ will never force $\mathcal{B}$ to make an illegal key derivation query. Hence, the result of the theorem holds. □

The above theorem is easily seen to extend to the case of converting an IND-sWID-CCA scheme into an IND-WID-CCA scheme, with an appropriate alteration of the error term in the advantage statement; to take into account the number of decryption oracle queries. Indeed, adversary $\mathcal{B}$ is modified so that when it obtains a decryption query it first hashes the identities to produce a decryption query suitable for $\mathcal{A}$. Such a simulation will fail if and only if the hashed identity is equivalent to the challenge identity for $\mathcal{A}$, but this would imply a collision in the random oracle.

6. IND-WID-CCA Secure WIBEs

In this section, we present constructions for IND-WID-CCA secure WIBEs. We present one generic transform from an IND-WID-CPA WIBE into an IND-WID-CCA WIBE

Algorithm $\mathsf{Encrypt}'(mpk, P, m)$:
$(sk, vk) \xleftarrow{\$} \mathsf{SigGen}$
$P' \leftarrow \mathsf{Encode}(P, vk)$
$C' \xleftarrow{\$} \mathsf{Encrypt}(mpk, P', m)$
$\sigma \xleftarrow{\$} \mathsf{Sign}(sk, (P, C'))$
$C \leftarrow (vk, C', \sigma)$
Return C

Algorithm $\mathsf{Decrypt}'(d_{ID}, C)$:
Parse C as (vk, C', σ)
If $\mathsf{Verify}(vk, C', \sigma) = \perp$ then return $\perp$
For i equals 1 to $|P| - |ID|$
If $P_{|ID|+i} \neq *$ then $ID'_i \leftarrow P_{|ID|+i}$
If $P_{|ID|+i} = *$ then $ID'_i \leftarrow 1^k$
For i equals 1 to $L - |P|$
$ID'_{|P|-|ID|+i} \leftarrow -$
$ID'_{L-|ID|+1} \leftarrow vk$
$d \xleftarrow{\$} \mathsf{KeyDer}(d_{ID}, ID')$
$m \leftarrow \mathsf{Decrypt}(d, C)$
Return m

Fig. 7. The Canetti–Halevi–Katz transform.

based on the Canetti–Halevi–Katz transform [11] and a generic random-oracle-based transform from an OW-WID-CPA WIBE into an IND-WID-CCA WIB-KEM based on a transform of Dent [15].

6.1. *The Canetti–Halevi–Katz Transform*

In this section, we construct a variant of the Canetti–Halevi–Katz transform [11] to convert an IND-WID-CPA secure WIBE with hierarchy depth $L + 1$ into an IND-WID-CCA secure WIBE with hierarchy depth L, using a one-time signature scheme (see Sect. 2.3).

In order to complete this transform, we will make liberal use of an "encoding" function Encode. We will need to restrict the space of allowable identities. We assume that "$-$" represents some fixed, public-known allowable identity for the CPA scheme; we will deliberately exclude "$-$" from the space of allowable identities in the CCA scheme. We assume that 1^k is an allowable identity in the CCA scheme. We then encode a pattern $P = (P_1, \ldots, P_\ell)$ and a verification key vk as the $L + 1$ level identity:

$$\mathsf{Encode}(P, vk) = (P_1, \ldots, P_\ell, -, \ldots, -, vk)\,.$$

We define a similar map for identities (interpreted as patterns without wildcards).

Given an IND-WID-CPA WIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with hierarchy depth $L + 1$, we define an IND-WID-CCA WIBE $\Pi' = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}', \mathsf{Decrypt}')$ with hierarchy depth L. This scheme is described in Fig. 7. The encryption algorithm now produces ciphertexts which are (a) encrypted under the pattern $\mathsf{Encode}(P, vk)$ for a randomly generated $(sk, vk) \xleftarrow{\$} \mathsf{SigGen}$, and (b) signed using sk. The decryption algorithm checks the signature and (if correct) decrypts the ciphertext using a key for an identity which matches $\mathsf{Encode}(P, vk)$ (using the valid identity 1^k in place of wildcards).

Theorem 9. *Suppose that there exists a (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of the WIBE Π' then there exists a $(t_w, q_K + q_D, \epsilon_w)$-adversary*

against the IND-WID-CPA security of Π and a (t_s, ϵ_s)-adversary against the one-time unforgeability of the signature scheme, where

$$t_w \leq t + t_{\mathsf{SigGen}} + t_{\mathsf{Sign}} + q_D(t_{\mathsf{Verify}} + t_{\mathsf{Decrypt}}),$$
$$t_s \leq t + t_{\mathsf{Setup}} + t_{\mathsf{Encrypt}} + q_K \cdot t_{\mathsf{KeyDer}} + q_D \cdot t_{\mathsf{Decrypt}},$$
$$\epsilon \geq \epsilon_w + 2\epsilon_s,$$

where t_{ALG} is the time to execute the algorithm ALG.

Proof. The proof closely follows that of [11]. Let $\mathcal{A}$ be an IND-WID-CCA adversary against the scheme Π'. Suppose P^* is the challenge pattern that $\mathcal{A}$ chooses and (vk^*, C^*, σ^*) is the challenge ciphertext that $\mathcal{A}$ receives during an execution of the attack game. Let FORGE be the event that at some point during its execution $\mathcal{A}$ queries the decryption oracle on an identity $ID \in_* P^*$ and a ciphertext of the form (vk^*, C, σ) such that the algorithm $\mathsf{Verify}(vk^*, C, \sigma)$ returns $\top$. Then we have that $\mathcal{A}$'s advantage is

$$\left|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1/2\right| \leq \left|2 \cdot \Pr[\mathcal{A} \text{ wins} \mid \neg \text{FORGE}] - 1\right| + 2 \cdot \Pr[\text{FORGE}].$$

Claim. $\Pr[\text{FORGE}] \leq \epsilon_s$.

We describe an algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ which breaks the one-time unforgeability of the signature scheme if the event FORGE occurs. The algorithm $\mathcal{B}_1$ runs as follows:

1. $\mathcal{B}_1$ receives vk^* as input.
2. $\mathcal{B}_1$ generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
3. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a decryption or key derivation oracle query, then $\mathcal{B}_1$ answers it using its knowledge of the master private key msk. $\mathcal{B}_1$ outputs a challenge pattern P^* and two equal-length messages (m_0, m_1).
4. If $\mathcal{A}_1$ submitted a decryption oracle query (vk^*, C, σ) for which $\mathsf{Verify}(vk^*, C, \sigma) = \top$, then $\mathcal{B}_1$ chooses a ciphertext $C^* \neq C$ and returns C^*. This is known as the error event.
5. Otherwise, $\mathcal{B}_1$ chooses $\beta \xleftarrow{\$} \{0, 1\}$, computes $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathsf{Encode}(P^*, vk^*), m_\beta)$ and returns C^*.

The challenger then computes a signature σ^* on the "message" C^*. This is input to the algorithm $\mathcal{B}_2$ described as follows:

1. $\mathcal{B}_2$ receives σ^* as input.
2. If the error event occurred during the first phase, then $\mathcal{B}_2$ outputs (C, σ).
3. Otherwise $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input (vk^*, C^*, σ^*). If $\mathcal{A}_2$ makes a key derivation or decryption oracle query, then $\mathcal{B}_2$ answers them using its knowledge of the master private key msk. $\mathcal{B}_2$ outputs a bit β'.
4. If $\mathcal{A}_2$ submitted a decryption oracle query (vk^*, C, σ) for which $\mathsf{Verify}(vk^*, C, \sigma) = \top$, then $\mathcal{B}_2$ outputs (C, σ). Otherwise $\mathcal{B}_2$ outputs the error symbol $\perp$.

Algorithm $\mathcal{B}$ is designed to output a valid forgery if the event FORGE occurs. If $\mathcal{A}_1$ makes a valid decryption oracle query on (vk^*, C, σ), then the error event occurs, and

$\mathcal{B}$ trivially wins. If $\mathcal{A}_2$ makes a valid decryption oracle query on (vk^*, C, σ), then, since $\mathcal{A}_2$ is forbidden from making a decryption oracle query on (vk^*, C^*, σ^*), $\mathcal{B}$ wins after $\mathcal{A}$ finishes its execution. Hence, we have $\epsilon_s = \Pr[\text{FORGE}]$.

Claim. $|2 \cdot \Pr[\mathcal{A} \text{ wins} \mid \neg\text{FORGE}] - 1| \leq \epsilon_w$.

We describe an algorithm $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ which breaks the IND-WID-CPA security of the WIBE scheme Π whenever $\mathcal{A}$ wins and FORGE did not occur. Algorithm $\mathcal{B}'_1$ runs as follows:

1. $\mathcal{B}'_1$ receives a master public key mpk as input.
2. $\mathcal{B}'_1$ generates $(vk^*, sk^*) \xleftarrow{\$} \mathsf{SigGen}$.
3. $\mathcal{B}'_1$ run $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation oracle query on identity ID, then $\mathcal{B}_1$ makes a key derivation oracle query on ID and returns the result. If $\mathcal{A}_1$ makes a decryption oracle query on identity ID and ciphertext (vk, C, σ), then $\mathcal{B}'_1$ returns $\perp$ if $vk = vk^*$ or if $\mathsf{Verify}(vk, C, \sigma) = \perp$. Otherwise, $\mathcal{B}'_1$ computes the extension identity ID' required so that $ID \| ID'$ matches the pattern $\mathsf{Encode}(P, vk)$ as in the decryption algorithm, queries the key extraction algorithm on $ID \| ID'$ to obtain a decryption key d and returns $\mathsf{Decrypt}(d, C)$. $\mathcal{A}_1$ outputs a pattern P^* and two equal-length messages (m_0, m_1).
4. $\mathcal{B}'_1$ returns the challenge pattern $\mathsf{Encode}(P^*, vk^*)$ and the messages (m_0, m_1).

The challenger will pick a random $\beta \xleftarrow{\$} \{0, 1\}$ and computes the ciphertext

$$C^* \xleftarrow{\$} \mathsf{Encrypt}\big(mpk, \mathsf{Encode}(P^*, vk^*), m_\beta\big).$$

This ciphertext is input to the algorithm $\mathcal{B}'_2$ below:

1. $\mathcal{B}'_2$ receives the ciphertext C. $\mathcal{B}'_2$ computes $\sigma^* \xleftarrow{\$} \mathsf{Sign}(sk^*, C^*)$.
2. $\mathcal{B}'_2$ runs $\mathcal{A}_2$ on the ciphertext (vk^*, C^*, σ^*). All oracle queries are answered in exactly the same way as in the first phase. $\mathcal{A}_2$ outputs a bit β'.
3. $\mathcal{B}'_2$ outputs β'.

It is clear that as long as $\mathcal{B}'$ does not make an illegal key derivation oracle query, then $\mathcal{B}'$ wins if and only if $\mathcal{A}$ wins (assuming that FORGE does not occur). $\mathcal{B}'$ may make key derivation oracle queries in response to $\mathcal{A}$ making a key derivation oracle query or a decryption oracle query. If $\mathcal{A}$ makes a decryption oracle query on an identity ID and ciphertext (vk, C, σ) then $\mathcal{B}'$ makes a key derivation query on $\mathsf{Encode}(ID \| ID', vk)$; however, $\mathsf{Encode}(ID \| ID', vk) \not\in_\star \mathsf{Encode}(P^*, vk^*)$ as both encodings are $(L+1)$-bits long and $vk \neq vk^*$ since FORGE does not occur. Furthermore, if $\mathcal{A}$ makes a key derivation oracle query on an identity ID then, by definition, we have $ID \not\in_\star P^*$. We need to show that $ID \not\in_\star \mathsf{Encode}(P^*, vk^*)$. This is true as:

- if $|ID| > |P^*|$ then ID and $\mathsf{Encode}(P^*, vk^*)$ do not agree at level $|P^*| + 1$ (where $\mathsf{Encode}(P^*, vk^*)$ is defined to be "$-$" and ID cannot be defined to be "$-$" since it was excluded from the message space);
- if $|ID| \leq |P^*|$ then $ID \not\in_\star \mathsf{Encode}(P^*, vk^*)$ as $\mathsf{Encode}(P^*, vk^*)_i = P^*_i$ for levels $1 \leq i \leq |ID|$ and $ID \not\in_\star P^*$.

Hence, $\mathcal{A}$ never forces $\mathcal{B}'$ to make an illegal key derivation oracle query and so $\mathcal{B}'$ wins whenever $\mathcal{A}$. Thus,

$$\left|2 \cdot \Pr[\mathcal{A} \text{ wins} \mid \neg\textsc{Forge}] - 1\right| \leq \epsilon_w.$$

A combination of the two claims gives the theorem. □

Applying the Transformation to Waters-WIBE We may optimise the CHK transform in the particular case of the Wates-WIBE scheme describe in Sect. 5.3. In particular, there is no implicit functional reason why we have to fix the encoded identity using "−" strings, as it is possible to determine a key for which the $(L+1)$th level is fixed to vk while leaving lower levels undetermined. In particular, we obtain the scheme given in Fig. 8 which is IND-CCA secure and has depth L. We assume (for simplicity) that verification keys vk are n-bits long.

6.2. *The Dent KEM Transform*

One approaching to building systems secure against adaptive chosen ciphertext attacks is to transform a weakly-secure (OW-WID-CPA) WIBE scheme into a strongly-secure (IND-WID-CCA) WIB-KEM scheme. This obviously gives rise to an IND-WID-CCA WIBE scheme when combined with a suitably secure DEM (see Sects. 2.9 and 4). We apply an analogue of the transformation of Dent [15].

Suppose $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be an OW-WID-CPA WIBE scheme (see Sect. 3.2) with a finite message space $\mathcal{M}$. We assume that the $\mathsf{Encrypt}$ algorithm uses random values taken from a set $\mathcal{R}$. We can write $\mathsf{Encrypt}$ as a deterministic algorithm $C \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$ where $r \xleftarrow{\$} \mathcal{R}$. We require that the scheme satisfies a notion of randomness called γ-uniformity.

Definition 13. A WIBE scheme Π is γ-uniform if for all master public keys mpk that could be output by the key generation algorithm, for all patterns P, for all messages m and ciphertexts C, we have

$$\Pr\left[\mathsf{Encrypt}(mpk, P, m; r) = C\right] \leq \gamma,$$

where the probability is taken over the choice of the randomness r used in the encryption function.

The only difficulty in applying the method of Dent [15] 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. Recall that the set $W(C) = \{i \in \mathbb{Z} : P_i = *\}$ of the pattern P used to encrypt the message, along with the length ℓ of the pattern, is easily derived from the ciphertext. We use this information to give an algorithm P, which on input (ID, C), where C is a ciphertext and $ID = (ID_1, \ldots, ID_\ell)$, returns the pattern $P = (P_1, \ldots, P_\ell)$ where

$$P_i = \begin{cases} * & \text{if } i \in W(C), \\ id_i & \text{if } i \notin W(C). \end{cases}$$

Algorithm Setup:

$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L+1$; $j = 0, \ldots, n$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L+1,n})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer($d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1}$):

Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot F_{\ell+1}(ID_{\ell+1})^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, P, m):

Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1 \ldots \ell$ do
  If $i \notin W(P)$ then $C_{2,i} \leftarrow F_i(ID_i)^r$
  If $i \in W(P)$ then $C_{2,i} \leftarrow (u_{i,0}^r, \ldots, u_{i,n}^r)$
$(sk, vk) \xleftarrow{\$}$ SigGen
$C_{2,L+1} \leftarrow F_{L+1}(vk)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
$\sigma \leftarrow$ Sign$(sk, (P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3))$
Return $(vk, P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3, \sigma)$

Algorithm Decrypt($d_{(ID_1,\ldots,ID_\ell)}, C$):

Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(vk, P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3, \sigma)$
If Verify$(vk, (P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3), \sigma) = \perp$ then
  Return $\perp$
For $i = 1, \ldots, \ell$ do
  If $i \notin W(P)$ then $C'_{2,i} \leftarrow C_{2,i}$
  If $i \in W(P)$ then
    Parse $C_{2,i}$ as $(v_0, \ldots, v_n)$
    $C'_{2,i} \leftarrow v_0 \prod_{i \in [ID_i]} v_i$
$m' \leftarrow C_3 \cdot \frac{\hat{e}(g_1, C_{2,L+1}) \cdot \prod_{i=1}^{\ell} \hat{e}(d_i, C'_{2,i})}{\hat{e}(C_1, F_{L+1}(vk)) \cdot \hat{e}(C_1, d_0)}$
Return m'

Fig. 8. The IND-WID-CCA Waters WIBE scheme.

Algorithm $\mathsf{Encap}(mpk, P)$:
$m \leftarrow \mathcal{M}$
$r \leftarrow H_1(P, m)$
$C \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$
$K \leftarrow H_2(m)$
Return (C, K)

Algorithm $\mathsf{Decap}(d_{ID}, C)$:
$m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$
$P \leftarrow \mathsf{P}(ID, C)$
$r \leftarrow H_1(P, m)$
$C' \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$
If $C = C'$ then return m
Otherwise return $\perp$

Fig. 9. The Dent transform.

We transform the WIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with a finite message space $\mathcal{M}$ and hierarchy depth L into a WIB-KEM scheme $\Pi' = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap})$ using two hash functions:

$$H_1 : \left(\{0,1\}^n \cap \{*\}\right)^* \times \{0,1\}^* \to \mathcal{R}$$

and

$$H_2 : \{0,1\}^* \to \{0,1\}^\lambda.$$

The complete scheme is given in Fig. 9.

Theorem 10. *Suppose that there exists a $(t, q_K, q_D, q_H, \epsilon)$-adversary, in the random oracle model, against the IND-WID-CCA security of the WIB-KEM Π' then there exists a (t', q_K, ϵ')-adversary against the OW-WID-CPA security of the WIBE Π, where*

$$\epsilon' \geq \frac{\epsilon - q_D(|\mathcal{M}|^{-1} + \gamma)}{q_H + q_D},$$
$$t' \leq t + q_H t_{\mathsf{Encrypt}},$$

where t_{Encrypt} is the time taken to perform an encryption, Π has finite message space $\mathcal{M}$, and Π is γ-uniform.

Proof. Suppose there exists a $(t, q_K, q_D, q_H, \epsilon)$-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the IND-WID-CCA security of the WIB-KEM in the random oracle model. We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the OW-WID-CPA security of the WIBE. The algorithm $\mathcal{B}_1$ runs as follows:

1. $\mathcal{B}_1$ receives a master public key mpk.
2. $\mathcal{B}_1$ initialises three lists T_1, E_1, and T_2 which are initially set to be empty.
3. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. $\mathcal{B}_1$ answers $\mathcal{A}_1$'s oracle queries as follows:
   - Suppose $\mathcal{A}_1$ queries the H_1-oracle on input (P, m). If $T_1[P, m]$ is defined, $\mathcal{B}_1$ returns $T_1[P, m]$. Otherwise, $\mathcal{B}_1$ chooses $r \xleftarrow{\$} \mathcal{R}$, sets $T_1[P, m] \leftarrow r$, sets $E_1[P, m] \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$, and returns r.
   - Suppose $\mathcal{A}_1$ queries the H_2-oracle on input r. If $T_2[r]$ is defined, $\mathcal{B}_1$ returns $T_2[r]$. Otherwise, $\mathcal{B}_1$ chooses $K \xleftarrow{\$} \{0,1\}^\lambda$, sets $T_2[r] \leftarrow K$, and returns K.
   - Suppose $\mathcal{A}_1$ queries the key derivation oracle on the input ID. $\mathcal{B}_1$ forwards this request to its own key derivation oracle and returns the result.

– Suppose $\mathcal{A}_1$ queries the decryption oracle on the identity ID and the ciphertext C. $\mathcal{B}_1$ searches the list T_1 for an entry $C = E_1[P, m]$ where $P = \mathsf{P}(ID, C)$. If no such entry exists, then $\mathcal{B}_1$ returns $\perp$. Otherwise, $\mathcal{B}_1$ computes $K \leftarrow H_2(m)$ as above and returns K.

The adversary outputs a challenge pattern P^*.

4. $\mathcal{B}_1$ outputs the challenge pattern P^*.

The challenger then computes a challenge encryption $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, P^*, m^*; r^*)$ for $m^* \xleftarrow{\$} \mathcal{M}$ and $r^* \xleftarrow{\$} \mathcal{R}$. This ciphertext is input to the algorithm $\mathcal{B}_2$:

1. $\mathcal{B}_2$ receives C^*.
2. $\mathcal{B}_2$ generates $K^* \xleftarrow{\$} \{0, 1\}^\lambda$.
3. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input (C^*, K^*). If $\mathcal{A}_2$ queries any oracle, then $\mathcal{B}_2$ answers these queries as before. $\mathcal{A}_2$ outputs a bit β'.
4. $\mathcal{B}_2$ randomly chooses a defined entry for one of the hash functions, either $T_1[P, m]$ or $T_2[m]$, and outputs m.

The basic strategy of this security proof is to take advantage of the fact that the only way that $\mathcal{A}$ can determine if C^* is an encapsulation of K^* is to query the H_2-oracle on m^*. However, we first have to show that the simulated hash function, key derivation, and decryption oracles are consistent with the real IND-WID-CCA game.

The simulated key derivation oracle is perfect, as is the hash function oracle, with the exception that the hash function oracle fails to respond to correctly to an H_1-oracle query on (P^*, m^*) or a H_2-oracle query on m^*. However, the decryption oracle is more problematic. There are two types of error event that can occur with the decryption oracle:

– The decryption oracle will respond incorrectly if $\mathcal{A}_1$ queries the oracle on an identity $ID \in_\star P^*$ and the ciphertext C^*. However, since m^* is information theoretically hidden from $\mathcal{A}_1$, this occurs with probability at most $1/|\mathcal{M}|$.
– The decryption oracle will respond incorrectly if $\mathcal{A}$ queries the decryption oracle on an identity ID and a ciphertext C for which $T_1[P, m]$ is undefined, where $P \leftarrow \mathsf{P}(ID, C)$ and $m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$, but for which

$$C = \mathsf{Encrypt}(mpk, \mathsf{P}(ID, C), m; T_1[P, m])$$

where $T_1[P, m]$ is randomly chosen at the end of the game if it is not defined later by an adversarial query. Since $T_1[P, m]$ is randomly chosen and Π is γ-uniform, we have that this occurs with probability γ.

We have that the probability that either of these events occurs is therefore bounded by $q_D(|\mathcal{M}|^{-1} + \gamma)$. Assuming none of these events occur, we have that the simulation is perfect unless $\mathcal{A}_1$ makes a query which defines the hash function values $T_1[P^*, m^*]$ or $T_2[m^*]$. Since $\mathcal{A}$ cannot determine whether K^* is the correct key for C^* without querying the H_2-oracle on m^*, we have that this event will occur with probability at least $\epsilon - q_D(|\mathcal{M}|^{-1} + \gamma)$. However, if this event occurs, then $\mathcal{B}$ will win the OW-WID-CPA with probability at least $1/(q_H + q_D)$ (as there exists at most $q_H + q_D$ entries on

T_1 and T_2). Hence, $\mathcal{B}$ wins with probability at least

$$\epsilon' \geq \frac{\epsilon - q_D(|\mathcal{M}|^{-1} + \gamma)}{q_H + q_D}$$

which gives the theorem. □

Acknowledgements

We would like to thank Brent Waters, the anonymous referees of ICALP 2006, and the anonymous referees of the Journal of Cryptology for their valuable input. We also thank Mihir Bellare for pointing out the relation between WIBE and fuzzy identity-based encryption.

The first author was supported in part by the French ANR-07-TCOM-013 PACE Project. The sixth author was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government. The eighth author is supported by a Royal Society Wolfson Merit Award. The work in this paper was conducted whilst the third and sixth authors were at École normale supérieure, the fifth author was at the University of Bristol, the sixth author was a Postdoctoral Fellow of the Research Foundation—Flanders (FWO-Vlaanderen), and the second and seventh authors were at Royal Holloway College, University of London.

All authors would like to thank the support of the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT and the ICT Programme under Contract ICT-2007-216646 ECRYPT II. 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.

References

[1] M. Abdalla, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, N. Smart, Identity-based encryption gone wild, in *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, Venice, Italy, July 10–14, 2006, ed. by M. Bugliesi, B. Preneel, V. Sassone, I. Wegener. Lecture Notes in Computer Science, vol. 4052 (Springer, Berlin, 2006), pp. 300–311

[2] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in *ACM CCS 93: 1st Conference on Computer and Communications Security*, Fairfax, Virginia, USA, November 3–5, 1993, ed. by V. Ashby (ACM Press, New York, 1993), pp. 62–73

[3] K. Bentahar, P. Farshim, J. Malone-Lee, N.P. Smart, Generic constructions of identity-based and certificateless KEMs. *J. Cryptol.* **21**(2), 178–199 (2008)

[4] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in *2007 IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 20–23, 2007 (IEEE Computer Society Press, Los Alamitos, 2007), pp. 321–334

[5] J. Birkett, A.W. Dent, G. Neven, J.C.N. Schuldt, Efficient chosen-ciphertext secure identity-based encryption with wildcards, in *ACISP 07: 12th Australasian Conference on Information Security and Privacy*, Townsville, Australia, July 2–4, 2007, ed. by J. Pieprzyk, H. Ghodosi, E. Dawson. Lecture Notes in Computer Science, vol. 4586 (Springer, Berlin, 2007), pp. 274–292

[6] M. Blum, S. Goldwasser, An efficient probabilistic public-key encryption scheme which hides all partial information, in *Advances in Cryptology—CRYPTO'84*, Santa Barbara, CA, USA, August 19–23, 1985, ed. by G.R. Blakley, D. Chaum. Lecture Notes in Computer Science, vol. 196 (Springer, Berlin, 1985), pp. 289–302

[7] D. Boneh, X. Boyen, Efficient selective-ID secure identity based encryption without random oracles, in *Advances in Cryptology—EUROCRYPT 2004*, Interlaken, Switzerland, May 2–6, 2004, ed. by C. Cachin, J. Camenisch. Lecture Notes in Computer Science, vol. 3027 (Springer, Berlin, 2004), pp. 223–238

[8] D. Boneh, M.K. Franklin, Identity based encryption from the Weil pairing. *SIAM J. Comput.* **32**(3), 586–615 (2003)

[9] D. Boneh, M. Hamburg, Generalized identity based and broadcast encryption schemes, in *Advances in Cryptology—ASIACRYPT 2008*, Melbourne, Australia, December 7–11, 2008, ed. by J. Pieprzyk. Lecture Notes in Computer Science, vol. 5350 (Springer, Berlin, 2008), pp. 455–470

[10] D. Boneh, X. Boyen, E.-J. Goh, Hierarchical identity based encryption with constant size ciphertext, in *Advances in Cryptology—EUROCRYPT 2005*, Aarhus, Denmark, May 22–26, 2005, ed. by R. Cramer. Lecture Notes in Computer Science, vol. 3494 (Springer, Berlin, 2005), pp. 440–456

[11] R. Canetti, S. Halevi, J. Katz, Chosen-ciphertext security from identity-based encryption, in *Advances in Cryptology—EUROCRYPT 2004*, Interlaken, Switzerland, May 2–6, 2004, ed. by C. Cachin, J. Camenisch. Lecture Notes in Computer Science, vol. 3027 (Springer, Berlin, 2004), pp. 207–222

[12] S. Chatterjee, P. Sarkar, Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model, in *ICISC: 8th International Conference on Information Security and Cryptology*, Seoul, Korea, December 1–2, 2005, ed. by D. Won, S. Kim. Lecture Notes in Computer Science, vol. 3935 (Springer, Berlin, 2005), pp. 424–440

[13] C. Cocks, An identity based encryption scheme based on quadratic residues, in *Cryptography and Coding, 8th IMA International Conference*, Cirencester, UK, December 17–19, 2001, ed. by B. Honary. Lecture Notes in Computer Science, vol. 2260 (Springer, Berlin, 2001), pp. 360–363

[14] R. Cramer, V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**(1), 167–226 (2003)

[15] A.W. Dent, A designer's guide to KEMs, in *Cryptography and Coding, 9th IMA International Conference*, Cirencester, UK, 2003, ed. by K.G. Paterson. Lecture Notes in Computer Science, vol. 2898 (Springer, Berlin, 2003), pp. 133–151

[16] C. Gentry, A. Silverberg, Hierarchical ID-based cryptography, in *Advances in Cryptology—ASIACRYPT 2002*, Queenstown, New Zealand, December 1–5, 2002, ed. by Y. Zheng. Lecture Notes in Computer Science, vol. 2501 (Springer, Berlin, 2002), pp. 548–566

[17] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in *ACM CCS 06: 13th Conference on Computer and Communications Security*, Alexandria, Virginia, USA, October 30–November 3, 2006, ed. by A. Juels, R.N. Wright, S. De Capitani di Vimercati (ACM Press, New York, 2006), pp. 89–98. Available as Cryptology ePrint Archive Report 2006/309

[18] J. Horwitz, B. Lynn, Toward hierarchical identity-based encryption, in *Advances in Cryptology—EUROCRYPT 2002*, Amsterdam, The Netherlands, April 28 – May 2, 2002, ed. by L.R. Knudsen. Lecture Notes in Computer Science, vol. 2332 (Springer, Berlin, 2002), pp. 466–481

[19] A. Joux, A one round protocol for tripartite Diffie-Hellman. *J. Cryptol.* **17**(4), 263–276 (2004).

[20] E. Kiltz, D. Galindom, Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. *Theor. Comput. Sci.* **410**(47–49), 5093–5111 (2009)

[21] S. Mitsunari, R. Saka, M. Kasahara, A new traitor tracing. *IEICE Trans.* **E85-A**(2), 481–484 (2002)

[22] D. Naccache, Secure and practical identity-based encryption. *IET Inf. Secur.* **1**(2), 59–64 (2007)

[23] A. Sahai, B.R. Waters, Fuzzy identity-based encryption, in *Advances in Cryptology—EUROCRYPT 2005*, Aarhus, Denmark, May 22–26, 2005, ed. by R. Cramer. Lecture Notes in Computer Science, vol. 3494 (Springer, Berlin, 2005), pp. 457–473

[24] R. Sakai, K. Ohgishi, M. Kasahara, Cryptosystems based on pairing, in *SCIS 2000*, Okinawa, Japan, January 2000

[25] A. Shamir, Identity-based cryptosystems and signature schemes, in *Advances in Cryptology—CRYPTO'84*, Santa Barbara, CA, USA, August 19–23, 1985, ed. by G.R. Blakley, D. Chaum. Lecture Notes in Computer Science, vol. 196 (Springer, Berlin, 1985), pp. 47–53

[26] N.P. Smart, Access control using pairing based cryptography, in *Topics in Cryptology—CT-RSA 2003*, San Francisco, CA, USA, April 13–17, 2003, ed. by M. Joye. Lecture Notes in Computer Science, vol. 2612 (Springer, Berlin, 2003), pp. 111–121

[27] B.R. Waters, Efficient identity-based encryption without random oracles, in *Advances in Cryptology—EUROCRYPT 2005*, Aarhus, Denmark, May 22–26, 2005, ed. by R. Cramer. Lecture Notes in Computer Science, vol. 3494 (Springer, Berlin, 2005), pp. 114–127