This is the merged full version of two independent papers that have appeared in the proceedings of ACISP 2006 and SCN 2006.

Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation without Random Oracles

Eike Kiltz[1] David Galindo[2]

August 4, 2006

Abstract

We describe a new and practical identity-based key encapsulation mechanism that is secure in the standard model against chosen-ciphertext (CCA2) attacks. Since our construction is direct and not based on hierarchical identity-based encryption, it is more efficient than all previously proposed schemes. Furthermore, we give the first chosen-ciphertext secure identity-based key encapsulation mechanism with threshold key delegation and decryption in the standard model.

[1] CWI Amsterdam, The Netherlands. Email: kiltz@cwi.nl. URL: http://kiltz.net.

[2] Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands. Email: d.galindo@cs.ru.nl. URL: http://www.cs.ru.nl/~dgalindo/.

Contents

1 Introduction

Identity-Based Encryption and Key Encapsulation. An Identity-Based Encryption (IBE) scheme is a public-key encryption scheme where any string is a valid public key. In particular, email addresses and dates can be public keys. The ability to use identities as public keys avoids the need to distribute public key certificates.

Instead of providing the full functionality of an IBE scheme, in many applications it is sufficient to let sender and receiver agree on a common random session key. This can be accomplished with an *identity-based key encapsulation mechanism* (IB-KEM) as formalized in [7]. Any IB-KEM can be updated to a full IBE scheme by adding a symmetric encryption scheme with appropriate security properties.

After Shamir proposed the concept of IBE in 1984 [39] it remained an open problem for almost two decades to come up with a satisfying construction for it. In 2001, Boneh and Franklin [11] proposed formal security notions for IBE systems and designed a fully functional secure IBE scheme using bilinear maps. This scheme and the tools developed in its design have been successfully applied in numerous cryptographic settings, transcending by far the identity based cryptography framework. IBE is currently in the process of getting standardized — from February 2006 on the new IEEE P1363.3 standard for "Identity-Based Cryptographic Techniques using Pairings" [29] accepts submissions. An alternative but less efficient IBE construction was proposed by Cocks [19] based on quadratic residues.

Both IBE schemes (through the Fujisaki-Okamoto [24] transformation) provide security against *chosen-ciphertext attacks*. In a chosen ciphertext attack, the adversary is given access to a decryption oracle that allows him to obtain the decryptions of ciphertexts of his choosing. Intuitively, security in this setting means that an adversary obtains (effectively) no information about encrypted messages, provided the corresponding ciphertexts are never submitted to the decryption oracle. For different reasons, the notion of chosen-ciphertext security has emerged as the "right" notion of security for encryption schemes. We stress that, in general, chosen-ciphertext security is a much stronger security requirement than chosen-plaintext attacks [4], where in the latter an attacker is not given access to the decryption oracle.

The drawback of the IBE scheme from Boneh-Franklin and Cocks is that security can only be guaranteed in the *random oracle* model [5], i.e. in an idealized world where all parties magically get black-box access to a truly random function. Unfortunately a proof in the random oracle model can only serve as a heuristic argument and has proved to possibly lead to insecure schemes when the random oracles are implemented in the standard model (see, e.g., [15]).

Waters' IBE. To fill this gap Waters [45] presents the first efficient Identity-Based Encryption scheme that is chosen-plaintext secure without random oracles. The proof of his scheme makes use of an algebraic method first used by Boneh and Boyen [8] and security of the scheme is based on the Bilinear Decisional Diffie-Hellman (BDDH) assumption. However, Waters' plain IBE scheme only guarantees chosen-plaintext security.

From 2-level Hierarchical IBE to chosen-chipertext secure IBE. Hierarchical identity-based encryption (HIBE) [28, 26] is a generalization of IBE allowing for hierarchical delegation of decryption keys. Recent results from Canetti, Halevi, and Katz [16], further improved upon by Boneh and Katz [13] show a generic and practical transformation from any chosen-plaintext secure 2-level HIBE scheme to a chosen-ciphertext secure IBE scheme. Since Waters' IBE scheme can naturally be extended to a 2-level HIBE this implies the first chosen-ciphertext secure IBE in the standard model. Key size, as well as the security reduction of the resulting scheme are comparable to the ones from Waters' IBE. However, the transformation involves some symmetric overhead to the ciphertext in form of a one-time signature or a MAC with their respective keys.

1.1 Our Contributions

Our three main contributions can be summarized as follows.

A DIRECT CHOSEN-CIPHERTEXT SECURE IB-KEM BASED ON WATERS' IBE. Our main idea is to enhance (the IB-KEM version of) Waters *chosen-plaintext* secure IBE by adding some redundant information to the ciphertext (consisting of a single group element) to make it *chosen-ciphertext* secure. This information is used to check whether a given IB-KEM ciphertext was "properly generated" by the encryption algorithm or not; if so decryption is done as before, otherwise the ciphertext is simply rejected. Intuitively, this "consistency check" is what gives us the necessary leverage to deal with the stronger chosen-ciphertext attacks. Unfortunately implementing the consistency check is relatively expensive and an equivalent "implicit rejection" method is used to improve efficiency.

This provides the first direct construction of a chosen-ciphertext secure IB-KEM that is not explicitly derived from hierarchical techniques. No exogenous consistency test relying on a symmetric primitive like one-time signatures or MACs is required. Our scheme can be proved secure under the Bilinear Decisional Diffie-Hellman (BDDH) assumption in pairing groups. Chosen-ciphertext security is obtained at sheer minimal cost. Compared to Waters' IB-KEM our scheme comes with a ciphertext overhead of only one single element whereas computational overhead is one more exponentiation for encryption and one pairing plus two exponentiations for decryption. The security reduction is comparable to the one for Waters' scheme, i.e. it introduces only a small additive component.

Using a chosen-ciphertext secure symmetric encryption scheme (also called a data-encapsulation mechanism DEM) our IB-KEM can be extended to a chosen-ciphertext secure IBE scheme [20, 7]. From a theoretical point of view IB-KEM and IBE are equivalent, i.e. the can can be transformed into the other and vice-versa. However, there are a numerous practical reasons to prefer a IB-KEM over an IBE scheme. The biggest advantage is its flexibility, i.e. an IB-KEM completely decouples the key encapsulation from the asymmetric part. So when performing encryption one is free to pick whatever security parameter necessary without changing the size of the message space. For (standard) public-key encryption the same modular approach is incorporated in many standards due to his simplicity and flexibility (see, e.g., [41, 21, 35]). The same is expected to happen in the new IEEE P1363.3 standard for "Identity-Based Cryptographic Techniques using Pairings" [29]. Since the IB-KEM part of a hybrid scheme is independent of the message to be encrypted, many session keys and the corresponding encapsulated ciphertexts can be pre-computed and stored in memory. So if one uses a particularly fast symmetric encryption scheme, such as a stream cipher using the MULTI-S01 mode of operation [27], then one can just about perform real-time symmetric encryption.

Our IB-KEM scheme can be extended in a natural way to obtain a chosen-ciphertext secure HIB-KEM with only one additional element in the ciphertext compared to Waters' chosen-plaintext secure HIB-KEM.

A RIGOROUS GAME-BASED PROOF. The proof of Waters' IBE is already quite complex and has many technical parts that we found pretty hard to verify. Additionally, many recent results [14, 18, 33] already use ingredients of Waters' IBE, some more or less in a "black-box" manner which makes verification nearly impossible without having completely understood the original work. This goes along with a general movement in our field to produce proofs that are increasingly hard to verify [6, 42] and in our opinion this situation has been getting worse and worse. Our additional components to make Waters IB-KEM chosen-ciphertext secure add even more complexity to the proof.

Motivated by this we give a rigorous, games-based proof of our result that can be easily understood and verified.

A THRESHOLD CHOSEN-CIPHERTEXT SECURE IB-KEM. Threshold techniques are applied to cryptographic protocols whenever one wants to decentralize crucial cryptographic operations that need some

additional secret input. The idea is to share this secret input among a number of independent players and only if a sufficiently large fraction of players (determined by a threshold bound) interact (in an honest way), the cryptographic operation can be successfully accomplished. No useful information should be leaked otherwise. We refer to [44] for the numerous applications of threshold cryptography.

In Identity-Based Key Encapsulation there are many operations to which one can possibly apply threshold techniques. Here we consider making key decapsulation and key delegation threshold. We will call such schemes threshold identity-based key encapsulation mechanisms, or threshold IB-KEM for short.

Threshold key delegation means that the user secret key (with respect to some identity) is shared among many players. Sufficiently many players are needed to reconstruct the full user secret key that enables to decapsulate any ciphertext received by the identity.

Threshold decapsulation means that a ciphertext is shared among many players into ciphertext shares. Again, sufficiently many ciphertext shares are needed to combine the shares into the original encapsulated session key. Note that no (shares of the) user secret key is needed to perform the reconstruction of the encapsulated key from its shares. Threshold decapsulation in the context of IBE was first introduced in [3], whereas threshold key delegation was first informally introduced in [12].

In this work we consider chosen-ciphertext secure threshold schemes. In such a chosen-ciphertext attack, the adversary is given access to an oracle that allows him to obtain partial decapsulation shares of ciphertexts and partial user secret key shares of identities of his choosing. Intuitively, security in this setting means that an adversary obtains (effectively) no information about an encapsulated session key, provided he did not receive sufficiently many partial decapsulation/user secret key shares.

Additionally every threshold IB-KEM has to fulfill some "consistency requirements". That is, roughly, it should be impossibly to "abuse" a set of valid shares (i.e. shares that pass their respective consistency tests) to make the scheme inconsistent, which means (for instance) that the same ciphertext is decapsulated into distinct session keys.

Extending [43, 3, 10] we introduce the concept *identity-based key encapsulation with threshold key-delegation and decapsulation* (or short "threshold IB-KEM") and provide full security definitions to model chosen-ciphertext attacks and consistency requirements. To the best of our knowledge we are the first to define a rigorous model for threshold IB-KEM – all previously proposed models either did not consider consistency requirements [3] or were only defined for a weaker threshold functionality (i.e., [17, 10] only consider threshold key-delegation and not threshold decryption).[1]

We give a new construction of a threshold IB-KEM in the above sense. Our scheme is unconditionally consistent and can be proved chosen-ciphertext secure under the BDDH assumption in the standard model. To the best of our knowledge, it is the first threshold IB-KEM proved secure in the standard model.

1.2 Related Work and Comparison

Our technique to obtain the chosen-ciphertext secure IB-KEM is somewhat reminiscent of the method used in [14, 31] to obtain chosen-ciphertext secure *standard encryption*. In fact, as it turns out, our scheme can be seen as a generalization of the standard public-key encryption scheme from [14], i.e. ignoring the "identity-based components" of our scheme simplifies to exactly their scheme.

In the same work [14] a technique is sketched how to avoid the CHK transformation to get a direct chosen-ciphertext secure IB-KEM construction based on Waters' 2-level HIB-KEM. Compared to our IB-KEM, however, this construction has a weaker (quadratic) security reduction and nearly doubles the public key size. In that light our construction can also be viewed as the schemes obtained by

[1] Here we don't claim that the scheme from [3] does not fulfill the necessary consistency requirements. We further remark that the security model from [10] is sufficient for their purpose.

combining the 2-level HIBE scheme obtained from Waters' IBE at the first level and Boneh-Boyen [8] at the second level with certain "direct chosen-ciphertext secure" techniques from [14] to obtain a direct chosen-ciphertext secure IBE scheme.

We will carefully review all known chosen-ciphertext secure IB-KEM constructions, including the above proposal, and make an extensive comparison with our scheme.

It turns out that, to the best of our knowledge, our IB-KEM is the most efficient chosen-ciphertext secure IB-KEM scheme in the standard model based on a standard complexity-theoretic assumption.

Baek and Zheng [3] give an IBE scheme with threshold decryption. The drawback of this scheme is that generic proofs of knowledge (POK) of the equality of two discrete logarithms are used and therefore it inherently relies on random oracles to make the POK non-interactive. Our threshold IB-KEM is the first such scheme that is provably secure in the standard model. Our construction is direct and avoids any form of generic POK. We remark that an existing threshold IBE in the standard model [17] is based on a much weaker security model that in particular avoids all difficulties encountered in [3] that would made POK necessary. More concretely, the scheme in [11] does not deal with chosen ciphertext attacks. In [10] it was shown how to transform any IBE scheme into a threshold (standard) encryption scheme. A special instance of that transformation was already worked out in [14]. A special instance of that transformation was already worked out in [14].

We stress that, however, it is not trivial to extend the techniques used in [10, 14] in order to obtain a full threshold IB-KEM. The main difficulty lies in making the IBE decapsulation algorithm threshold, i.e. to define the partial decapsulation shares. To avoid this difficulties we (roughly) make use of certain linearity properties of the user secret key of the underlying Waters IBE scheme that can be used to define and (in conjunction with bilinear maps) check for consistency of the decapsulation shares.

1.3 Publication Info

An extended abstract of the IB-KEM part of this paper was published in the proceedings of ACISP 2006 [32]; an extended abstract of the threshold IB-KEM part was published in the proceedings of SCN 2006 [25]. This is the merged full version of the two papers containing all proofs.

2 Definitions

2.1 Notation

If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. If S is a set then $s \xleftarrow{\$} S$ denotes the operation of picking an element s of S uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \xleftarrow{\$} \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting z be the output. We write $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ and by $z \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$, and letting z be the output.

2.2 Identity Based Key Encapsulation

An *identity-based key-encapsulation mechanism* (IB-KEM) scheme [39, 12] $\mathcal{IBKEM} = (\mathsf{IBKEMkg}, \mathsf{IBKEMkeyder}, \mathsf{IBKEMenc}, \mathsf{IBKEMdec})$ consists of four polynomial-time algorithms. Via $(pk, msk) \xleftarrow{\$} \mathsf{IBKEMkg}(1^k)$ the randomized key-generation algorithm produces master keys for security parameter $k \in \mathbb{N}$; via $sk[id] \xleftarrow{\$} \mathsf{IBKEMkeyder}(msk, id)$ the master computes the secret key for identity id; via

$(C, K) \xleftarrow{\$} \mathsf{IBKEMenc}(pk, id)$ a sender creates a random session key K and a corresponding ciphertext C with respect to identity id; via $K \leftarrow \mathsf{IBKEMdec}(sk, C)$ the possessor of secret key sk decapsulates ciphertext C to get back a the session key K. Associated to the scheme is a key space KeySp. For consistency, we require that for all $k \in \mathbb{N}$, all identities id, and all $(C, K) \xleftarrow{\$} \mathsf{IBKEMenc}(pk, id)$, we have $\Pr[\mathsf{IBKEMdec}(\mathsf{IBKEMkeyder}(msk, id), C) = K] = 1$, where the probability is taken over the choice of $(pk, msk) \xleftarrow{\$} \mathsf{IBKEMkg}(1^k)$, and the coins of all the algorithms in the expression above.

Let $\mathcal{IBKEM} = (\mathsf{IBKEMkg}, \mathsf{IBKEMkeyder}, \mathsf{IBKEMenc}, \mathsf{IBKEMdec})$ be an IB-KEM with associated key space KeySp. To an adversary $\mathcal{A}$ we associate the following experiment:

$$
\begin{array}{l}
\textbf{Experiment } \mathbf{Exp}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k) \\
\quad (pk, msk) \xleftarrow{\$} \mathsf{IBKEMkg}(1^k) \\
\quad (id^*, state) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer}(\cdot),\textsc{Dec}(\cdot,\cdot)}(\mathtt{find}, pk) \\
\quad K_0^* \xleftarrow{\$} \mathsf{KeySp}\,;\ (C^*, K_1^*) \xleftarrow{\$} \mathsf{IBKEMenc}(pk, id) \\
\quad \gamma \xleftarrow{\$} \{0,1\}\,;\ K^* \leftarrow K_\gamma^* \\
\quad \gamma' \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer},\textsc{Dec}}(\mathtt{guess}, K^*, C^*, state) \\
\quad \text{If } \gamma \neq \gamma' \text{ then return 0 else return 1}
\end{array}
$$

The oracle $\textsc{KeyDer}(id)$ returns $sk[id] \xleftarrow{\$} \textsc{KeyDer}(msk, id)$ with the restriction that $\mathcal{A}$ is not allowed to query oracle $\textsc{KeyDer}(\cdot)$ for the target identity id^*. The oracle $\textsc{Dec}(id, C)$ first computes $sk[id] \xleftarrow{\$} \textsc{KeyDer}(msk, id)$ as above and then returns $K \leftarrow \mathsf{IBKEMdec}(sk[id], id, C)$ with the restriction that in the guess stage $\mathcal{A}$ is not allowed to query oracle $\textsc{Dec}(\cdot,\cdot)$ for the tuple (id^*, C^*). *state* is some internal state information of adversary $\mathcal{A}$ and can be any (polynomially bounded) string. We define the advantage of $\mathcal{A}$ in the IND-CCA experiment as

$$
\mathbf{Adv}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k) \;=\; \left| \Pr\left[\mathbf{Exp}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k) = 1 \right] - \frac{1}{2} \right| .
$$

An IB-KEM $\mathcal{IBKEM}$ is said to be *secure against adaptively-chosen ciphertext attacks* if the advantage functions $\mathbf{Adv}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k)$ is a negligible function in k for all polynomial-time adversaries $\mathcal{A}$.

We remark that our security definition is given with respect to "full-identity" attacks, as opposed to the much weaker variant of "selective-identity" attacks where the adversary has to commit to its target identity id^* in advance, even before seeing the public key.

2.3 Target Collision Resistant Hash Functions

Let $\mathcal{F} = (\mathsf{TCR}_s)_{s \in S}$ be a family of hash functions for security parameter k and with seed $s \in S = S(k)$. $\mathcal{F}$ is said to be *collision resistant* if, for a hash function $\mathsf{TCR} = \mathsf{TCR}_s$ (where the seed is chosen at random from S), it is infeasible for an efficient adversary to find two distinct values $x \neq y$ such that $\mathsf{TCR}(x) = \mathsf{TCR}(y)$.

A weaker notion is that of *target collision resistant hash functions*. Here it should be infeasible for an efficient adversary to find, given a randomly chosen element x and a randomly drawn hash function $\mathsf{TCR} = \mathsf{TCR}_s$, a distinct element $y \neq x$ such that $\mathsf{TCR}(x) = \mathsf{TCR}(y)$. (In collision resistant hash functions the value x may be chosen by the adversary.) Such hash functions are also called *universal one-way hash functions* [34] and can be built from arbitrary one-way functions [34, 36]. We define (slightly informal)

$$
\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) = \Pr[\mathcal{H} \text{ finds a collision in } \mathsf{TCR}].
$$

Hash function family TCR is said to be a *target collision resistant* if the advantage function $\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}$ is a negligible function in k for all polynomial-time adversaries $\mathcal{H}$.

In practice, to build a target collision resistant hash function TCR, one can use a dedicated cryptographic hash function, like SHA-1 [38]. For that reason and to simplify our presentation, in what follows we will consider the hash function TCR to be a fixed function.

3 Assumptions

3.1 Parameter generation algorithms for Bilinear Groups.

All pairing based schemes will be parameterized by a *pairing parameter generator*. This is a PTA $\mathcal{G}$ that on input 1^k returns the description of an multiplicative cyclic group $\mathbb{G}_1$ of prime order p, where $2^k < p < 2^{k+1}$, the description of a multiplicative cyclic group $\mathbb{G}_T$ of the same order, and a non-degenerate bilinear pairing $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. See [12] for a description of the properties of such pairings. We use $\mathbb{G}_1^*$ to denote $\mathbb{G}_1 \setminus \{0\}$, i.e. the set of all group elements except the neutral element. Throughout the paper we use $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_T, p, \hat{e})$ as shorthand for the description of bilinear groups.

3.2 The BDDH assumption

Let $\mathcal{PG}$ be the description of pairing groups. Consider the following problem first considered by Joux [30] and later formalized by Boneh and Franklin [12]: Given $(g, g^a, g^b, g^c, W) \in \mathbb{G}_1^4 \times \mathbb{G}_T$ as input, output yes if $W = \hat{e}(g,g)^{abc}$ and no otherwise. More formally, to a parameter generation algorithm for pairing-groups $\mathcal{G}$ and an adversary $\mathcal{B}$ we assotiate the following experiment.

Experiment $\mathbf{Exp}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k)$
$\mathcal{PG} \xleftarrow{\$} \mathcal{G}(1^k)$
$a, b, c, w \xleftarrow{\$} \mathbb{Z}_p^*$
$\beta \xleftarrow{\$} \{0,1\}$
If $\beta = 1$ then $W \leftarrow \hat{e}(g,g)^{abc}$ else $W \leftarrow \hat{e}(g,g)^{w}$
$\beta' \xleftarrow{\$} \mathcal{B}(1^k, \mathcal{PG}, g, g^a, g^b, g^c, W)$
If $\beta \neq \beta'$ then return 0 else return 1

We define the advantage of $\mathcal{B}$ in the above experiment as

$$\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k) \;=\; \left| \Pr\left[\mathbf{Exp}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k) = 1 \right] - \frac{1}{2} \right| .$$

We say that the *Bilinear Decision Diffie-Hellman (BDDH) assumption relative to generator* $\mathcal{G}$ holds if $\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}$ is a negligible function in k for all PTAs $\mathcal{B}$. The BDDH assumption was shown to hold in the generic group model in [9].

4 A chosen-ciphertext secure IB-KEM based on BDDH

In this section we present our new chosen-ciphertext secure IB-KEM. From now on let $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_T, p, \hat{e}, g)$ be public system parameters obtained by running the group parameter algorithm $\mathcal{G}(1^k)$.

4.1 Waters' Hash

We review the hash function $\mathsf{H} : \{0,1\}^n \to \mathbb{G}_1$ used in Waters' identity based encryption schemes [45]. On input of an integer n, the randomized hash key generator $\mathsf{HGen}(\mathbb{G}_1)$ chooses $n+1$ random groups elements $h_0, \ldots, h_n \in \mathbb{G}_1$ and returns $h = (h_0, h_1, \ldots, h_n)$ as the public description of the hash

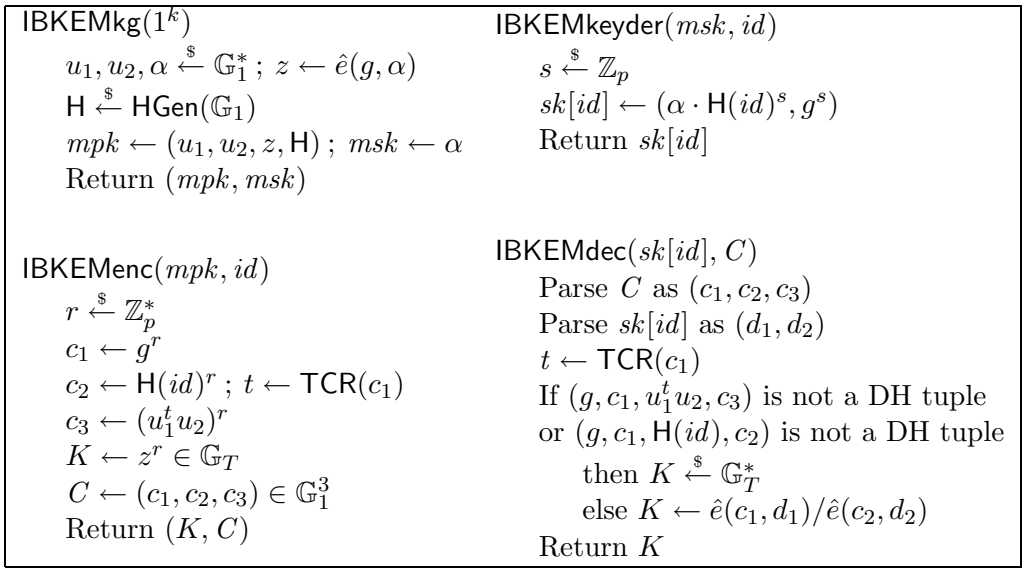


Figure 1: Our chosen-ciphertext secure IB-KEM.

function. The hash function $\mathsf{H} : \{0,1\}^n \to \mathbb{G}_1^*$ is evaluated on a string $id = (id_1, \ldots, id_n) \in \{0,1\}^n$ as the product

$$\mathsf{H}(id) = h_0 \prod_{i=1}^{n} h_i^{id_i}.$$

4.2 The IB-KEM Construction

Let $\mathsf{TCR} : \mathbb{G}_1 \to \mathbb{Z}_p$ be a target collision-resistant hash function (which we assume to be included in the system parameters). Our IB-KEM with identity space $\mathsf{IDSp} = \{0,1\}^n$ ($n = n(k)$) and key space $\mathsf{KeySp} = \mathbb{G}_T$ is depicted in Figure 1.

A tuple $(g, c_1, u_1^t u_2, c_3)$ is a Diffie-Hellman tuple[2] if $\hat{e}(g, c_3) = \hat{e}(u_1^t u_2, c_1)$. Analogously, $(g, c_1, \mathsf{H}(id), c_2)$ is a Diffie-Hellman tuple if $\hat{e}(g, c_2) = \hat{e}(\mathsf{H}(id), c_1)$. Therefore the check in the decapsulation algorithm $\mathsf{IBKEMdec}$ can be implemented by evaluating the bilinear map four times.

We now show correctness of the scheme, i.e. that the session key K computed in the encapsulation algorithm matches the K computed in the decapsulation algorithm. A correctly generated ciphertext for identity id has the form $C = (c_1, c_2, c_3) = (g^r, \mathsf{H}(id)^r, (u_1^t u_2)^r)$ and therefore $(g, c_1, u_1^t u_2, c_3) = (g, g^r, u_1^t u_2, (u_1^t u_2)^r)$ is always a DH tuple. A correctly generated secret key for identity id has the form $sk[id] = (d_1, d_2) = (\alpha \cdot \mathsf{H}(id)^s, g^s)$. Therefore the decapsulation algorithm computes the session key K as

$$\begin{aligned} K &= \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \\ &= \hat{e}(g^r, \alpha \mathsf{H}(id)^s)/\hat{e}(\mathsf{H}(id)^r, g^s) \\ &= \hat{e}(g^r, \alpha) \cdot \hat{e}(g^r, \mathsf{H}(id)^s)/\hat{e}(\mathsf{H}(id)^r, g^s) \\ &= z^r \cdot \hat{e}(g^s, \mathsf{H}(id)^r)/\hat{e}(\mathsf{H}(id)^r, g^s) \\ &= z^r, \end{aligned}$$

as the key computed in the encapsulation algorithm. This shows correctness.

[2] A tuple $(h, h^a, h^b, h^c) \in \mathbb{G}_1^4$ is said to be a *Diffie-Hellman tuple* if $ab = c \bmod p$.

Let $C = (c_1, c_2, c_3) \in \mathbb{G}_1^3$ be a (possibly malformed) ciphertext. Ciphertext C is called *consistent* (w.r.t the public key pk and identity id) if $(g, c_1, u_1^t u_2, c_3)$ and $(g, c_1, \mathsf{H}(id), c_2)$ are Diffie-Hellman tuples, where $t = \mathsf{TCR}(c_1)$. Note that any ciphertext properly generated by the encapsulation algorithm is always consistent. The decapsulation algorithm tests for consistency of the ciphertext. Note that this consistency test can be performed by anybody knowing the public-key. We call this property "public verification" of the ciphertext. In the words of [1] the IB-KEM ciphertext is not *anonymous*.

4.3 More Efficient Decapsulation

We now describe an alternative decapsulation algorithm which is more efficient (but less intuitive). The idea is to make the Diffie-Hellman consistency check implicit in the computation of the key K. This is done by choosing a random values $r_1, r_2 \in \mathbb{Z}_p^*$ and computing the session key as

$$K \leftarrow \frac{\hat{e}(c_1, d_1 \cdot (u_1^t u_2)^{r_1} \cdot H(id)^{r_2})}{\hat{e}(c_2, d_2 \cdot g^{r_2}) \cdot \hat{e}(g^{r_1}, c_3)} \ .$$

We claim that this is equivalent to first checking for consistency and then computing the key as $K \leftarrow \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2)$ as in the original decapsulation algorithm.

To prove this claim we define the functions $\Delta_1(C) = \hat{e}(c_1, u_1^t u_2)/\hat{e}(g, c_3)$ and $\Delta_2(C) = \hat{e}(\mathsf{H}(id), c_1)/\hat{e}(g, c_2)$. Then $\Delta_1(C) = \Delta_2(C) = 1$ if and only if C is consistent. Consequently, for random $r_1, r_2 \in \mathbb{Z}_p^*$, $K = \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \cdot (\Delta_1(C))^{r_1} \cdot (\Delta_2(C))^{r_2} \in \mathbb{G}_T^*$ evaluates to $\hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \in \mathbb{G}_T$ if C is consistent and to a random group element otherwise. As in the original decapsulation algorithm. The claim then follows by

$$\begin{aligned} K &= \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \cdot \Delta_1(C)^{r_1} \cdot (\Delta_2(C))^{r_2} \\ &= \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \cdot (\hat{e}(c_1, u_1^t u_2)/\hat{e}(g, c_3))^{r_1} \cdot (\hat{e}(\mathsf{H}(id), c_1)/\hat{e}(g, c_2))^{r_2} \\ &= \frac{\hat{e}(c_1, d_1 (u_1^t u_2)^{r_1} H(id)^{r_2})}{\hat{e}(c_2, d_2 \cdot g^{r_2}) \cdot \hat{e}(g^{r_1}, c_3)} \ . \end{aligned}$$

We remark that the alternative decapsulation algorithm roughly saves two pairing operation (for the cost of a couple of exponentiations).

4.4 Security

Theorem 4.1 Assume TCR is a target collision resistant hash function. Under the Bilinear Decisional Diffie-Hellman (BDDH) assumption relative to generator $\mathcal{G}$, the IB-KEM from Section 4.2 is secure against chosen-ciphertext attacks.

In particular, given an adversary $\mathcal{A}$ attacking the chosen-ciphertext security of the IB-KEM with advantage $\varepsilon_{\mathcal{A}} = \mathbf{Adv}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}$ and running time $\mathbf{Time}_{\mathcal{A}}(k)$ we construct an adversary $\mathcal{B}$ breaking the BDDH assumption with advantage $\varepsilon_{\mathcal{B}} = \mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k)$ and running time $\mathbf{Time}_{\mathcal{B}}(k)$ with

$$\begin{aligned} \varepsilon_{\mathcal{B}}(k) &\geq \frac{\varepsilon_{\mathcal{A}}(k) - \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)}{8(n+1)q} - q/p; \\ \mathbf{Time}_{\mathcal{B}}(k) &\leq \mathbf{Time}_{\mathcal{A}} + \tilde{\mathcal{O}}(nq \cdot \varepsilon_{\mathcal{A}}^{-2}(k)), \end{aligned}$$

where q is an upper bound on the number of key derivation/decryption queries made by adversary $\mathcal{A}$.

A game-based proof of Theorem 4.1 will be given in Appendix A. The proof is mainly based on the one given by Waters [45]. However, we have to do some important modifications to be able to deal with chosen-ciphertext attacks.

Intuitively, security can be best understood by observing that our scheme is a generalization of Waters' (chosen-plaintext secure) IBE scheme, as well as of the chosen-ciphertext secure *public-key* encapsulation scheme from [14]. We remark that unfortunately, there does not seem to be a way to derive security of our IBE scheme directly from security of either of the two schemes and hence details of the whole proof have to be worked out from scratch.

Relation to Waters' IBE scheme. The ciphertext in our scheme is basically identical to the ciphertext from Waters' IBE scheme plus one redundant element (the element c_3) used to check for consistency of the ciphertext. Hence Waters' IBE scheme is obtained by ignoring the computation of c_3 in encapsulation as well as the consistency check in decapsulation.

Relation to the Encryption Scheme from BMW. Clearly, IB-KEM implies (standard) public-key encapsulation by simply ignoring all operations related to the identity. We remark that viewed in this light (i.e. ignoring the element c_2 in encapsulation/decapsulation and ignoring the key derivation algorithm) our IB-KEM can be simplified to the chosen-ciphertext secure encryption scheme recently proposed by Boyen, Mei, and Waters [14].

5 IB-KEM with threshold key-delegation and decapsulation

We start with some history and motivation. Threshold key-delegation for IBE was introduced in [11, 10]. The idea is that the master key msk is distributed among different secret key generation players. Given a master-key share sk_i, each player can compute a partial secret key $sk[id]_i$ for the user with identity id. Finally, a sufficiently large fraction of partial user secret keys is needed to reconstruct the user secret key $sk[id]$.

In contrast, in the model given in [3] the master key is not shared but only the user secret key $sk[id]$. Then partial user secret key shares $sk[id]_i$ are distributed among a number of decryption players. Given the share $sk[id]_i$, the i-th decryption player can compute a partial decryption share C_i of a given an ciphertext C. A sufficiently large fraction of correctly generated ciphertext shares is needed to finally reconstruct the message. No information about the message should be leaked otherwise.

Our model of threshold IB-KEM is aimed at capturing the functionalities of both threshold key-delegation IBE and threshold decryption IBE. That is, in a threshold IB-KEM the players can act at the same time as private key generation players and decapsulation players, so that they can choose which role they want to assume depending on the application. Therefore, an encapsulation C sent to user id can be decapsulated either by reconstructing the user secret key $sk[id]$, or by joining together a large enough fraction of decapsulation shares C_i.

5.1 Definitions

We now give a formal definition of the functionality of a threshold IB-KEM. A threshold IB-KEM with participating players $1, \ldots, m$ consists of nine polynomial-time algorithms $\mathcal{TIBKEM} = (\mathsf{TIBKEMkg}, \mathsf{TIBKEMkey.Share}, \mathsf{TIBKEMkey.Vfy}, \mathsf{TIBKEMkey.Combine}, \mathsf{TIBKEMenc}, \mathsf{TIBKEMdec}, \mathsf{TIBKEMdec.Share}, \mathsf{TIBKEMdec.Vfy}, \mathsf{TIBKEMdec.Combine})$. Via $(pk, vk, sk) \xleftarrow{\$} \mathsf{TIBKEMkg}(1^k, l, m)$ the randomized key-generation algorithm produces a public key pk, a public verification key vk, and the m master-key shares $sk = (sk_i)_{1 \le i \le m}$ for security parameter $k \in \mathbb{N}$ and threshold parameter l; via $(C, K) \xleftarrow{\$} \mathsf{TIBKEMenc}(pk, id)$ a sender creates a random session key K and a corresponding ciphertext C with re-

spect to identity id; via $sk[id]_i \xleftarrow{\$} \mathsf{TIBKEMkey.Share}(pk, i, id, sk_i)$ the ith share $sk[id]_i$ of the user secret key $sk[id]$ is generated; via $\{\mathtt{accept}, \mathtt{fail}\} \leftarrow \mathsf{TIBKEMkey.Vfy}(vk, i, id, sk[id]_i)$ the validity of the ith user secret key share $sk[id]_i$ is verified; via $\{sk[id], \mathtt{fail}\} \leftarrow \mathsf{TIBKEMkey.Combine}(pk, vk, id, (sk[id]_i)_{i \in I_r})$ sufficiently many valid user secret key shares $\{(sk[id]_i\}_{i \in I_r}$ are combined to reconstruct the user secret key $sk[id]$. The set of players I_r is called the *user secret key reconstruction set*. Via $\{K, \mathtt{fail}\} \leftarrow \mathsf{TIBKEMdec}(pk, sk[id], C)$ the possessor of the user secret key $sk[id]$ decapsulates the ciphertext C; via $\{(i, C_i), \mathtt{fail}\} \leftarrow \mathsf{TIBKEMdec.Share}(pk, vk, id, i, sk[id]_i, C)$ the possessor of the ith user secret key share $sk[id]_i$ partially decapsulates the ciphertext C encrypted with respect to id to get back the ith decapsulation share C_i; via $\{\mathtt{accept}, \mathtt{fail}\} \leftarrow \mathsf{TIBKEMdec.Vfy}(i, pk, vk, id,\ C_i, C)$ it can be publicly verified if the ith decapsulation share C_i is valid; via $\{M, \mathtt{fail}\} \leftarrow \mathsf{TIBKEMdec.Combine}(pk, vk, id, (C_i)_{i \in I'_r}, C)$ sufficiently many valid decapsulation shares $\{C_i\}_{i \in I'_r}$ are combined to reconstruct the session key K. The set of players I'_r is called the *session key reconstruction set* (and may be distinct from I_r).

Roughly speaking, for correctness we require that all correctly generated shares pass their respective verification tests. Furthermore, any set of at least l honest players holding shares of a common identity id should be able to correctly operate the threshold IB-KEM, i.e. they should be able to reconstruct the user secret key $sk[id]$, or alternatively to decapsulate any correctly generated encapsulation sent to id. We say that a user secret key or decapsulation share is *correctly generated* if it has been obtained by following the protocol specification. Moreover, a user secret key or decapsulation share is said to be *valid* if it passes the corresponding verification test.

Each threshold IB-KEM naturally has to fulfull security and consistency requirements. In terms of security we have to extend the security models in [3, 10] to our setting, meaning that an adversary, in addition to the master-key shares for corrupted players, gets access to oracles for user secret key and decapsulation shares. Regarding consistency, we must recall that often one wants threshold key-delegation (resp. threshold decryption) to be robust, namely if the reconstruction of $sk[id]$ (resp. threshold decapsulation of a valid encapsulation C) fails, it is useful to detect the players that supplied invalid partial user secret keys (resp. invalid partial decapsulation shares). This also means that it should be impossibly to "abuse" shares that passed their respective consistency tests (i.e. shares that are valid) to make the scheme inconsistent, for instance by decapsulating the same encapsulation into distinct session keys. We will call this property consistency of the threshold IB-KEM.

5.2 Security requirements

Formally, we associate to a threshold IB-KEM $\mathcal{TIBKEM}$ and an adversary $\mathcal{A}$ the experiment $\mathbf{Exp}^{\textit{tibkem}\text{-cca}}_{\mathcal{TIBKEM},\mathcal{A}}$ as follows:

Experiment $\mathbf{Exp}^{\textit{tibkem}\text{-cca}}_{\mathcal{TIBKEM},\mathcal{A}}(k)$

$(I_c, state_0) \xleftarrow{\$} \mathcal{A}(1^k, \mathtt{init})$ //adversary outputs the set of corrupted users
$(pk, sk, vk) \xleftarrow{\$} \mathsf{TIBKEMkg}(1^k, l, m)$
$(id^*, state) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyShare}(\cdot,\cdot),\textsc{DecShare}(\cdot,\cdot,\cdot)}(\mathtt{find}, pk, vk, \{sk_i\}_{i \in I_c}, state_0)$
$K_0^* \xleftarrow{\$} \mathsf{KeySp}\ ;\ (C^*, K_1^*) \xleftarrow{\$} \mathsf{TIBKEMenc}(pk, id)$
$\delta \xleftarrow{\$} \{0,1\}\ ;\ K^* \leftarrow K_\delta^*$
$\delta' \xleftarrow{\$} \mathcal{A}^{\textsc{KeyShare}(\cdot,\cdot),\textsc{DecShare}(\cdot,\cdot,\cdot)}(\mathtt{guess}, K^*, C^*, state)$
If $\delta \neq \delta'$ then return 0 else return 1

The set $I_c \subset \{1, \ldots, m\}$ is called the *set of corrupted players* and its cardinality, $|I_c|$ must be upper bounded by $l - 1$. The oracle $\textsc{KeyShare}(i, id)$ returns $sk_i \xleftarrow{\$} \mathsf{TIBKEMkey.Share}(pk, i, id, sk_i)$ with the restriction that $\mathcal{A}$ is not allowed to query for $id \neq id^*$ for non-corrupted players $i \notin I_c$. The oracle $\textsc{DecShare}(i, id, C)$ returns $C_i \xleftarrow{\$} \mathsf{TIBKEMdec.Share}(pk, vk, i, id, sk_i, C)$ (where the user secret

key $sk[id]_i$ was generated using $sk[id]_i \xleftarrow{\$} \mathsf{TIBKEMkey.Share}(pk, i, id, sk_i)$) with the restriction that $\mathcal{A}$ is not allowed to query for (i, id^*, C^*) for non-corrupted players $i \notin I_c$. We define the advantage of $\mathcal{A}$ in the experiment as

$$\mathbf{Adv}^{\textit{tibkem}\text{-cca}}_{\mathcal{TIBKEM},\mathcal{A}}(k) \;=\; \left|\Pr\left[\,\mathbf{Exp}^{\textit{tibkem}\text{-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k) = 1\,\right] - \frac{1}{2}\right| .$$

Definition 5.1 A threshold IB-KEM $\mathcal{TIBKEM}$ is said to be *secure against chosen-ciphertext attacks* if for any l, m with $0 < l \leq m$, the advantage function $\mathbf{Adv}^{\textit{tibkem}\text{-cca}}_{\mathcal{TIBKEM},\mathcal{A}}(k)$ is a negligible function in k for all polynomial-time adversaries $\mathcal{A}$.

Consistency requirements. Any threshold IB-KEM $\mathcal{TIBKEM}$ should satisfy two consistency requirements. On the one hand, *user secret key consistency* requires that for any reconstructed user secret key $sk[id]$ (obtained from a set of l valid user secret key shares) the same session key is obtained when decapsulating (via $\mathsf{TIBKEMdec}$) a valid ciphertext under the corresponding id. Secondly, *decapsulation consistency* requires that reconstructing the session key via $\mathsf{TIBKEMdec.Combine}$ for the same ciphertext C and identity id but for several different sets of l valid decapsulation shares results in the same session key.

Following [43, 10] we will formalize both consistency requirements using an adversary "attacking" the consistency of the schemes. Here we refer to reader to [1] for a general discussion on defining adveraries attacking consistency of a cryptographic scheme. As usual, "adversary" refers to a PTA but we stress that the consistency of our particular scheme can be proven with respect to unbounded adversaries.

For secret key consistency, we associate to an adversary $\mathcal{A}$ the experiment

Experiment $\mathbf{Exp}^{\textit{tibkem}\text{-key-consist}}_{\mathcal{TIBKEM},\mathcal{A}}(k)$

$(I_c, state_0) \xleftarrow{\$} \mathcal{A}(1^k, \mathtt{init})$ //adversary outputs the set of corrupted users
$(pk, sk, vk) \xleftarrow{\$} \mathsf{TIBKEMkg}(1^k, l, m)$
$(id, D, D', C) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyShare}(\cdot,\cdot),\textsc{DecShare}(\cdot,\cdot,\cdot)}(\mathtt{find}, pk, vk, \{sk_i\}_{i\in I_c}, state_0)$
$sk[id] \leftarrow \mathsf{TIBKEMkey.Combine}(pk, vk, id, D)$; $K \leftarrow \mathsf{TIBKEMdec}(pk, sk[id], C)$
$sk[id]' \leftarrow \mathsf{TIBKEMkey.Combine}(pk, vk, id, D')$; $K' \leftarrow \mathsf{TIBKEMdec}(pk, sk[id]', C)$
If $\mathtt{fail} \neq sk[id] \neq sk[id]' \neq \mathtt{fail}$ and $K \neq K'$ then output 1 else output 0

The set I_c and the oracles $\textsc{KeyShare}(i, id)$ and $\textsc{DecShare}(i, id, C)$ are as defined in the experiment $\mathbf{Exp}^{\textit{tibkem}\text{-cca}}_{\mathcal{TIBKEM},\mathcal{A}}$. The sets $D = \{D_1, \ldots, D_l\}$ and $D' = \{D'_1, \ldots, D'_l\}$ are two sets of valid key shares with respect to identity id. We define the advantage of $\mathcal{A}$ in the experiment as

$$\mathbf{Adv}^{\textit{tibkem}\text{-key-consist}}_{\mathcal{TIBKEM},\mathcal{A}}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\textit{tibkem}\text{-key-consist}}_{\mathcal{TIBKEM},\mathcal{A}}(k) = 1\,\right] .$$

For decapsulation consistency we associate to an adversary $\mathcal{A}$ the following experiment:

Experiment $\mathbf{Exp}^{\textit{tibkem}\text{-dec-consist}}_{\mathcal{TIBKEM},\mathcal{A}}(k)$

$(I_c, state_0) \xleftarrow{\$} \mathcal{A}(1^k, \mathtt{init})$ //adversary outputs the set of corrupted users
$(pk, sk, vk) \xleftarrow{\$} \mathsf{TIBKEMkg}(1^k, l, m)$
$(id, S, S', C) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyShare}(\cdot,\cdot),\textsc{DecShare}(\cdot,\cdot,\cdot)}(\mathtt{find}, pk, vk, \{sk_i\}_{i\in I_c}, state_0)$
$K \leftarrow \mathsf{TIBKEMdec.Combine}(pk, vk, id, S, C)$
$K' \leftarrow \mathsf{TIBKEMdec.Combine}(pk, vk, id, S', C)$
If $\mathtt{fail} \neq K \neq K' \neq \mathtt{fail}$ then return 1 else return 0

The set I_c and the oracles $\textsc{KeyShare}(i, id)$ and $\textsc{DecShare}(i, id, C)$ are as defined in the experiment $\mathbf{Exp}^{\textit{tibkem}\text{-cca}}_{\mathcal{TIBKEM},\mathcal{A}}$. The sets $S = \{C_1, \ldots, C_l\}$ and $S' = \{C'_1, \ldots, C'_l\}$ are two sets of valid decapsulation

shares with respect to (id, C). We define the advantage of $\mathcal{A}$ in the experiment as

$$\mathbf{Adv}^{\textit{tibkem}\text{-dec-consist}}_{\mathcal{TIBKEM},\mathcal{A}}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\textit{tibkem}\text{-dec-consist}}_{\mathcal{TIBKEM},\mathcal{A}}(k) = 1\,\right].$$

The experiment $\mathbf{Exp}^{\textit{tibkem}\text{-key-consist}}_{\mathcal{TIBKEM},\mathcal{A}}$ has already been considered in [10], while the experiment $\mathbf{Exp}^{\textit{tibkem}\text{-dec-consist}}_{\mathcal{TIBKEM},\mathcal{A}}$ is considered here for the first time. In particular, previous papers [3, 17] did not consider decapsulation consistency.

Definition 5.2 A threshold IB-KEM $\mathcal{TIBKEM}$ is said to be *consistent* if for any l, m with $0 < l \leq m$, and for any PTA adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ the two functions $\mathbf{Adv}^{\textit{tibkem}\text{-key-consist}}_{\mathcal{TIBKEM},\mathcal{A}_1}(k)$ and $\mathbf{Adv}^{\textit{tibkem}\text{-dec-consist}}_{\mathcal{TIBKEM},\mathcal{A}_2}(k)$ are negligible.

5.3 Discussion and Difficulties

It is already known how to make the key derivation threshold [10]. The crucial trick is to use bilinear pairings to explicitly check if a shared secret key $sk[id]_i$ was correctly generated. If not it can be rejected *before* the secret is reconstructed.

The difficulty for a full fledged threshold IB-KEM lies in the decapsulation shares. A similar method as above for generating decapsulation shares does not work since the plaintext M in an element from the second group $\mathbb{G}_T$ and we are not given a bilinear pairing on the group $\mathbb{G}_T$ (which does not exist since DDH in $\mathbb{G}_T$ and hence BDDH would be easy otherwise). In existing solutions [3] (based on the Boneh-Franklin IBE [12]) generic proofs of knowledge (POK) are used instead to prove consistency and random oracles are essential to make the proofs non-interactive.

We propose a different technique that completely avoids generic POK. The key idea is to make the decapsulation shares elements from $\mathbb{G}_1$. That makes possible to employ our techniques from the last section to prove consistency of the decapsulation shares. Our technique is reminiscent to the one proposed in [22] based on the 2-level hierarchical IBE from Gentry and Silverberg [26]. However, chosen-ciphertext security was not considered in [22]. In contrast to [22] our scheme does not add any further information to the ciphertext, i.e. we basically get "threshold for free" from chosen-ciphertext properties.

5.4 The Scheme

For the user secret key reconstruction set $I_r \subseteq \{1, \ldots, m\}$ we define the *Lagrange Coefficients* λ_i $(i \in I_r)$ as $\lambda_i = \prod_{j \in I_r \setminus \{i\}} \frac{j}{j-i} \in \mathbb{Z}_p^*$. For any polynomial $F \in \mathbb{Z}_p[X]$ of degree at most $|I_r| - 1$ this entails $\sum_{i \in I_r} F(i)\lambda_i = F(0)$. The coefficients $\lambda'_i = \prod_{j \in I'_r \setminus \{i\}} \frac{j}{j-i} \in \mathbb{Z}_p^*$ are defined analogously for the session key reconstruction set $I'_r \subseteq \{1, \ldots, m\}$. We call a (user secret key/ciphertext) share valid if it passes the respective consistency check. Let $\mathsf{TCR} : \mathbb{G}_1 \to \mathbb{Z}_p$ be a *target collision restant hash function*. Our threshold IB-KEM for identity space $\{0,1\}^n$ and threshold parameters m and l (l-out-of-m threshold scheme – at least l honest players are needed to perform threshold operations) is described by the following algorithms:

Key generation $\mathsf{TIBKEMkg}(1^k, l, m)$**.**

Choose $u_1, u_2 \xleftarrow{\$} \mathbb{G}_1^*$ and $b \xleftarrow{\$} \mathbb{Z}_p$, and compute $\alpha = u_1^b$ and $z \leftarrow \hat{e}(g, \alpha)$. Choose a random hash function $\mathsf{H} \xleftarrow{\$} \mathsf{HGen}(\mathbb{G}_1)$ and a parameters the the target collision resistant hash function TCR. The public key is defined as $pk = (u_1, u_2, z, \mathsf{TCR}, \mathsf{H})$.

Generate shared keys using l-out-of-m secret sharing by choosing $F_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1, \ldots, l-1$ and defining $F(X) = b + \sum_{i=1}^{l-1} F_i \cdot X^i$. The verification key is defined as $vk = (vk_1, \ldots, vk_m)$, where $vk_i = g^{F(i)}$. The shared secret key is defined as $sk = (sk_1, \ldots, sk_m)$, where $sk_i = u_1^{F(i)}$.

Shared user secret key delegation $\mathsf{TIBKEMkey.Share}(pk, i, id, sk_i)$

Choose $s_i \xleftarrow{\$} \mathbb{Z}_p$ and compute $d_{i,1} \leftarrow sk_i \cdot \mathsf{H}(id)^{s_i}$ and $d_{i,2} \leftarrow g^{s_i}$. The shared user secret key for player i is defined as $sk[id]_i = (d_{i,1}, d_{i,2})$.

Shared user secret key verification $\mathsf{TIBKEMkey.Vfy}(pk, vk, i, id, sk[id]_i)$

Parse $sk[id]_i = (d_{i,1}, d_{i,2})$ and check if $\hat{e}(d_{i,1}, g) = \hat{e}(vk_i, u_1) \cdot \hat{e}(d_{i,2}, \mathsf{H}(id))$

Shared user secret key combine $\mathsf{TIBKEMkey.Combine}(pk, vk, id, (sk[id]_i)_{i \in I_r})$

If $|I_r| < l$ or if one of the shares user secret keys $sk[id]_i$ $(i \in I_r)$ is not valid return `fail`. Otherwise parse $sk[id]_i = (d_{i,1}, d_{i,2})$ and return $sk[id] = (d_1, d_2) = (\prod_{i \in I_r} d_{i,1}^{\lambda_i}, \prod_{i \in I_r} d_{i,2}^{\lambda_i})$.

Encapsulation $\mathsf{TIBKEMenc}(pk, id)$

Choose $r \xleftarrow{\$} \mathbb{Z}_p^*$ and compute the encapsulation $C = (c_1, c_2, c_3) \in \mathbb{G}_1^3$ as

$$(c_1 = g^r, \quad c_2 = \mathsf{H}(id)^r, \quad c_3 = (u_1^t u_2)^r),$$

where $t = \mathsf{TCR}(c_1)$. The corresponding session key is $K = z^r \in \mathbb{G}_T$.

Decapsulation $\mathsf{TIBKEMdec}(pk, sk[id], C)$

Parse C as (c_1, c_2, c_3) and $sk[id]$ as (d_1, d_2). Compute $t = \mathsf{TCR}(c_1)$. We call a encapsulation C consistent iff $(g, c_1, u_1^t u_2, c_3)$ and $(g, c_1, \mathsf{H}(id), c_2)$ are DH tuples[3]. (Checking for a DH tuple can be done by computing the ratio of two pairings, i.e. $(g, c_1, u_1^t u_2, c_3)$ is a DH tuple if $\hat{e}(g, c_1) = \hat{e}(u_1^t u_2, c_3)$.) If C is not consistent then return `fail`. Otherwise reconstruct the session key as

$$K = \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \ .$$

Shared decapsulation $\mathsf{TIBKEMdec.Share}(pk, vk, i, id, sk[id]_i, C)$

Parse C as (c_1, c_2, c_3) and compute $t = \mathsf{TCR}(c_1)$. If C is not consistent then return `fail`. Otherwise choose $r_i \xleftarrow{\$} \mathbb{Z}_p$ and return the shared decapsulation for player i, $C_i = (C_{i,1}, C_{i,2}, C_{i,3})$ as

$$(C_{i,1} = g^{r_i}, \quad C_{i,2} = d_{i,1} \cdot (u_1^t u_2)^{r_i}, \quad C_{i,3} = d_{i,2}) \ .$$

Decapsulation share verification $\mathsf{TIBKEMdec.Vfy}(pk, vk, i, id, C_i, C)$

If C is not consistent or if

$$\hat{e}(g, C_{i,2}) \neq \hat{e}(vk_i, u_1) \cdot \hat{e}(C_{i,3}, \mathsf{H}(id)) \cdot \hat{e}(C_{i,1}, u_1^t u_2)$$

then return `fail`.

Combine decapsulation shares $\mathsf{TIBKEMdec.Combine}(pk, vk, id, (C_i)_{i \in I'_r}, C)$

Parse C as (c_1, c_2, c_3) and compute $t = \mathsf{TCR}(c_1)$. If $|I'_r| < l$ or if one of the shares C_i is not valid then return `fail`. Otherwise compute the values $B_1 = \prod_{i \in I'_r} C_{i,1}^{\lambda'_i}$, $B_2 = \prod_{i \in I'_r} C_{i,2}^{\lambda'_i}$, and $B_3 = \prod_{i \in I'_r} C_{i,3}^{\lambda'_i}$. Reconstruct the session key as

$$K = \frac{\hat{e}(c_1, B_2)}{\hat{e}(c_2, B_3) \cdot \hat{e}(u_1^t u_2, B_1)}.$$

[3] A tuple $(g, g^a, g^b, g^c) \in \mathbb{G}_1^4$ is said to be a *Diffie-Hellman tuple* (DH tuple) if $ab = c \bmod p$.

CORRECTNESS AND SECURITY. It is easy to verify that all verification checks are passed for correctly generated keys/encapsulations.

We now show correctness of the reconstructed private key. Let I_r a set of cardinality at least l. Assume the shares $sk[id]_i$ are all correct, that is $d_{i,1} = sk_i \cdot \mathsf{H}(id)^{s_i}$ and $d_{i,2} = g^{s_i}$. Let us define $s = \sum_{i \in I_r} s_i \lambda_i$. Then $d'_2 = \prod_{i \in I_r} d_{i,2}^{\lambda_i} = g^s$ and

$$d'_1 = \prod_{i \in I_r} d_{i,1}^{\lambda_i} = \prod_{i \in I_r} sk[id]_i \cdot \mathsf{H}(id)^{s_i \lambda_i} = \mathsf{H}(id)^s \prod_{i \in I_r} u_1^{F(i)} = u_1^b \mathsf{H}(id)^s,$$

as in key derivation.

We now show correctness of the reconstructed session key. Let I'_r be a set of cardinality at least l. Assume the shares C_i are all correct, i.e. $C_{i,1} = g^{r'_i}$, $C_{i,2} = d_{i,1} \cdot (u_1^t u_2)^{r'_i} = sk_i \cdot \mathsf{H}(id)^{s_i} \cdot (u_1^t u_2)^{r'_i}$, and $C_{i,3} = d_{i,2} = g^{s_i}$. We define $r' = \sum_{i \in I'_r} r'_i \lambda'_i$ and $s = \sum_{i \in I'_r} s_i \lambda'_i$. Then $B_1 = \prod_{i \in I'_r} C_{i,1}^{\lambda'_i} = g^{r'}$ and $B_3 = \prod_{i \in I'_r} C_{i,3}^{\lambda'_i} = g^s$. Furthermore,

$$\begin{aligned} B_2 &= \prod_{i \in I'_r} C_{i,2}^{\lambda'_i} = \prod_{i \in I'_r} g^{F(i) \cdot \lambda'_i} \cdot \mathsf{H}(id)^{s_i \cdot \lambda'_i} \cdot (u_1^t u_2)^{r'_i \cdot \lambda'_i} \\ &= g^{\sum_{i \in I'_r} F(i) \lambda'_i} \cdot \mathsf{H}(id)^s \cdot (u_1^t u_2)^{r'} \\ &= \alpha \cdot \mathsf{H}(id)^s \cdot (u_1^t u_2)^{r'} \end{aligned}$$

The key is computed as

$$\begin{aligned} K &= \hat{e}(c_1, C'_2)/(\hat{e}(c_2, C'_3) \cdot \hat{e}(c_3, C'_1)) \\ &= \hat{e}(g^r, \alpha \cdot \mathsf{H}(id)^s \cdot (u_1^t u_2)^{r'})/(\hat{e}(\mathsf{H}(id)^r, g^s) \cdot \hat{e}(u_1^{rt} u_2^r, g^{r'})) \\ &= \hat{e}(g^r, \alpha) \cdot \hat{e}(g^r, \mathsf{H}(id)^s)/\hat{e}(\mathsf{H}(id)^r, g^s) \cdot \hat{e}(g^r, (u_1^t u_2)^{r'})/\hat{e}(u_1^{rt} u_2^r, g^{r'})) \\ &= z^r \ , \end{aligned}$$

as in encapsulation.

Theorem 5.3 Assume TCR is a target collision resistant hash function. Under the Bilinear Decisional Diffie-Hellman (BDDH) assumption relative to generator $\mathcal{G}$, our threshold IB-KEM is secure against chosen-ciphertext attacks. In particular, given an adversary $\mathcal{A}$ attacking the chosen-ciphertext security of the threshold IB-KEM with advantage $\varepsilon_{\mathcal{A}} = \mathbf{Adv}^{tibkem\text{-cca}}_{\mathcal{TIBKEM},\mathcal{A}}$ and running time $\mathbf{Time}_{\mathcal{A}}(k)$ we construct an adversary $\mathcal{B}$ breaking the BDDH assumption with advantage $\varepsilon_{\mathcal{B}} = \mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k)$ and running time $\mathbf{Time}_{\mathcal{B}}(k)$ with

$$\begin{aligned} \varepsilon_{\mathcal{B}}(k) &\geq \frac{\varepsilon_{\mathcal{A}}(k) - \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)}{8(n+1)q} - q/p; \\ \mathbf{Time}_{\mathcal{B}}(k) &\leq \mathbf{Time}_{\mathcal{A}} + \tilde{\mathcal{O}}(nq \cdot \varepsilon_{\mathcal{A}}^{-2}(k)), \end{aligned}$$

where q is an upper bound on the number of key derivation/decapsulation share queries made by adversary $\mathcal{A}$.

Theorem 5.4 Our threshold IB-KEM is consistent. In particular, we have

$$\mathbf{Adv}^{tibkem\text{-key-consist}}_{\mathcal{TIBKEM},\mathcal{A}_1}(k) = \mathbf{Adv}^{tibkem\text{-dec-consist}}_{\mathcal{TIBKEM},\mathcal{A}_2}(k) = 0 \ .$$

The above statement even holds for unbounded adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e. we have perfect consistency.

The proofs of the two theorems will be given in Appendix B.

Relation to the threshold KEM from BMW. Clearly, our threshold IB-KEM implies (standard) threshold KEM by simply ignoring all operations related to the identity. We remark that viewed in this light, our threshold IB-KEM can be simplified to the chosen-ciphertext secure threshold KEM scheme recently proposed by Boyen, Mei, and Waters [14]. Although the computational cost remains the same, the public key of our scheme still saves two elements in $\mathbb{G}_1$ compared to BMW's public key.

From threshold IB-KEM to full threshold IBE. Designing a full threshold IBE from a threshold IB-KEM is not an easy task. Let us have a glimpse on that.

In the IB-KEM framework [7], a standard (hybrid) IBE scheme can be obtained by using the IB-KEM to securely transport a random session key that is fed into a symmetric encryption scheme to encrypt the plaintext message. If both the IB-KEM and the symmetric encryption scheme are chosen-ciphertext secure, then the resulting hybrid IBE is also chosen-ciphertext secure, with a tight reduction. A symmetric encryption scheme secure against chosen-ciphertext attacks can be built from relatively weak primitives, i.e. from any (one-time) symmetric encryption scheme by essentially adding a MAC. Alas, sharing a MAC is not trivial in general, and will often lead to costly computations.

This problem can be solved by using the related notion of Tag-IB-KEM, which is obtained by translating the notion of Tag-KEM [2] to the identity-based encryption setting. A Tag-IB-KEM is an IB-KEM which takes an additional input called *tag.* Such a tag is a binary string of appropriate length, and need not have any particular structure. By extending the results in [2] to the IBE framework, is it possible to show that combining a CCA secure Tag-IB-KEM with chosen-plaintext secure symmetric encryption yields a CCA secure IBE scheme. This immediately implies a construction for threshold IBE from any threshold Tag-IB-KEM, since now we do not need to share the symmetric encryption component anymore. Finally, a CCA secure threshold Tag-IB-KEM can be built from combining a threshold IB-KEM, a key derivation function and a MAC, without considerable overheads.

6 Extensions

6.1 Chosen-ciphertext secure Hierarchical Identity-Based Key Encapsulation

Hierarchical identity-based key encapsulation (HIB-KEM) is a generalization of IB-KEM to identities supporting hierarchical structures [28, 26]. By the relation to Waters IBE scheme it is easy to see that our technique can also be used to make (the KEM variant of) Waters' HIBE chosen-ciphertext secure. To be more precise, we modify Waters' HIB-KEM and add one more element $h_1^{rt}h_2^r$ to the the ciphertext, where t was computed by applying a target-collision hash function to g^r (here r is the randomness used to create the ciphertext). The additional element is used for a consistency check at decryption. The security reduction is exponential in the depth d of the hierarchy, i.e. it introduces, roughly, a multiplicative factor of $(nq)^d$.

6.2 Identity-based Encryption

Given a IB-KEM and a symmetric encryption scheme, a hybrid identity-based encryption scheme can be obtained by using the IB-KEM to securely transport a random session key that is fed into the symmetric encryption scheme to encrypt the plaintext message. It is well known [20, 7] that if both the IB-KEM and the symmetric encryption scheme are chosen-ciphertext secure, then the resulting hybrid encryption is also chosen-ciphertext secure. The security reduction is tight.

A symmetric encryption scheme secure against chosen-ciphertext attacks can be built from relatively weak primitives, i.e. from any (one-time) symmetric encryption scheme by essentially adding a

MAC. For concreteness we mention that a chosen-ciphertext secure IBE scheme can be built from our IB-KEM construction with an additional overhead of about 128 bits (for the MAC).

We note that for the natural task of securely generating a joint random session key, a IB-KEM is sufficient and a fully-fledged identity-based encryption scheme is not needed.

The same remark holds for hierarchical identity-based encryption (HIBE) and threshold identity-based encryption.

6.3 A Tradeoff between public key size and security reduction

As independently discovered in [18, 33], there exists an interesting trade-off between key-size of Waters' hash H and the security reduction of the IBE scheme.

The construction modifies Waters hash H as follows: Let the integer $l = l(k)$ be a new parameter of the scheme. In particular, we represent an identity $id \in \{0,1\}^n$ as an n/l-dimensional vector $id = (id_1, \ldots, id_{n/l})$, where each id_i is an l bit string. Waters hash is then redefined to $\mathsf{H} : \{0,1\}^n \to \mathbb{G}_1$, with $\mathsf{H}(id) = h_0 \prod_{i=1}^{n/l} h_i^{id_i}$ for random public elements $h_0, h_1, \ldots, h_{n/l} \in \mathbb{G}_1$. Waters' original hash function is obtained as the special case $l = 1$. It is easy to see that using this modification in our IBE scheme (i) reduces the size of the public key from $n+4$ to $n/l+4$ group elements, whereas (ii) it adds another multiplicative factor of 2^l to the security reduction of the IBE scheme (Theorem 4.1).[4]

6.4 Selective-identity chosen-ciphertext secure IB-KEM

For the definition of a selective-identity chosen-ciphertext secure IB-KEM we change the security experiment such that the adversary has to commit to the target idenity id^* before seeing the public key. Clearly, this is a weaker security requirement. We quickly note that (using an algebraic technique from [8]) by replacing Waters' hash H with $\mathsf{H}(id) = h_0 \cdot h_1^{id}$ (for $id \in \mathbb{Z}_p$) we get a selective-id chosen-ciphertext secure IB-KEM. Note that the size of the public-key of this scheme drops to 3 elements.

6.5 Implementing the collision resistant hash function TCR

In practice, to build a target collision resistant hash function, one can use a dedicated cryptographic hash function, like SHA-1 [38].

Every injective function $\mathsf{TCR} : \mathbb{G}_1 \to \mathbb{Z}_p$ trivially also is (target) collision resistant (with zero advantage). Boyen, Mei and Waters [14] note that for bilinear maps defined on elliptic curves there exists a very efficient way to implement such injective mappings. We refer to [14] for more details.

7 Efficiency comparison of our IB-KEM

In this section we compare our IB-KEM from Section 4 with the previous chosen-ciphertext secure IB-KEM/IBE schemes in the literature.

Basically, there are two proposals of IBE scheme in the literature, one by combining the IBE schemes [8, 45] with the generic transformation from [16], the other one stems from a remark from [14]. We will now carefully review both constructions and compare them to our proposed scheme.

7.1 IB-KEM scheme obtained by the generic CHK transformation

We begin by reviewing the generic transformation from any (weakly secure) 2-HIBE into a chosen-ciphertext secure IBE scheme by Canetti, Halevi, and Katz [16] (CHK), which was later improved by

[4] On the technical side our proof basically stays the same, only the bound from Lemma A.2 needs to be adapted to take the modified Waters' hash into account.

Boneh-Katz [13] (BK). We describe the CHK transformation in terms of key encapsulation and note that this is not possible for the improved BK transformation.

The CHK method transforms any two level HIB-KEM into an IB-KEM scheme as follows: the identity of the IB-KEM scheme becomes the identity of the first level HIB-KEM. To create a ciphertext of the IB-KEM a random pair of signing/verification keys is chosen. A HIB-KEM ciphertext for the message is created with respect to the two-level identity consisting of the HIB-KEM identity at the first level and the verification key at the second level. The resulting HIB-KEM ciphertext is signed using the signing key. Finally, the HIB-KEM ciphertext is then composed by the HIB-KEM ciphertext, the signature, and the corresponding verification key.

For decapsulation first the validity of the signature is checked and then, conditioned it was correct, the HIB-KEM ciphertext is decapsulated using the hierarchical key-derivation algorithm for the 2-level "identity" consisting of id plus the verification key.

It was proved in [16] that any chosen-plaintext secure 2-HIB-KEM with a weak security property with respect to the second level of the hierarchy (i.e. selective-identity security) and (full) security at the first level is sufficient to obtain a chosen-ciphertext secure IBE. Consequently, as noted in [45], the most efficient instantiation of this transformation is obtained from the hybrid HIB-KEM using Waters IB-KEM [45] at the first level and Boneh/Boyen's IB-KEM [8] at the second level.

Combining the results from [16] with [45, 8] we get a chosen-ciphertext secure IB-KEM under the BDDH assumption. Similar to our scheme the security reduction roughly comes with a multiplicative factor of $\approx nq$.

7.2 IB-KEM mentioned in BMW

In contrast to [16, 13], Boyen, Mei, and Waters [14] propose a non black-box technique to obtain a chosen-ciphertext secure IB-KEM from a 2-level HIB-KEM without relying on additional primitives like signatures or MACs. For concreteness we cite their concrete statement (from Sec. 5.3 of the full version of [14]), referring to the two HIBE constructions from Boneh-Boyen [8] and Waters [45]:

> "It is easy to see that we obtain the desired result [i.e. a construction avoiding a signature/MAC] very simply, by extending the hierarchy in either HIBE construction by one level, and setting the "identity" for that last level to be the hash value of the previous ciphertext components. This gives us (in the Waters case) an adaptive-identity CCA2-secure HIBE, and (in the Boneh-Boyen case) a selective-identity CCA2-secure HIBE."

No theorem statement (or further explanation beyond that) is given but it is clear that security relies on Waters 2-HIBE which has a loss-factor of roughly $(nq)^2$ in the reduction from BDDH. We want to stress that in their construction the "hashing the previous ciphertext" makes it basically impossible to replace the second level of Waters HIBE with the more efficient (but only weakly=selective-identity secure) IBE scheme from Boneh-Boyen. (This is since the challenge ciphertext depends on the target identity which is used in the second-level of the HIBE scheme and the target identity is not known until the adversary outputs it.)

Since the construction uses Waters 2-HIBE the public-key has to include two independent sets of hash public-keys, i.e. the public key contains roughly $2n$ elements from $\mathbb{G}_1$. For the same reason the security reduction of the proposed IBE scheme depends on the security of Waters' 2-HIBE which is quadratic in q and n.

7.3 A comparison

An efficiency comparison between the above two schemes (plus Waters original scheme) and our IBE is given in Figure 2 which will be further commented in the following prose. We stress that the

Scheme	CCA?	Ciphertext Overhead	Encapsulation #pairings + #[multi,regular,fixed-base]-exp+...	Decapsulation	Keysize pk	Security Reduction
Ours (§4)	√	3ℓ	$0 + [1, 3, 1]$	$3 + [1, 0, 2]$	$n + 4$	nq
Hybrid + CHK (§7.1)	√	3ℓ+25600 (704)	$0 + [1, 3, 1]$+Sig	$3 + [1, 0, 1]$+Vfy	$n + 4$	nq
BMW (§7.2)	√	$3\ell'$	$0 + [0, 5, 1]$	$4 + [0, 1, 0]$	$2n + 3$	$(nq)^2$
Waters	—	2ℓ	$0 + [0, 3, 1]$	$2 + [0, 0, 0]$	$n + 2$	nq

Figure 2: Efficiency comparison for CCA-secure IB-KEMs for identity-space $\mathsf{IDSp} = \{0,1\}^n$. BMW is the IB-KEM as proposed in [14], Hybrid + BK is the Waters/BB hybrid HIB-KEM scheme applied to the CHK transformation as proposed in [45], and Waters is Waters' original chosen-plaintext secure IBE scheme [45] (which is added for comparison). The keysize is measured in terms of the number of group elements of the public key pk. Ciphertext overhead represents the difference (in bits) between the ciphertext length and the message length. ℓ is the length of the representation of an element in $\mathbb{G}_1$ with respect to the security reduction $O(nq)$, while ℓ' is the length of a $\mathbb{G}_1$ group element with respect to the security reduction $O(n^2q^2)$ (thus $\ell \ll \ell'$). For comparison we mention that relative timings for the various operations are as follows: bilinear pairing ≈ 5 [37], multi-exponentiation ≈ 1.5, regular exponentiation $= 1$, fixed-base exponentiation $\ll 0.2$.

performance of the security reduction is a crucial parameter here. In light of the keysize/security reduction tradeoff from Section 6.3 we can also compare the BMW scheme to all other schemes by "normalizing" the security reduction for all schemes to $O(n^2q^2)$, i.e. by setting the tradeoff parameter l to $l = \log_2(nq) \approx 20 + 7 = 27$ (for very optimistic $< 2^{20}$ adversary queries and identities of $n = 160$ bits[5]) we get a public-key size of $n/27 + 4 \approx 9$ group elements compared to the $2n + 3 = 323$ group elements of BMW with the *same security*.

The symmetric overhead of the CHK transformation consists of a strong one-time signature plus a verification key which sums up to $\approx 160^2 = 25600$ ("security parameter squared") bits [23].

Since computing Waters hash requires computing $n/2$ products in $\mathbb{G}_1$ on the average, where $n \approx \log_2 p$, it can be seen as a single exponentiation. Therefore we count computing $\mathsf{H}(id)^r$ for random r as two (regular) exponentiations. In the decapsulation algorithm of our IB-KEM we assume $\mathsf{H}(id)$ to be precomputed. The size of the secret key sk is the same for all three schemes (a single element in $\mathbb{G}_1$).

To summarize, compared to BMW (from Section 7.2) our proposed chosen-ciphertext secure IBE scheme achieves better performance and public key sizes with half of the BMW public key size. In addition, the fact that our security reduction is more efficient than that of the BMW scheme means that for concrete values of the security parameter our ciphertexts are much shorter even if the two schemes have the same number of elements in the ciphertext.

In terms of computational efficiency our scheme has one fixed-based exponentiation more than the the Hybrid + CHK scheme from Section 7.1 but it does not have to resort on any kind of exogenous consistency check such as a signature or a MAC. Since one fixed-based exponentiation is $\ll 0.2$ regular exponentiations (which can fairly be neglected) we conclude that encapsulation/decapsulation in our scheme is as efficient as in the Hybrid + CHK scheme. The most striking difference, however, is that our scheme comes with shorter ciphertexts: for current security requirements the ciphertexts difference (a strong one-time signature plus a verification key) amounts to a couple of thousand bits [23].

We remark that, in order to get a direct full IBE scheme (in contrast to an IB-KEM) we can also apply the BK-transformation [13] to the construction from Section 7.1 and get a full IBE scheme with shorter ciphertexts. The latter construction significantly reduces the ciphertext overhead compared

[5] 160 bits seems to be a reasonable value since for larger identity space you can always first apply a collision resistant hash function $\mathsf{CR} : \{0,1\}^* \to \{0,1\}^{160}$.

to the CHK-transformation (by replacing the signature with a MAC). Compared to our construction, however, there is still a difference in the ciphertext size of a MAC tag plus a "commitment" which sums up to ≈ 576 bits [13].

We conclude that our scheme seems to outperform both existing chosen-ciphertext secure IB-KEM proposals.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In Victor Shoup, editor, *CRYPTO 2005*, LNCS, Santa Barbara, CA, USA, August 14–18, 2005. Springer-Verlag, Berlin, Germany.

[2] Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of kurosawa-desmedt KEM. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 128–146, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

[3] Joonsang Baek and Yuliang Zheng. Identity-based threshold decryption. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 262–276, Singapore, March 1–4, 2004. Springer-Verlag, Berlin, Germany.

[4] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45, Santa Barbara, CA, USA, August 23–27, 1998. Springer-Verlag, Berlin, Germany.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[6] Mihir Bellare and Phillip Rogaway. The game-playing technique. Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/`.

[7] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005. `http://eprint.iacr.org/`.

[8] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[9] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

[10] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *Topics in Cryptology—CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243. Berlin: Springer-Verlag, 2006.

[11] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.

[12] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[13] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103, San Francisco, CA, USA, February 14–18, 2005. Springer-Verlag, Berlin, Germany.

[14] Xavier Boyen, Qixiang Mei, and Brent Waters. Simple and efficient CCA2 security from IBE techniques. In *ACM Conference on Computer and Communications Security—CCS 2005*, pages 320–329. New-York: ACM Press, 2005. Available at `http://eprint.iacr.org/2005/288/`, August 2005.

[15] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *30th ACM STOC*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press.

[16] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[17] Zhenchuan Chai, Zhenfu Cao, and Rongxing Lu. ID-based threshold decryption without random oracles and its application in key escrow. Proceedings of ICISC, 2004.

[18] Sanjit Chatterjee and Palash Sarkar. Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. Proceedings of ICISC, 2004.

[19] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer-Verlag, Berlin, Germany.

[20] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[21] CRYPTREC (cryptography research & evaluation committees): The cryptographic technique evaluation project, August 2003. http://www.ipa.go.jp/security/enc/CRYPTREC/index.html.

[22] Y. Dodis and M Yung. Exposure-resilience for free: The hierarchical id-based encryption case. In *Proceedings of IEEE Security in Storage Workshop 2002*, pages 45–52, 2002.

[23] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.

[24] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer-Verlag, Berlin, Germany.

[25] David Galindo and Eike Kiltz. Threshold chosen-ciphertext secure identity-based key encapsulation without random oracles. In *SCN 2006*, volume 4116 of *LNCS*, pages 173–185. Springer-Verlag, 2006.

[26] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer-Verlag, Berlin, Germany.

[27] Hitachi Ldt. A symmetric key encyption algorithm: MULTI-S01, 2001. available at `http://www.sdl.hitachi.co.jp/crypto/s01/01espec.pdf`.

[28] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer-Verlag, Berlin, Germany.

[29] IEEE P1363.3 Committee. IEEE 1363.3 / CfS — standard for identity-based cryptographic techniques using pairings. `http://grouper.ieee.org/groups/1363/index.html/`, February 2006. Call for submissions.

[30] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *Algorithmic Number Theory – ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.

[31] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, LNCS, pages 581–600, New York, USA, February 5–7, 2006. Springer-Verlag, Berlin, Germany.

[32] Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In *ACISP 2006*, volume 4058 of *LNCS*, pages 336–347. Springer-Verlag, 2006.

[33] David Naccache. Secure and *practical* identity-based encryption. Cryptology ePrint Archive, Report 2005/369, 2005. `http://eprint.iacr.org/`.

[34] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43, Seattle, Washington, USA, May 15–17, 1989. ACM Press.

[35] NESSIE Final report of European project IST-1999-12324: New European Schemes for Signatures, Integrity, and Encryption, April 2004. Working draft.

[36] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.

[37] Michael Scott. Faster pairings using an elliptic curve with an efficient endomorphism. Cryptology ePrint Archive, Report 2005/252, 2005. `http://eprint.iacr.org/`.

[38] Secure hash standard. National Institute of Standards and Technology, NIST FIPS PUB 180-1, U.S. Department of Commerce, April 1995.

[39] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer-Verlag, Berlin, Germany.

[40] Victor Shoup. OAEP reconsidered. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 239–259, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.

[41] Victor Shoup. A proposal for an ISO standard for public key encryption (version 2.1). manuscript, 2001. Available on `http://shoup.net/papers/`.

[42] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. manuscript, 2004. Available from `http://shoup.net/papers/`.

[43] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 1–16, Espoo, Finland, May 31 – June 4, 1998. Springer-Verlag, Berlin, Germany.

[44] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, 2002.

[45] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

A Security of the IB-KEM

In this section we provide a game-based proof of Theorem 4.1. We will make use of the following simple "Difference Lemma" [40].

Lemma A.1 Let X_1,X_2, B be events defined in some probability distribution, and suppose that $X_1 \wedge \neg B \Leftrightarrow X_2 \wedge \neg B$. Then $|\Pr[X_1] - \Pr[X_2]| \leq \Pr[B]$.

Game 0. Fix an efficient adversary $\mathcal{A}$. We now define a game, Game 0, an interactive game between adversary $\mathcal{A}$ and a simulator. Game 0 is simply the same game as the IB-KEM security experiment of Section 2.2 in which the simulator provides adversary $\mathcal{A}$'s environment. While describing the experiment we wll make a couple of conventions on how the simulator chooses the values appearing in its simulation. These conventions will be purely conceptual and, compared to the original experiment, do not change the distribution of any value appearing during the experiment. We will also make a couple of definitions of values appearing during the experiments.

We assume that in the beginning the simulator chooses some values a, b, and c, uniformly distributed over $\mathbb{Z}_p$. The simulation will depend on these values (i.e., the key generation will depend on a, b, where the challenge ciphertext will depend on c). In sequel games the simulator will "forget" the values a, b, and c and instead only use the values g^a, g^b, and g^c for its simulation of the security experiment.

Key Generation. Initially the simulator runs the IB-KEM key generation algorithm $\mathsf{IBKEMkg}(1^k)$ and obtains the public key $mpk = (u_1, u_2, z, \mathsf{H})$ and secret key $msk = \alpha$. We make the convention that the public key is generated as

$$u_1 \leftarrow g^a, \quad u_2 \xleftarrow{\$} \mathbb{G}_1, \quad z \leftarrow \hat{e}(g^a, g^b), \quad \mathsf{H} \xleftarrow{\$} \mathsf{HGen}(\mathbb{G}_1)\ , \tag{1}$$

depending on the elements g^a, g^b. Note that the way the value $z = \hat{e}(g^a, g^b) = \hat{e}(g, g^{ab})$ from the public key is generated implies $\alpha = g^{ab} = u_1^b$. The secret key α can be computed since a and b are known to the simulator. The public key is given to the adversary $\mathcal{A}$ to start its `find` phase.

Find Phase. During its execution adversary $\mathcal{A}$ makes a number of key derivation and decapsulation requests. If the adversary makes a key derivation query $\mathsf{IBKEMKeyDer}(id)$ then (using its secret key α) the simulator computes the secret key $sk[id]$ and returns it to the adversary. If the adversary makes a decapsulation query $\textsc{Dec}(id, C)$ the simulator (using α) decrypts the ciphertext and returns the plaintext to the adversary.

Eventually, the adversary returns a target identity id^*. The simulator chosen a random key K_0^* and run the encapsulation algorithm to create a key K_1^* together with the the challenge ciphertext

$C^* = (c_1^*, c_2^*, c_3^*)$. We make the convention that the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$ is computed as

$$c_1^* \leftarrow g^c, \quad t^* \leftarrow \mathsf{TCR}(g^c), \quad c_2^* \leftarrow \mathsf{H}(id^*)^c, \quad c_3^* \leftarrow (u_1^{t^*} u_2)^c \tag{2}$$

depending on the random value c chosen by the simulator in advance, and the key $K_1^* = z^c$. Then the simulator chooses a random bit γ and the challenge ciphertext C^* together with the key $K^* = K_\gamma^*$ is returned to the adversary.

Guess Phase. The adversary continues to make its oracle queries, subsequent key derivation requests must be different from the target identity id^* and decapsulation requests must be different from (id^*, C^*). Finally, adversary $\mathcal{A}$ returns a bit $\gamma' \in \{0,1\}$. If $\gamma \neq \gamma'$ the simulator returns $\beta' = 0$, else it returns $\beta' = 1$. This completes the description of the simulator. Note that the simulator behaves exactly as in the original IB-KEM security experiment.

Now a few important definitions are in place. During its execution $\mathcal{A}$ may query the key derivation oracle for some identity id or the decapsulation oracle for the identity/ciphertext pair (id, C). We collect all those identities used to make queries to the key derivation and decapsulation oracle in the set $\widetilde{ID}$. Note that $\widetilde{ID}$ may contain the target identity id^* or one identity more than once. Let ID be the subset of queried identities obtained by removing from $\widetilde{ID}$ all multiples and the target identity. We write $ID = \{id^{(1)}, \ldots, id^{(q_0)}\}$ (without any particular order) for some $q_0 \leq q$ such that $id^{(i)} \neq id^{(j)}$ for each $1 \leq i \neq j \leq q_0$ and $id^* \notin ID$. Furthermore, we define $ID^* = ID \cup \{id^*\} = \{id^{(1)}, \ldots, id^{(q_0)}, id^*\}$.

The proof of the theorem is obtained by considering subsequent games, Game 1, Game 2, ..., Game 10. These games will be quite similar to Game 0. In every game the simulators' output bit β' will be well-defined. For $0 \leq i \leq 10$ we define the event

$$X_i : \text{The simulator outputs } \beta' = 1 \text{ in Game } i.$$

Then, since in Game 0 the simulator exactly plays the IB-KEM security experiment with adversary $\mathcal{A}$, we have

$$|\Pr[X_0] - 1/2| = \mathbf{Adv}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}} .$$

Game 1. (Eliminate hash collisions) Note that the values $c_1^* = g^c$ and $t^* = \mathsf{TCR}(g^c)$ from the challenge ciphertext Eqn. (2) are completely independent of the view of adversary $\mathcal{A}$ until $\mathcal{A}$'s `guess` phase (since c is simply not touched by the simulator before generating the challenge ciphertext). Therefore we may assume that the value c_1^* and t^* are already generated by the simulator before the key generation.

In this game the simulator changes its answers to all decapsulation queries $\textsc{Dec}(id, C)$ made by $\mathcal{A}$ as follows: Let $C = (c_1, c_2, c_3)$ and $t = \mathsf{TCR}(c_1)$. If $t = t^*$ and $c_1 \neq c_1^*$, the simulator aborts. Otherwise it continues as in the last game. Let HashAbort be the event that this new abortion rule applies. Until HashAbort happens Game 0 and Game 1 are identical. Therefore by Lemma A.1 we have

$$|\Pr[X_1] - \Pr[X_0]| \leq \Pr[\textsc{HashAbort}] .$$

Furthermore,

$$\Pr[\textsc{HashAbort}] \leq \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) ,$$

i.e. there exists an adversary $\mathcal{H}$ against the target collision resistence of TCR (note that $c_1^* = g^c$ is a random element) running in time $\mathbf{Time}_{\mathcal{H}}(k) \approx \mathbf{Time}_{\mathcal{A}}(k)$ that succeeds with probability at least $\Pr[\textsc{HashAbort}]$.

Game 2. (Change of the hash keys) This is the same as Game 1 except that the simulator changes the generation of the hash keys $h = (h_0, h_1, \ldots, h_n)$ as follows.

Set $m = 2q$ (the choice of m will become clear later). Instead of generating the hash keys with the hash key-generation algorithm $\mathsf{HGen}(\mathbb{G}_1)$ (cf. Section 4.1) as in the last game the simulator chooses

$$\begin{aligned} x_0, x_1, \ldots, x_n &\xleftarrow{\$} \{0, \ldots, p-1\} \\ y'_0, y_1, \ldots, y_n &\xleftarrow{\$} \{0, \ldots, m-1\} \\ k &\xleftarrow{\$} \{0, \ldots, n\} \end{aligned} \tag{3}$$

and sets

$$y_0 \leftarrow p - km + y'_0 \,.$$

The public keys $h = (h_0, \ldots, h_n)$ of the hash function H are then defined as $h_i = g^{x_i} u_1^{y_i}$, for $0 \leq i \leq n$. By definition the public hash function evaluated in identity $id \in \{0,1\}^n$ is given as $\mathsf{H}(id) = h_0 \prod_{i=1}^n h_i^{id_i}$. From the simulator's point of view, however, the hash function evaluated in $id \in \{0,1\}^n$ looks like

$$\mathsf{H}(id) = g^{x(id)} u_1^{y(id)}, \tag{4}$$

with $x(id) = x_0 + \sum_{i=1}^n id_i x_i$ and $y(id) = y_0 + \sum_{i=1}^n id_i y_i$ only known to the simulator. On the other hand note that this change does not affect the distribution of the hash keys $h = (h_0, h_1, \ldots, h_n)$. Therefore we have

$$\Pr[X_2] = \Pr[X_1] \,.$$

Game 3. (Abort at the end of the game) Fix all the random variables adversary $\mathcal{A}$ gets to see during its execution, including its random coin tosses: fix mpk, the challenge bit γ, and the randomness used in answering the key derivation and decapsulation queries. Now the adversary can be seen as a deterministic algorithm, in particular the set of all queried (distinct) identities $ID^* = \{id^{(1)}, \ldots, id^{(q_0)}, id^*\}$ can be seen as fixed. By $view_{\mathcal{A}}$ we denote all these fixed variables.

Define $\mathbf{Y} = (y'_0, y_1, \ldots, y_n, k)$, where the random variables $(y'_0, y_1, \ldots, y_n, k)$ are distributed as in Eqn. (3). It is clear that once $view_{\mathcal{A}}$ is fixed, the random variable $\mathbf{Y}$ still has its original distribution. Define the event

$$\textsc{ForcedAbort} : \bigvee_{i=1}^{q_0} \Big(y(id^{(i)}) = 0 \bmod p \Big) \vee y(id^*) \neq 0 \bmod p \,.$$

We call this abort *forced* since in sequel games the simulator is modified such that it always *has to* abort once this event happens. For fixed $view_{\mathcal{A}}$ we define

$$\eta := \Pr_{\mathbf{Y}}[\neg \textsc{ForcedAbort}] \tag{5}$$

and let λ be a lower bound on η (that holds for every $view_{\mathcal{A}}$). The following lemma provides a lower bound on η.

Lemma A.2 For each possible choice of identities $ID^* = \{id^{(1)}, \ldots, id^{(q_0)}, id^*\}$ we have $\eta \geq \lambda = \frac{1}{4(n+1)q}$.

The proof of the lemma is quite technical and is postponed to Section A.2.

Compared to Game 2 we will make two modifications to the simulator in Game 3. The simulation is exactly the same as in Game 2 until adversary $\mathcal{A}$ outputs his guess bit γ'. Since adversary $\mathcal{A}$ already terminated we can assume $view_{\mathcal{A}}$ to be fixed from now on.

First modification: add forced abort. After adversary $\mathcal{A}$ outputs his guess bit γ', the simulator checks if the event ForcedAbort occurs. If yes, it aborts the game and returns a random bit as its output bit β'. If not the simulation is continued as before.

Let's first make an unsuccessful attempt to relate the two events X_3 and X_2. Clearly we have $\Pr[X_3] = \Pr[X_2 \wedge \neg\textsc{ForcedAbort}] + 1/2 \cdot \Pr[\textsc{ForcedAbort}]$. Now we would like to continue with $\Pr[X_2 \wedge \neg\textsc{ForcedAbort}] \geq \Pr[X_2] \cdot \Pr[\neg\textsc{ForcedAbort}]$. However, this is not correct since the simulator may aborts with a probability that is a function in the choices of the identities $ID^* = \{id^{(1)}, \ldots, id^{(q_0)}, id^*\}$ queried by adversary $\mathcal{A}$ and hence the two events X_2 and $\neg\textsc{ForcedAbort}$ cannot be considered as independent.

To get rid of this unwanted dependence the simulator adds some *artificial abort* such that it always aborts with probability nearly λ (recall that λ was is upper bound on the abortion probability), independent of the choices of the identities $ID^* = \{id^{(1)}, \ldots, id^{(q_0)}, id^*\}$. This way it will be possible to decorrelate the event X_2 with the abortion.

Second modification: add artificial abort. After the simulator has checked for the event ForcedAbort (and decided not to abort), it continues as follows: First it samples (using sufficiently many samples) an estimate η' of the probability η (over $\mathbf{Y}$) that the ForcedAbort happens (cf. Eqn. (5)).[6] We want to stress that $view_{\mathcal{A}}$ is fixed at this point so sampling does not involve running adversary $\mathcal{A}$ again. This estimate η' is a function in $id^{(1)}, \ldots, id^{(q_0)}, id^*$.
Depending on the estimate η' the simulator distinguishes two cases:
Case $\eta' \leq \lambda$: the simulator continues as before.
Case $\eta' > \lambda$: With probability $1 - \lambda/\eta'$ the simulator aborts and outputs a random bit β'. With probability λ/η' the simulator does not abort and continues as before.
This concludes the description of Game 3.

The following lemma relating the events X_2 and X_3 will be proved in Section A.1.

Lemma A.3 Let $0 < \rho \leq 1$ be a function in k. If the simulator takes $\mathcal{O}(\rho^{-2}\ln(\rho^{-1}) \cdot \lambda^{-1}\ln(\lambda^{-1}))$ samples when computing the estimate η', then

$$\left|\Pr[X_2] - \frac{1}{2} - \frac{\Pr[X_3] - 1/2}{\lambda}\right| \leq \frac{\rho}{2}$$

The parameter ρ will be determined at the end of the proof.

Game 4. (Forced abort during the game I) Compared to the last game we make the following changes to the simulator: When identity $id \in ID$ is queried to the key derivation oracle, the simulator immediately aborts if $y(id) = 0 \bmod p$. When receiving the challenge identity id^*, the simulator immediately aborts if $y(id^*) \neq 0 \bmod p$. On abort the simulator returns a random bit β'. The artificial abort at the end of the simulation is the same as in the last game.

Clearly, this modification does not affect the adversary if there is no forced abort. In case there is a new forced abort the simulator outputs a random bit β' as in Game 3. Therefore we have

$$\Pr[X_4] = \Pr[X_3] .$$

Game 5. (Change key derivation oracle) The simulator changes its answers to all key derivation queries $\mathsf{IBKEMKeyDer}(id)$ made by the adversary $\mathcal{A}$ as follows: By Eqn. (4) we have $\mathsf{H}(id) = g^{x(id)} u_1^{y(id)}$ for some values $x(id)$ and $y(id)$ known to the simulator.
Case $y(id) = 0 \bmod p$: The simulator aborts (as in the last game).
Case $y(id) \neq 0 \bmod p$: The derived key $sk[id] = (d_1, d_2)$ is computed as follows:
For a random $r' \in \mathbb{Z}_p$, the simulator implicitly defines $r = -b/y(id) + r' \bmod p$ and computes

$$\begin{aligned} d_1 &\leftarrow (g^b)^{-x(id)/y(id)} g^{x(id)r'} u_1^{y(id)r'} , \\ d_2 &\leftarrow (g^b)^{-1/y(id)} \cdot g^{r'} . \end{aligned}$$

[6] Unfortunately, there seems not to be an efficient way to compute the exact value η. If there was one we could greatly simplify our analysis.

Note that the randomness r is not known to the simulator. Furthermore, the generation of the derived keys $sk[id] = (d_1, d_2)$ only depends on g^b and does not involve the knowledge of the secret key $\alpha = g^{ab}$ anymore.

Lemma A.4 $\Pr[X_5] = \Pr[X_4]$.

Proof: We have to verify that each derived key $sk[id] = (d_1, d_2)$ is identically distributed as in the last game. Let us abbreviate $x = x(id)$, and $y = y(id) \neq 0 \bmod p$. Clearly, if r' is uniform in $\mathbb{Z}_p$ so is r. Then by Eqn. (1) and since $r' = r + b/y$,

$$\begin{aligned} d_1 &= (g^b)^{-x/y} g^{xr'} u_1^{yr'} \\ &= g^{-bx/y} g^{xr'} u_1^{yr'} \\ &= g^{-bx/y} g^{x(r+b/y)} u_1^{y(r+b/y)} \\ &= g^{-bx/y} g^{xr+bx/y} u_1^{yr+b} \\ &= u_1^b \cdot g^{xr} u_1^{yr} \\ &= \alpha \cdot (g^x u_1^y)^r \\ &= \alpha \cdot (\mathsf{H}(id))^r \ , \end{aligned} \qquad \begin{aligned} d_2 &= (g^b)^{-1/y} \cdot g^{r'} \\ &= g^{-b/y} \cdot g^{r+b/y} \\ &= g^r \ , \end{aligned}$$

are distributed as in the last game (the original experiment). ∎

Game 6. (Change of the public key) In this game the simulator will modify the generation of the value u_2 from the public key mpk. The simulator picks a random $d \in \mathbb{Z}_p$ and computes the value u_2 as $u_2 = (g^a)^{-t^*} g^d$, where $t^* = \mathsf{TCR}(c_1^*)$. To summarize, the public key $mpk = (u_1, u_2, z, \mathsf{H})$ is now computed as

$$u_1 \leftarrow g^a, \quad u_2 \leftarrow (g^a)^{-t^*} g^d, \quad z \leftarrow \hat{e}(g^a, g^b) \ , \tag{6}$$

the hash keys as in Eqn. (3), and the secret key msk as $\alpha = g^{ab} = u_1^b$ that is still known to the simulator. The simulation of $\mathcal{A}$'s queries is done as before, using the secret key α. Note that the public key is identically distributed as in the last game. Therefore we have

$$\Pr[X_6] = \Pr[X_5] \ .$$

Game 7. (Forced abort during the game II) Compared to the last game we make the following changes to the simulator: When the tuple (id, C) is queried to the decapsulation oracle for $id \in ID \cup \{id^*\}$ and $C = (c_1, c_2, c_3)$ the simulator computes $t = \mathsf{TCR}(c_1)$ and immediately aborts if $y(id) = 0 \bmod p$, C is consistent, and $t = t^*$. In case of abort the simulator returns a random bit β'.

Lemma A.5 $|\Pr[X_7] - \Pr[X_6]| \leq \frac{2q}{p}$.

Proof: Clearly, this modification does not affect the adversary if there is no new forced abort. Note that any new forced abort implies $c_1 = c_1^*$ since otherwise by $t = t^*$ the simulator already aborted in the last game (having found a collision in the hash function TCR). In case of a new forced abort we distinguish between two cases:
Case 1: the new forced abort happens during the `guess` stage. Recall that we call a ciphertext $C = (c_1, c_2, c_3)$ consistent if $(g, c_1, u_1^t u_2, c_3)$ is a Diffie-Hellman tuple (where $t = \mathsf{TCR}(c_1)$), i.e. if

$(g, c_1, u_1^t u_2, c_3) = (g, g^r, u_1^t u_2, (u_1^t u_2)^r)$ for some value $r \in \mathbb{Z}_p$. Note that the way the public-key mpk is generated by Eqn. (6) and since $c_1 = c_1^*$, and $t = t^*$, for a consistent ciphertext C we have

$$c_3 = (u_1^t u_2)^r = ((g^a)^{t-t^*} g^d)^r = (c_1^a)^{t-t^*} \cdot c_1^d = (c_1^*)^d = c_3^* \,, \tag{7}$$

where $d \in \mathbb{Z}_p$ is only known to the simulator. If $id = id^*$ (i.e., if $\mathcal{A}$ queries the decapsulation oracle with the target identity) then $c_2^* = c_2$. Consequently $C = C^*$ and so the simulator rejects as in the original IB-KEM security experiment. If $id \neq id^*$ then, by definition, $id \in ID$ and the simulator outputs a random bit β' as in Game 6 where the abort was still done at the end of the experiment. Therefore, conditioned on case 1 this does not change the distribution of β' and we have $\Pr[X_7] = \Pr[X_6]$.

Case 2: the new forced abort happens during $\mathcal{A}$'s `find` stage. Since in the `find` stage the adversary has no information (in a statistical sense) about c_1^* from the challenge ciphertext C^*, and the adversary makes at most q decapsulation queries in its `find` stage, this implies

$$|\Pr[X_7] - \Pr[X_6]| \leq \frac{1}{p} + \frac{1}{p-1} + \ldots + \frac{1}{p-q+1} \leq \frac{q}{p-q} \leq \frac{2q}{p}$$

and concludes the proof. ∎

Game 8. (Change the answers to the decapsulation queries.) In the last game decapsulation queries were either aborted or answered using the secret key α, as in the original experiment. In this game the simulator changes its answers to its decapsulation queries $\textsc{Dec}(id, C)$ made by $\mathcal{A}$ as follows: By Eqn. (4) we have $\mathsf{H}(id) = g^{x(id)} u_1^{y(id)}$ for some values $x(id)$ and $y(id)$ known to the simulator.
Case $y(id) \neq 0 \bmod p$: the query is answered using $sk[id]$ obtained from the key derivation oracle.
Case $y(id) = 0 \bmod p$: the simulator simulates the decapsulation queries as follows: Let $C = (c_1, c_2, c_3,)$ be the queried ciphertext and let $t = \mathsf{TCR}(c_1)$.

If the ciphertext is not consistent then return a random message M
If $t = t^*$ then the simulator aborts (as in the last game)
If $t \neq t^*$ then
  If $\hat{e}(c_1, \mathsf{H}(id)) \neq \hat{e}(c_2, g)$ then return a random key K
  Else return $K \leftarrow \hat{e}(c_3/c_1^d, g^b)^{(t-t^*)^{-1}}$.

Lemma A.6 $\Pr[X_8] = \Pr[X_7]$.

Proof: Let $C = (c_1, c_2, c_3, E)$ be an arbitrary ciphertext submitted to the decapsulation oracle with respect to identity id. If $y(id) \neq 0 \bmod p$ then decapsulation is done using the simulation of the key derivation oracle which we already showed to be correct so we may now assume $y(id) = 0 \bmod p$. Furthermore we may assume C is consistent because otherwise a random message M gets returned, as in the last game.

Case 1a: $t = t^*$ and $c_1 \neq c_1^*$. In this case the simulator has found a collision in the hash function TCR and aborts as in the last game.
Case 1b: $t = t^*$ and $c_1 = c_1^*$. In the case the simulator aborts as in forced abort introduced in the last game.

Case 2: $t \neq t^*$. Similar to Eqn. (7) consistency of C implies

$$c_3 = (u_1^t u_2)^r = ((g^a)^{t-t^*} g^d)^r = (c_1^a)^{t-t^*} \cdot c_1^d \,, \tag{8}$$

and we obtain

$$(c_3/c_1^d)^{(t-t^*)^{-1}} = ((c_1^a)^{t-t^*} c_1^d / c_1^d)^{(t-t^*)^{-1}} = c_1^a \,. \tag{9}$$

In the original IB-KEM decapsulation algorithm first the secret key for identity id is computed as $sk[id] = (d_1, d_2) = (\alpha \cdot \mathsf{H}(id)^s, g^s)$ for random s, and then the session key K is reconstructed as

$$\begin{aligned} K = \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) &= \hat{e}(c_1, \alpha) \cdot \hat{e}(c_1, \mathsf{H}(id)^s)/\hat{e}(c_2, g^s) \\ &= \hat{e}(c_1^a, g^b) \cdot (\hat{e}(c_1, \mathsf{H}(id))/\hat{e}(c_2, g))^s \\ &\overset{(9)}{=} \hat{e}((c_3/c_1^d)^{(t-t^*)^{-1}}, g^b) \cdot (\hat{e}(c_1, \mathsf{H}(id)^s/\hat{e}(c_2, g))^s \\ &= \hat{e}(c_3/c_1^d, g^b)^{(t-t^*)^{-1}} \cdot (\Delta'(C))^s, \end{aligned}$$

with $\Delta'(C) = \hat{e}(c_1, \mathsf{H}(id))/\hat{e}(c_2, g)$. Since $(\Delta'(C))^s = 1$ if $\hat{e}(c_1, \mathsf{H}(id)) = \hat{e}(c_2, g)$ and $(\Delta'(C))^s$ is a random element in $\mathbb{G}_T$ otherwise, the decapsulated session key K in the original scheme is distributed as in the simulation. ▌

Game 9. (Modify the challenge) After $\mathcal{A}$'s `find` stage the simulator inputs the target identity id^* from $\mathcal{A}$. The simulator modifies the computation of the challenge ciphertext C^* follows:
Case $y(id^*) \neq 0 \bmod p$: The simulator aborts (as in the last game).
Case $y(id^*) = 0 \bmod p$: The simulator chooses a random bit γ and creates the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$ and key K_1^* as

$$c_1^* \leftarrow g^c, \quad c_2^* \leftarrow (g^c)^{x(id^*)}, \quad c_3^* \leftarrow (g^c)^d, \quad K_1^* \leftarrow \hat{e}(g, g)^{abc} . \tag{10}$$

By virtue of Eqns. (4), (8), and since $\mathsf{TCR}(c_1^*) = t^*$ and $y(id^*) = 0 \bmod p$, C^* is a correctly distributed ciphertext of K_1^*. Clearly,

$$\Pr[X_9] = \Pr[X_8] .$$

Game 10. (Replace the Challenge) The simulator replaces the value K_1^* from the challenge C^* with a random element from $\mathbb{G}_T$. Since K_1^* is now completely independent of the challenge bit γ, we have

$$\Pr[X_{10}] = 1/2 .$$

Observe that Game 10 does not use the secret key anymore and that the whole simulation only depends on the values g^a, g^b, g^c (i.e., the simulator "forgot the values a, b, and c). Game 9 and Game 10 are equal unless adversary $\mathcal{A}$ can distinguish $\hat{e}(g, g)^{abc}$ (the value of K_1^* in Game 9) from a random element in $\mathbb{G}_T$ (the value of K_1^* in Game 10). Therefore we have

$$|\Pr[X_{10}] - \Pr[X_9]| \leq \mathbf{Adv}^{\mathrm{bddh}}_{\mathcal{G},\mathcal{B}}(k) ,$$

for any adversary $\mathcal{B}$ against the hardness of BDDH running in the same time as the simulator, i.e. $\mathbf{Time}_{\mathcal{B}} = \mathbf{Time}_{\mathcal{A}} + \mathcal{O}(\rho^{-2} \ln(\rho^{-1}) \cdot \lambda^{-1} \ln(\lambda^{-1}) + q)$. The time bound of the simulator is justified by the $\rho^{-2} \ln(\rho^{-1}) \cdot \lambda^{-1} \ln(\lambda^{-1})$ samples to compute η' and the q answers to the key kerivation/decapsulation queries.

Analysis. We collect the probabilities relating the different games as follows:

$$\begin{aligned}
\mathbf{Adv}^{\text{ib-kem-cca}}_{\mathit{IBKEM},\mathcal{A}} &= |\Pr[X_0] - \frac{1}{2}| \\
&\leq |\Pr[X_1] + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) - \frac{1}{2}| \\
&\leq |\Pr[X_2] - 1/2 + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)| \\
&\leq |\frac{\Pr[\,X_3\,] - \frac{1}{2}}{\lambda} + \frac{\rho}{2} + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)| \\
&\leq \frac{|\Pr[\,X_7\,] + \frac{2q}{p} - \frac{1}{2}|}{\lambda} + \frac{\rho}{2} + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) \\
&\leq \frac{|\Pr[\,X_9\,] + \frac{2q}{p} - \frac{1}{2}|}{\lambda} + \frac{\rho}{2} + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) \\
&\leq \frac{\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k) + \frac{2q}{p}}{\lambda} + \frac{\rho}{2} + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)\,.
\end{aligned}$$

Using $\lambda = \frac{1}{4(n+1)q}$ (by Lemma A.2) and defining $\rho = \mathbf{Adv}^{\text{ib-kem-cca}}_{\mathit{IBKEM},\mathcal{A}}(k)$ we conclude the proof with

$$\begin{aligned}
\mathbf{Adv}^{\text{ib-kem-cca}}_{\mathit{IBKEM},\mathcal{A}}(k) &\leq 8(n+1)q \cdot (\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k) + 2q_2/p) + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) \\
&= \mathcal{O}\left(nq \cdot (\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k) + q/p) + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)\right)\,,
\end{aligned}$$

where q is an upper bound on all (derivation plus decapsulation) queries made by $\mathcal{A}$,

$$\begin{aligned}
\mathbf{Time}_{\mathcal{B}}(k) &= \mathbf{Time}_{\mathcal{A}}(k) + \mathcal{O}(\rho^{-2}\ln(\rho^{-1}) \cdot \lambda^{-1} \cdot \ln(\lambda^{-1})) \\
&= \mathbf{Time}_{\mathcal{A}}(k) + \mathcal{O}(\epsilon_{\mathcal{A}}^{-2}(k)\ln(\epsilon_{\mathcal{A}}^{-1}(k)) \cdot \lambda^{-1} \cdot \ln(\lambda^{-1}))\,,
\end{aligned}$$

where $\epsilon_{\mathcal{A}}(k) = \mathbf{Adv}^{\text{ib-kem-cca}}_{\mathit{IBKEM},\mathcal{A}}(k)$, and

$$\mathbf{Time}_{\mathcal{H}}(k) = \mathbf{Time}_{\mathcal{A}}(k) + \mathcal{O}(1)\,.$$

This concludes the proof of Theorem 4.1.

A.1 Proof of Lemma A.3

Proof: Let ArtAbort be the event that the simulator artificially aborts at the end of the simulation. Let Abort = ArtAbort ∨ ForcedAbort be the event that it aborts artificially or forced. First we claim

Claim A.7 For any fixed $\mathit{view}_{\mathcal{A}}$, $|\Pr[\neg\text{Abort}] - \lambda| \leq \lambda\rho/4$.

The proof of the claim is postponed until later.

Since the claim holds for any fixed $\mathit{view}_{\mathcal{A}}$ it also remains true for random $\mathit{view}_{\mathcal{A}}$, conditioned on $\gamma' = \gamma$ and $\gamma' \neq \gamma$:

$$|\Pr[\,\neg\text{Abort} \mid \gamma' = \gamma\,] - \lambda| \leq \lambda\rho/2, \quad |\Pr[\,\neg\text{Abort} \mid \gamma' \neq \gamma\,] - \lambda| \leq \lambda\rho/2 \tag{11}$$

We continue computing $\Pr[X_3]$:

$$\begin{aligned}\Pr[X_3] &= \Pr[\beta' = 1 \wedge \textsc{Abort}] + \Pr[\beta' = 1 \wedge \neg\textsc{Abort}] \\ &= \Pr[\beta' = 1 \mid \textsc{Abort}](1 - \Pr[\neg\textsc{Abort}]) + \Pr[\beta' = 1 \wedge \neg\textsc{Abort}]\end{aligned}$$

In case of abort the simulator outputs a random bit β'. If the simulator does not abort then it outputs $\beta' = 1$ if $\gamma = \gamma'$. Therefore we can continue with

$$\begin{aligned}\Pr[X_3] &= \frac{1}{2} \cdot (1 - \Pr[\neg\textsc{Abort}]) + \Pr[\gamma' = \gamma \wedge \neg\textsc{Abort}] \\ &= \frac{1}{2} + \frac{1}{2}(\Pr[\gamma' = \gamma \wedge \neg\textsc{Abort}] - \Pr[\gamma' \neq \gamma \wedge \neg\textsc{Abort}]) \\ &= \frac{1}{2} + \frac{1}{2}(\Pr[\neg\textsc{Abort} \mid \gamma' = \gamma]\Pr[\gamma' = \gamma] - \Pr[\neg\textsc{Abort} \mid \gamma' \neq \gamma]\Pr[\gamma' \neq \gamma]) .\end{aligned}$$

Since $\Pr[\gamma' = \gamma] = \Pr[X_2]$ we get

$$\begin{aligned}\Pr[X_3] - \frac{1}{2} &= \frac{1}{2}(\Pr[\neg\textsc{Abort} \mid \gamma' = \gamma]\Pr[X_2] - \Pr[\neg\textsc{Abort} \mid \gamma' \neq \gamma](1 - \Pr[X_2])) \\ &= \frac{1}{2} \cdot (\Pr[X_2] \cdot (\Pr[\neg\textsc{Abort} \mid \gamma' = \gamma] + \Pr[\neg\textsc{Abort} \mid \gamma' \neq \gamma]) - \Pr[\neg\textsc{Abort} \mid \gamma' \neq \gamma])\end{aligned}$$

Combining this with Eqn. (11) we get

$$\begin{aligned}\left|\Pr[X_3] - \frac{1}{2} - \lambda \cdot (\Pr[X_2] - \frac{1}{2})\right| &\leq \Pr[X_2] \cdot \lambda\frac{\rho}{4} + \lambda\frac{\rho}{4} \\ &\leq \lambda\frac{\rho}{2} ,\end{aligned}$$

where the last unequation stems from $0 \leq \Pr[X_2] \leq 1$. To prove the lemma it leaves to prove Claim A.7.

Proof of Claim A.7. By construction the two events ArtAbort and ForcedAbort are independent and consequently we have

$$\Pr[\neg\textsc{Abort}] = \Pr[\neg\textsc{ForcedAbort}] \cdot \Pr[\neg\textsc{ArtAbort}] = \eta \cdot \Pr[\neg\textsc{ArtAbort}] \tag{12}$$

Set $0 < \rho' := \rho/8 \leq 1/8$. Using Chernoff's bound for the estimate η' of η we get

$$\Pr[|\eta' - \eta| > \eta\rho'] < \lambda\rho'. \tag{13}$$

We have

$$\begin{aligned}\Pr[\neg\textsc{ArtAbort}] &= \Pr[\neg\textsc{ArtAbort} \mid |\eta' - \eta| > \eta\rho']\Pr[|\eta' - \eta| > \eta\rho'] \\ &\quad + \Pr[\neg\textsc{ArtAbort} \mid |\eta' - \eta| \leq \eta\rho']\Pr[|\eta' - \eta| \leq \eta\rho'] \\ &\leq \lambda\rho' + \Pr[\neg\textsc{ArtAbort} \mid |\eta' - \eta| \leq \eta\rho'] \\ &= \lambda\rho' + \frac{\lambda}{\eta'},\end{aligned}$$

where the last equation is true since for fixed η' with $|\eta' - \eta| \leq \eta\rho'$ we have $\eta' > \eta(\rho' + 1) \geq \lambda(\rho' + 1) \geq \lambda$ and therefore $\Pr[\neg\textsc{ArtAbort}] = \frac{\lambda}{\eta'}$. We continue with

$$\begin{aligned}\Pr[\neg\textsc{Abort}] &= \Pr[\neg\textsc{ForcedAbort}] \cdot \Pr[\neg\textsc{ArtAbort}] \\ &\leq \eta\lambda\rho' + \frac{\eta\lambda}{\eta'} \leq \lambda\rho' + \frac{\lambda}{1 - \rho'} \leq \lambda(1 + 2\rho'),\end{aligned}$$

since $0 \leq \rho' \leq 1/8$. For all fixed η' with $|\eta' - \eta| \leq \eta\rho'$ we have $\Pr[\neg\textsc{ArtAbort}] = \min\{1, \frac{\lambda}{\eta'}\} > \frac{\lambda}{\eta(1+\rho')}$ (since $\eta > \lambda$ and hence $\eta \cdot (1+\rho') > \lambda$). Therefore

$$\begin{aligned}
\Pr[\neg\textsc{Abort}] &= \eta \cdot \Pr[\neg\textsc{ArtAbort}] \\
&\geq \eta \cdot \Pr\left[\neg\textsc{ArtAbort} \mid |\eta' - \eta| \leq \eta\rho'\right] \cdot \Pr\left[|\eta' - \eta| \leq \eta\rho'\right] \\
&\geq \eta \cdot \frac{\lambda}{\eta(1+\rho')} \cdot (1 - \lambda\rho') \geq \lambda \cdot (1-\rho')^2 \geq \lambda \cdot (1 - 2\rho')
\end{aligned}$$

We showed $\lambda(1-2\rho') \leq \Pr[\neg\textsc{Abort}] \leq \lambda(1+2\rho')$ which implies $|\Pr[\neg\textsc{Abort}] - \lambda| \leq \lambda \cdot 2\rho' < \lambda \cdot \frac{\rho}{4}$ which proves the claim. ∎

A.2 Proof of Lemma A.2

Proof: Fix the queried identities $id^{(1)}, \ldots, id^{(q_0)}, id^*$. We want to show that

$$\eta = \Pr_{\mathbf{Y}}[\bigwedge_{i=1}^{q_0} \Big(y(id^{(i)}) \neq 0 \bmod p\Big) \wedge y(id^*) = 0 \bmod p] \geq \lambda\,, \tag{14}$$

for $\lambda = \frac{1}{4(n+1)q}$.

Over the integers we have by Eqn. (3), $y(id^*) = 0 = p - km + y'_0 + \sum_{i=1}^n id_i^* y_i$, where $0 \leq y'_0 + \sum_{i=1}^n id_i^* y_i < (n+1)m$. This shows that if $y(id^*) = 0 \bmod m$, then there is a unique $0 \leq k < n+1$ such that $y(id^*) = 0$ over the integers. Since k is uniformly and independently distributed between 0 and n, we have that :

$$\begin{aligned}
\eta &= \Pr_{\mathbf{Y}}[\bigwedge_{i=1}^{q_0} (y(id^{(i)}) \neq 0 \bmod p) \wedge y(id^*) = 0 \bmod p] \\
&\geq \frac{1}{n+1} \cdot \Pr_{\mathbf{Y}'}[\bigwedge_{i=1}^{q_0} (y(id^{(i)}) \neq 0 \bmod m) \wedge y(id^*) = 0 \bmod m]\,,
\end{aligned}$$

were $\mathbf{Y}' = (y'_0, y_1, \ldots, y_n)$ and the random variables $(y'_0, y_1, \ldots, y_n)$ are distributed as in (3), i.e. from now on we consider k to be fixed.

Let $id \neq id'$ and $a, b \in \mathbb{Z}$. We collect some simple observations on $y(\cdot)$:

$$\Pr_{\mathbf{Y}'}[y(id) = b \bmod m] = 1/m \tag{15}$$

$$\Pr_{\mathbf{Y}'}[y(id) = a \bmod m \mid y(id') = b \bmod m] = 1/m \tag{16}$$

$$\Pr_{\mathbf{Y}'}[y(id) = a \bmod m) \wedge y(id') = b \bmod m)] = 1/m^2\,. \tag{17}$$

Eqn. (15) follows since for any choice of $y_1, ..., y_n$ there is a single choice of y'_0 that will make the condition hold. To show Eqn. (16) assume there exists an index $1 \leq i \leq n$ such that $id_i = 1$ and $id'_i = 0$. Then fix all y_j's for $j \neq i$ except y_i so that $y(id') = b$. Therefore $\Pr\left[\, y(id) = a \mid y(id') = b \,\right] = 1/m$. If there is no such i then we can use Bayes to reverse roles of id and id'. Eqn. (17) basically means that the $y(\cdot) \bmod m$ are pairwise independent and follows directly from Eqn. (15) and Eqn. (16).

We continue to bound η with

$$\begin{aligned}
\eta &\geq \frac{1}{n+1} \cdot \Pr_{\mathbf{Y}'}[\bigwedge_{i=1}^{q_0} y(id^{(i)}) \neq 0 \bmod m \mid y(id^*) = 0 \bmod m] \cdot \Pr[y(id^*) = 0 \bmod m] \\
&\overset{(15)}{=} \frac{1}{(n+1)m} \cdot (1 - \Pr_{\mathbf{Y}'}[\bigvee_{i=1}^{q_0} y(id^{(i)}) = 0 \bmod m \mid y(id^*) = 0 \bmod m]) \\
&\overset{(17)}{=} \frac{1}{(n+1)m} \cdot (1 - \sum_{i=1}^{q_0} \Pr_{\mathbf{Y}'}[y(id^{(i)}) = 0 \bmod m \mid y(id^*) = 0 \bmod m]) \\
&\overset{(16)}{=} \frac{1}{(n+1)m} \cdot (1 - \sum_{i=1}^{q_0} \frac{1}{m}) \\
&\geq \frac{1}{(n+1)m} \cdot (1 - \frac{q}{m}) \\
&= \frac{1}{4(n+1)q} ,
\end{aligned}$$

where the last equation follows by our choice of $m = 2q$ which minimizes the term. ∎

B Security of the Threshold IB-KEM

B.1 Proof of Theorem 5.4

We now prove consistency of the threshold IB-KEM from Section 5. Fix id and an arbitrary $C = (c_1, c_2, c_3)$ consistent with id, i.e. we have $c_1 = g^v$, $c_2 = \mathsf{H}(id)^v$, and $c_3 = (u_1^t u_2)^v$. First we show key consistency. Let $D_i = (d_{i,1}, d_{i,2})$ be two sets of private key shares. Note that the key derivation shares verification algorithm $\mathsf{TIBKEMkey.Vfy}(pk, i, id, D_i)$ checks if $\hat{e}(d_{i,1}, g) = \hat{e}(vk_i, u_1) \cdot \hat{e}(d_{i,2}, \mathsf{H}(id))$. Fix $d_{i,2} = g^{s_i}$. The above expression implies that any pair $(d_{i,1}, d_{i,2})$ passing the test satisfies $d_{i,1} = u_1^{F(i)} \cdot \mathsf{H}(id)^{s_i}$. Therefore the $\mathsf{TIBKEMkey.Combine}$ algorithms returns $sk[id] = (d_1, d_2) = (u_1^a \cdot \mathsf{H}(id)^s, g^s)$ with $s = \sum s_i \lambda_i$. Analogously, for the set D' it returns $sk'[id] = (d_1', d_2') = (u_1^a \cdot \mathsf{H}(id)^{s'}, g^{s'})$ for some other s'. By correctess we know that the two valid private keys for identity id decrypt the same ciphertext C to the same session key and therefore we have $K' = K$ and any (even computationally unbounded adversary) has zero advantage in breaking key consistency.

Decryption consistency is proved in an analogous manner. Let $S = \{C_1, \ldots, C_l\}$ with $C_i = (C_{i,1}, C_{i,2}, C_{i,3})$. The decryption shares verification algorithm $\mathsf{TIBKEMdec.Vfy}(vk, i, id, C_i, C)$ checks if $\hat{e}(g, C_{i,2}) = \hat{e}(vk_i, u_1) \cdot \hat{e}(C_{i,3}, \mathsf{H}(id)) \cdot \hat{e}(C_{i,1}, u_1^t u_2)$. Fix $C_{i,1} = g^{r_i}$ and $C_{i,3} = g^{s_i}$. Now it is easy to verify that the above equation implies that $C_{i,2} = u_1^{F(i)} \cdot \mathsf{H}(id)^{s_i} \cdot (u_1^t u_2)^{r_i}$, i.e. all the ciphertext shares C_i are identically distributed as if they were correctly generated. The same holds for the ciphertext shares C_i' generated from the set S'. Finally, by correctness of the scheme, the $\mathsf{TIBKEMdec.Combine}$ algorithm recovers the same key $K = K'$ in both cases. This completes the proof.

B.2 Proof of Theorem 5.3

The proof is similar to the proof of Theorem 4.1. Assuming Waters hash function H "behaves well" (i.e. the event ForcedAbort does not happen), we can perfectly simulate all oracle queries in a similar manner as in the proof of Theorem 4.1. More precisely, by using the properties of the Lagrange interpolation polynomial, the KeyShare$(\cdot, \cdot)$ queries for non-corrupt users are simulated

slightly changing the KeyDer$(\cdot)$ outputs and the DecShare$(\cdot,\cdot,\cdot)$ queries are simulated slightly changing Dec$(\cdot,\cdot)$ answers. In the following the games based proof for Theorem 5.3 is presented. Note that some of the games are identical to the proof of Theorem 4.1; in this case we refer the reader to Appendix A.

Game 0′. Threshold key Generation. Without loss of generality, let $I_c = \{1, \ldots, l-1\}$ be the set of corrupted players. Initially the simulator runs the TIBE key generation algorithm $\mathsf{TIBKEMkg}(1^k, l, m)$ and obtains the public key $mpk = (u_1, u_2, z, \mathsf{TCR}, \mathsf{H})$, the verification key vk and secret key shares $sk = \{sk_i\}_{1 \le i \le m}$. We make the convention that the public key is generated as

$$u_1 \leftarrow g^a, \quad u_2 \xleftarrow{\$} \mathbb{G}_1, \quad z \leftarrow \hat{e}(g^a, g^b), \quad \mathsf{H} \xleftarrow{\$} \mathsf{HGen}(\mathbb{G}_1)\ , \tag{18}$$

depending on the elements g^a, g^b. Note that the way the value $z = \hat{e}(g^a, g^b) = \hat{e}(g, g^{ab})$ from the public key is generated implies $\alpha = g^{ab} = u_1^b$. The secret key α can be computed since a and b are known to the simulator. The verification and secret key shares are computed by following the specification of the $\mathsf{TIBKEMkg}$ algorithm. That is, using the knowledge of b, the simulator explicitly computes a polynomial $F[X] \in \mathbb{Z}_p[X]$ of degree $l-1$, such that $F(0) = b$. The secret and verification keys are generated as

$$\text{For } i \in \{1, \ldots, m\}: \ sk_i \leftarrow u_1^{F(i)}\ ;\ vk_i \leftarrow g^{F(i)}\ ;\ vk \leftarrow (vk_1, \ldots, vk_m)\ ;\ sk \leftarrow (sk_1, \ldots, sk_m) \tag{19}$$

The public, verification and secret key shares for corrupted players are given to the adversary in its find phase.

Find Phase. During its execution adversary $\mathcal{A}$ makes a number of key derivation and decryption share requests. If the adversary makes a key derivation share query to oracle KeyShare(j, id), then the simulator computes the secret key share $sk[id]_j$ by using the knowledge of the sharing polynomial $F(X)$, and returns it to the adversary. If the adversary makes a decryption share query DecShare(j, id, C) the simulator (using again $F(X)$) computes the jth decryption share and gives it to the adversary.

The rest of the game is as described in Game 0, including definitions. Now, for $0 \le i \le 10$, we define the event X_i' : The simulator outputs $\beta' = 1$ in Game i'. We have $|\Pr[X_0'] - 1/2| = \mathbf{Adv}^{\textit{tibkem-cca}}_{\mathcal{TIBKEM},\mathcal{A}}$.

Game 1′. (Eliminate hash collisions). As in Game 1.

Game 2′. (Change of the hash keys). As in Game 2. Now, the hash function H is computed as $\mathsf{H}(id) = g^{x(id)} u_1^{y(id)}$, where $x(id)$ and $y(id)$ are values only known by the simulator.

Game 3′. (Abort at the end of the game). As in Game 3.

Game 4′. (Forced abort during the game I) As in Game 4. The relation between X_0' and X_4' are as in Section A, i.e.

$$\left| \frac{1}{2} - \Pr[X_0'] - \frac{\Pr[X_4'] - 1/2}{\lambda} \right| \le \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) + \frac{\rho}{2}.$$

Game 5′. (Change key derivation oracle) The simulator changes its answers to all key derivation queries KeyShare(id, j) $(1 \le j \le m)$ made by the adversary $\mathcal{A}$ as follows: We have $\mathsf{H}(id) = g^{x(id)} u_1^{y(id)}$ for some values $x(id)$ and $y(id)$ known to the simulator.
Case $y(id) = 0 \bmod p$: The simulator aborts.
Case $y(id) \neq 0 \bmod p$: The derived key $sk[id]_j = (d_{j,1}, d_{j,2})$ is computed as follows:
For a random $r' \in \mathbb{Z}_p$, the simulator implicitly defines $r = -F(j)/y(id) + r' \bmod p$ and computes

$$\begin{aligned} d_{j,1} &\leftarrow (vk_j)^{-x(id)/y(id)} g^{x(id)r'} u_1^{y(id)r'} \\ d_{j,2} &\leftarrow (vk_j)^{-1/y(id)} \cdot g^{r'} \end{aligned}$$

Note that the randomness r is not known to the simulator and that the generation of the derived keys shares $sk[id]_j$ does not involve the knowledge of the secret key share $sk_j = g^{aF(j)}$ anymore. Using a similar argument as in Lemma A.4, the key derivation shares generated as above are distributed as in the previous game. Therefore

$$\Pr[X'_4] = \Pr[X'_5] \ .$$

Game 6′. (Change of public key and secret key shares.) The simulator modifies the generation of the public key $mpk = (u_1, u_2, z, \mathsf{TCR}, \mathsf{H})$, and only uses the values g^a, g^b and g^c as in Game 6 in the proof of Theorem 4.1 as follows (cf. Eqn. (6)):

$$u_1 \leftarrow g^a, \quad u_2 \leftarrow (g^a)^{-t^*} g^d, \quad z \leftarrow \hat{e}(g^a, g^b) \ , \tag{20}$$

where $d \xleftarrow{\$} \mathbb{Z}_p^*$ is chosen by the simulator.

In the threshold case the simulator additionally has to compute secret key shares and the verification shares. In the last games the was done using b to compute the polynomial $F(X)$, as in the original key generation algorithm. We now modify the simulator such that the generation of the secret key shares of corrupted parties, as well as the generation of the verification key does only depend on the values g^a and g^b.

1. Generate secret key shares for the $l-1$ corrupted players. This is done by picking $f_1, \ldots, f_{l-1} \xleftarrow{\$} \mathbb{Z}_q$. Let $F \in \mathbb{Z}_p[X]$ of degree $l-1$ be the unique polynomial implicitly defined by $F(0) = b$ and $F(j) = f_j$ for $1 \le j \le l-1$. The simulator computes $sk_j = (g^a)^{f_j}$ for $1 \le j \le l-1$. Note that the simulator does not know F since it does not know b.

2. Generate verification key $vk = (vk_1, ..., vk_m)$ such that $vk_j = g^{F(j)}$. For $1 \le j < l$, the simulator knows $f_j = F(j)$, so that vk_j can be computed as g^{f_j}. Next, for every index j such that $l \le j \le m$, the simulator computes the Lagrange Coefficients $\lambda_0, \ldots, \lambda_{l-1}$ such that

$$F(j) = \sum_{i=0}^{l-1} \lambda_i F(i) \ . \tag{21}$$

   Following Eqn. (21), each $vk_j = g^{F(j)}$ can be computed as $vk_j = (g^b)^{\lambda_0} \cdot vk_1^{\lambda_1} \cdots vk_{l-1}^{\lambda_{l-1}}$.

From now on the simulator does not know the secret key shares $\{sk_j\}_{\{l \le j \le m\}}$ of the uncorrupted players. Note that the public and verification keys, as well as the secret key shares for the corrupted players are distributed as in the previous game. Therefore we have

$$\Pr[X'_6] = \Pr[X'_5] \ .$$

Game 7′.(Forced abort during the game II) As in Game 7. We have $|\Pr[X'_7] - \Pr[X'_6]| \le \frac{2q}{p}$.

Game 8′. (Change the answers to the decryption share queries.) The simulator changes its answers to all decryption queries $\textsc{DecShare}(j, id, C)$ made by $\mathcal{A}$ as follows: By Eqn. (4) we have $\mathsf{H}(id) = g^{x(id)} u_1^{y(id)}$ for some values $x(id)$ and $y(id)$ known to the simulator.
Case $y(id) \neq 0 \bmod p$: the query is answered using the key derivation share oracle.
Case $y(id) = 0 \bmod p$. Then $\mathsf{H}(id) = g^{x(id)}$ for a value $x(id)$ known to the simulator. Decryption share queries are simulated as follows: Let $C = (c_1, c_2, c_3)$ be the queried ciphertext and let $t = \mathsf{TCR}(c_1)$.

If the ciphertext is not consistent with id and t then return `fail`
If $t = t^*$ then abort (as before)

If $t \neq t^*$ then
  if $s_j(id)$ is undefined then $s_j(id) \xleftarrow{\$} \mathbb{Z}_p$
  $s_j \leftarrow s_j(id)$; $r'_j \xleftarrow{\$} \mathbb{Z}_p$
  $C_{j,1} \leftarrow vk_j^{-1/(t-t^*)} \cdot g^{r'_j}$
  $C_{j,2} \leftarrow g^{x(id)s_j} \cdot (u_1^t u_2)^{r'_j} \cdot vk_j^{-d/(t-t*)}$
  $C_{j,3} \leftarrow g^{s_j}$

Note that the generation of the decryption shares for player j only depends on vk_j and does not involve the knowledge of the secret key share sk_j anymore. We also ensure that for a fixed identity id the same randomness $s_j = s_j(id)$ is used to compute the decryption shares for player j.

Lemma B.1 $\Pr[X'_8] = \Pr[X'_7]$.

Proof: We have to verify that each ciphertext share $C_j = (C_{j,1}, C_{j,2}, C_{j,3})$ is identically distributed as in the last game. Let us abbreviate $x = x(id)$. As in Game 8 in Appendix A we only have to consider the interesting case $y(id) = 0 \bmod p$ and $t \neq t^*$. Implicitly define $r_j = r'_j - F(j)/(t-t^*)$. Clearly, if r'_j is uniform in $\mathbb{Z}_p$ so is r_j. Then by Eqn. (20),

$$\begin{aligned}
C_{j,1} &= (g^{F(j)})^{-1/(t-t^*)} \cdot g^{r'_j} \\
&= g^{-F(j)/(t-t^*)} \cdot g^{r_j + F(j)/(t-t^*)} \\
&= g^{r_j} \ , \\
C_{j,2} &= g^{xs_j} \cdot (u_1^t u_2)^{r'_j} \cdot vk_j^{-d/(t-t*)} \\
&= g^{xs_j} \cdot g^{(a(t-t^*)+d)r'_j} \cdot g^{-F(j)d/(t-t*)} \\
&= g^{xs_j} \cdot g^{a(t-t^*)r'_j} \cdot g^{dr'_j} \cdot g^{-F(j)d/(t-t*)} \\
&= g^{xs_j} \cdot g^{a(t-t^*)r_j} \cdot g^{aF(j)} \cdot g^{dr_j} \cdot g^{F(j)d/(t-t^*)} \cdot g^{-F(j)d/(t-t*)} \\
&= g^{xs_j} \cdot g^{(a(t-t^*)+d)r_j} \cdot g^{aF(j)} \ , \\
&= \mathsf{H}(id)^{s_j} \cdot (u_1^t u_2)^{r_j} \cdot sk_j \ ,
\end{aligned}$$

are distributed as in the last game. ∎

Game 9′. (Modify the challenge.) As in Game 9. We have $\Pr[X'_9] = \Pr[X'_8]$.

Game 10′. (Replace the challenge.) As in Game 10. We have $|\Pr[X'_{10}] - \Pr[X'_9]| \leq \mathbf{Adv}^{\mathrm{bddh}}_{\mathcal{G},\mathcal{B}}(k)$ and $\Pr[X'_{10}] = 1/2$, where $\mathcal{B}$ denotes any adversary against the security of BDDH running in the same time as the simulator.

Analysis. Now relating the probabilities as in Appendix A completes the proof.

C The IBE scheme from BMW [14]

Let $\mathsf{CR} : \mathbb{G}_1^2 \times \mathbb{G}_T \to \{0,1\}^m$ be a collision-resistant hash function, where we assume $m \approx n$. The BMW IBE scheme with identity space $\mathsf{IDSp} = \{0,1\}^n$ and message space $\mathsf{MsgSp} = \mathbb{G}_1$ is depicted in Figure 3.

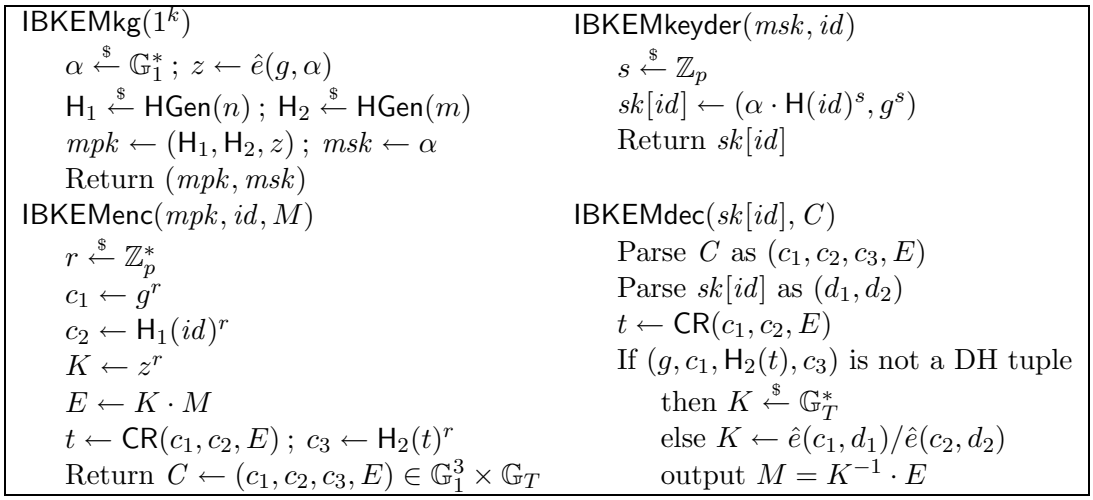
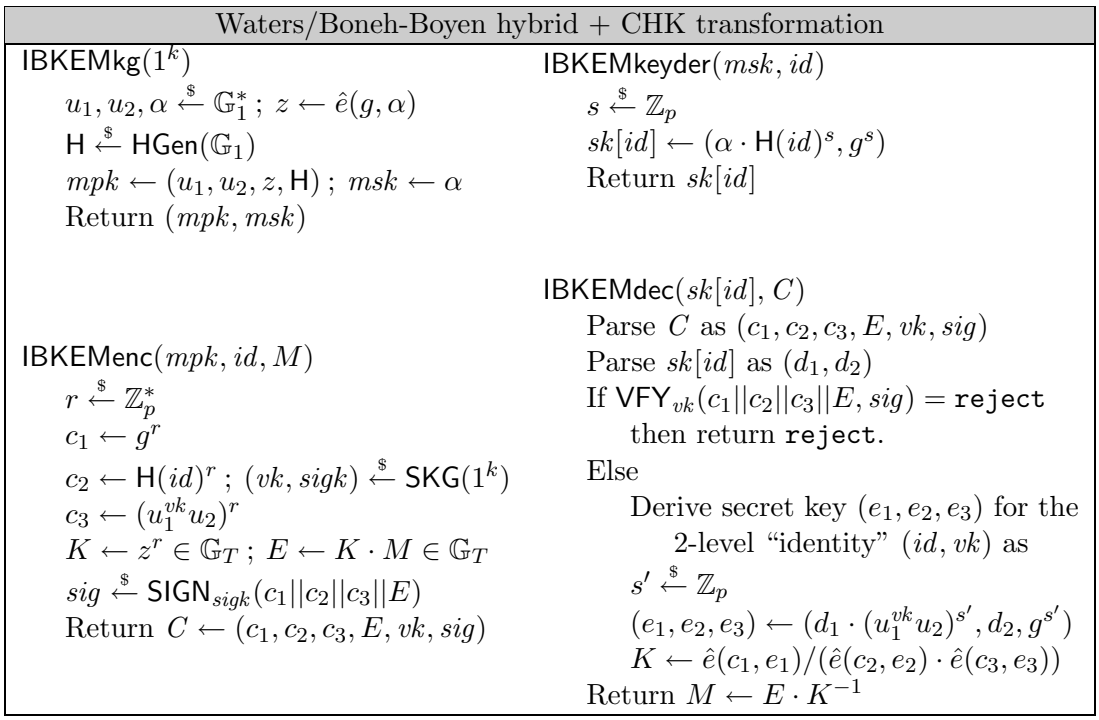


Figure 3: IBE scheme from BMW.

C.1 IBE scheme obtained by the generic CHK transformation

This construction employs a one-time signature scheme $\mathcal{OTS} = (\mathsf{SKG}, \mathsf{SIGN}, \mathsf{VFY})$. The key generation algorithm SKG is run to obtain a random pair of verification/signing keys $(vk, sigk) \xleftarrow{\$} (\mathsf{SKG}(1^k)$; the signing key $sigk$ is used to sign a message M to obtain a signature $sig \xleftarrow{\$} \mathsf{SIGN}_{sigk}(M)$ on a message M; using the public verification key vk, a signature sig can be verified by running $\mathsf{VFY}_{vk}(M, sig)$. We require that this scheme be secure in the sense of *strong unforgeability*, see [16] for exact definitions and constructions (details can be skipped here).



The following theorem combines results from [16] with [45, 8]:

Theorem C.1 Assume $\mathcal{OTS}$ is a strongly-secure one-time signature scheme. Under the Biliear Decisional Diffie-Hellman (BDDH) assumption relative to generator $\mathcal{G}$, the IBE scheme is secure against

chosen-ciphertext attacks. In particular, assuming BDDH is ϵ-hard, then the IBE scheme is $\approx \mathcal{O}(nq\epsilon)$-secure.