Construction of a Hybrid Hierarchical Identity Based Encryption Protocol Secure Against Adaptive Attacks (Without Random Oracle)

Palash Sarkar and Sanjit Chatterjee

Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108.
e-mail:{palash,sanjit_t}@isical.ac.in

Abstract. We describe a hybrid hierarchical identity based encryption (HIBE) protocol which is secure in the full model without using the random oracle heuristic and whose security is based on the computational hardness of the decisional bilinear Diffie-Hellman (DBDH) problem. The new protocol is obtained by augmenting a previous construction of a HIBE protocol which is secure against chosen plaintext attacks (CPA-secure). The technique for answering decryption queries in the proof is based on earlier work by Boyen-Mei-Waters. Ciphertext validity testing is done indirectly through a symmetric authentication algorithm in a manner similar to the Kurosawa-Desmedt public key encryption protocol. Additionally, we perform symmetric encryption and authentication by a single authenticated encryption algorithm. A net result of all these is that our construction improves upon previously known constructions in the same setting.

1 Introduction

Identity based encryption [28, 8] is a kind of public key encryption where the public key can be the identity of the receiver. The secret key corresponding to the identity is generated by a private key generator (PKG) and is securely provided to the relevant user. The notion of IBE simplifies the issues of certificate management in public key infrastructure. The PKG issues the private key associated with an identity. The notion of hierarchical IBE (HIBE) [21, 19] was introduced to reduce the workload of the PKG. The identity of any entity in a HIBE structure is a tuple $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$. The private key corresponding to such an identity can be generated by the entity whose identity is $(\mathsf{v}_1, \ldots, \mathsf{v}_{j-1})$ and which possesses the private key corresponding to this identity. The security model for IBE was extended to that of HIBE in [21, 19].

The first construction of an IBE which can be proved to be secure in the full model without the random oracle heuristic was given by Boneh and Boyen in [5]. Later, Waters [30] presented an efficient construction of an IBE which is secure in the same setting. An extension of Waters' construction has been independently described in [13] and [26]. This leads to a controllable trade-off between the size of the public parameters and the efficiency of the protocol (see [13] for details).

A construction of a HIBE secure in the full model without using the random oracle heuristic was suggested in [30]. A recent work [14], describes a HIBE which builds on [30] by reducing the number of public parameters. The constructed HIBE is secure against chosen plaintext attacks (CPA-secure).

The Problem: We consider the problem of constructing a HIBE under the following conditions.

– Security is in the full model [8], i.e., the adversary can mount an adaptive chosen ciphertext attack and can choose the challenge identity adaptively.
– The reduction is from the decisional bilinear Diffie-Hellman problem.
– The security proof does not use the random oracle heuristic.

1.1 Our Contributions

We describe a hybrid HIBE protocol for the above setting. The new construction is obtained by augmenting the construction in [14]. The idea for this augmentation is based on the technique of [9] and algebraic ideas from the construction of IBE given in [4]. In addition, we make use of two new things. First, we incorporate information about the length of the identity into the ciphertext. Second, we use symmetric key authentication to verify ciphertext well formedness. We also show that the two tasks of symmetric key encryption and authentication can be combined by using an authenticated encryption (AE) protocol.

The idea of using symmetric authentication technique to verify the well formedness of the ciphertext is based on the hybrid PKE protocol due to Kurosawa-Desmedt (KD) [25]. To the best of our knowledge, this technique has not been earlier applied to the (H)IBE setting.

We can specialize the HIBE protocol described in this paper to obtain a PKE and an IBE. With some natural simplifications, the PKE turns out to be the key encapsulation mechanism (KEM) proposed by BMW [9] composed with a one-time secure data encapsulation mechanism (DEM). On the other hand, the IBE is different from previous work. Kiltz-Galindo [24] had proposed an IB-KEM. Composed with a suitable symmetric encryption algorithm, this provides an IBE. The decryption algorithm of our IBE is faster than the IBE obtained from the KEM given in [24].

Our construction has a security degradation of approximately q^h (where q is the number of queries and h is the number of levels). This is better than a degradation of q^{h+1} which is what one would obtain by a straightforward application of the known techniques. Another advantage is that by instantiating the AE protocol with a single pass algorithm [27, 22, 20, 12], it is possible to obtain a speed-up by a factor of two for both encryption and decryption of the symmetric part of the hybrid encryption. Also, by using the authentication aspect of the AE protocol for verifying the well formedness of the ciphertext we can avoid a number of pairing based verifications. This leads to a faster decryption algorithm.

We make a few remarks on the proof. Since the new protocol is obtained by augmenting the protocol in [14], the proof of the new protocol is also obtained by augmenting the proof in [14] (which is actually based on the construction and proof in [30]). We do not repeat the aspects of the proof that already appear in [14]. Incorporating the length of the identity in the ciphertext is required to avoid certain attacks as we discuss later. Verifying ciphertext well formedness using symmetric authentication requires us to adapt the proof technique (especially the method of deferred analysis) of [1] to the identity based setting. The combination of different techniques introduces several subtleties in the proof.

1.2 Related Work

The construction in [19] is based on the random oracle assumption and does not constitute a solution to the problem considered in this paper. A generic technique [11, 7] is known which converts an $(h+$

1)-level CPA-secure HIBE protocol into an h-level CCA-secure HIBE protocol while preserving the other features (security model, with/without random oracle, hardness assumption) of the original CPA-secure protocol. This technique is based on one-time signatures and requires prepending each identity component by a bit. Applying this technique directly to the protocol in [14] does not provide a protocol which is more efficient than the protocol we describe in this paper.

The BMW paper [9] provided a method of constructing a PKE from an IBE. They also mentioned that the technique can be used for constructing (H)IBE. Later work by Kiltz-Galindo [24] built on the BMW paper and described an efficient CCA-secure IB-KEM. The KG paper suggested a method for extending their IB-KEM to a HIB-KEM. Details were provided in [3]. Our work also uses the BMW technique, but introduces several other ideas to obtain a more efficient (H)IBE compared to previous work.

In an interesting paper, Boneh-Boyen-Goh [6] have shown how to construct a constant size ciphertext (H)IBE based on the weak decisional bilinear Diffie-Hellman exponent problem which is a variant of the DBDH problem. Their protocol is CPA-secure in the selective-ID model. Using the technique of Waters, this protocol can be made CPA-secure in the full model. Further, using the techniques of Boyen-Mei-Waters this can be converted into a CCA-secure protocol. For details of this conversion and also for a protocol secure in a different model see [15]. The work [23] also considers the same problem.

The main difference between the current work and that of [15, 23] is that the hardness assumptions are different. This makes a direct comparison difficult. We, however, note that the ciphertext expansion in the later is constant while in the former it increases linearly with the number of components in the identity. This is due to the fact that the assumption used in [15, 23] is tailored to ensure constant size ciphertext. On the other hand, the number of public parameters in the current construction is significantly less than the number of public parameters in [15, 23]. This is due to the fact that the current protocol is built using the protocol in [14] which significantly reduces the number of public parameters.

2 Preliminaries

2.1 HIBE Protocol

Following [21, 19], a hierarchical identity based encryption (HIBE) scheme is specified by four algorithms: Setup, KeyGen, Encrypt and Decrypt. For a HIBE of height h (henceforth denoted as h-HIBE) any identity v is a tuple $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$ where $1 \leq j \leq h$.

- $\mathsf{HIBE.Setup}$: Takes as input a security parameter and outputs (pk, sk), where pk is the public parameter of the PKG and sk is the master secret of the PKG. It also defines the domains of identities, messages and ciphertexts.
- $\mathsf{HIBE.KeyGen}(\mathsf{v}, d_{\mathsf{v}|_{j-1}}, pk)$: Takes as input a j-level identity v, the secret $d_{\mathsf{v}|_{j-1}}$ corresponding to its $(j-1)$-level prefix and pk and returns as output d_{v}, the secret key corresponding to v. In case $j = 1$, $d_{\mathsf{v}|_{j-1}}$ is equal to sk, the master secret of the PKG.
- $\mathsf{HIBE.Encrypt}(\mathsf{v}, M, pk)$: Takes as input v, the message M and pk, and returns C, the ciphertext obtained by encrypting M under v and pk.
- $\mathsf{HIBE.Decrypt}(\mathsf{v}, d_{\mathsf{v}}, C, pk)$: Takes as input v, the secret key d_{v} corresponding to v, a ciphertext C and pk. Returns either bad or M, the message which is the decryption of C.

As usual, for soundness, we require that $\mathsf{HIBE.Decrypt}(\mathsf{v}, d_{\mathsf{v}}, C, pk) = M$ must hold for all v, d_{v}, C, pk, sk and M associated by the above four algorithms.

2.2 Security Model for HIBE

Security is defined using an adversarial game. An adversary $\mathcal{A}$ is allowed to query two oracles – a decryption oracle and a key-extraction oracle. At the initiation, it is provided with the public parameters of the PKG. The game has two query phases with a challenge phase in between.

Query Phase 1: Adversary $\mathcal{A}$ makes a finite number of queries where each query is addressed either to the decryption oracle or to the key-extraction oracle. In a query to the decryption oracle it provides a ciphertext as well as the identity under which it wants the decryption. It gets back the corresponding message or bad if the ciphertext is invalid. Similarly, in a query to the key-extraction oracle, it asks for the private key of the identity it provides and gets back this private key. Further, $\mathcal{A}$ is allowed to make these queries adaptively, i.e., any query may depend on the previous queries as well as their answers. The adversary is not allowed to make any useless queries, i.e., queries for which it can compute the answer itself. For example, the adversary is not allowed to ask for the decryption of a message under an identity if it has already obtained a private key corresponding to the identity.

Challenge: At this stage, $\mathcal{A}$ outputs an identity $\mathsf{v}^* = (\mathsf{v}_1^*, \ldots, \mathsf{v}_j^*)$ for $1 \leq j \leq h$, and a pair of messages M_0 and M_1. There is the natural restriction on the adversary, that it cannot query the key extraction oracle on v^* or any of its proper prefixes in either of the phases 1 or 2. A random bit δ is chosen and the adversary is provided with C^* which is an encryption of M_δ under v^*.

Query Phase 2: $\mathcal{A}$ now issues additional queries just like Phase 1, with the (obvious) restrictions that it cannot ask the decryption oracle for the decryption of C^* under v^*, nor the key-extraction oracle for the private key of v^* or any of its prefixes.

Guess: $\mathcal{A}$ outputs a guess δ' of δ.
The advantage of the adversary $\mathcal{A}$ is defined as:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{HIBE}} = |\mathsf{Pr}[(\delta = \delta')] - 1/2|.$$

The quantity $\mathsf{Adv}^{\mathsf{HIBE}}(t, q_{\mathsf{ID}}, q_{\mathsf{C}})$ denotes the maximum of $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{HIBE}}$ where the maximum is taken over all adversaries running in time at most t and making at most q_{C} queries to the decryption oracle and at most q_{ID} queries to the key-extraction oracle. A HIBE protocol is said to be $(\epsilon, t, q_{\mathsf{ID}}, q_{\mathsf{C}})$-CCA secure if $\mathsf{Adv}^{\mathsf{HIBE}}(t, q_{\mathsf{ID}}, q_{\mathsf{C}}) \leq \epsilon$.

In the above game, we can disallow the adversary $\mathcal{A}$ from querying the decryption oracle. $\mathsf{Adv}^{\mathsf{HIBE}}(t, q)$ in this context denotes the maximum advantage where the maximum is taken over all adversaries running in time at most t and making at most q queries to the key-extraction oracle. A HIBE protocol is said to be (t, q, ϵ)-CPA secure if $\mathsf{Adv}^{\mathsf{HIBE}}(t, q) \leq \epsilon$.

2.3 Cryptographic Bilinear Map

Let G_1 and G_2 be cyclic groups having the same prime order p and $G_1 = \langle P \rangle$, where we write G_1 additively and G_2 multiplicatively. A mapping $e : G_1 \times G_1 \to G_2$ is called a cryptographic bilinear map if it satisfies the following properties.

– Bilinearity : $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in \mathbb{Z}_p$.

– Non-degeneracy : If $G_1 = \langle P \rangle$, then $G_2 = \langle e(P, P) \rangle$.
– Computability : There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Since $e(aP, bP) = e(P, P)^{ab} = e(bP, aP)$, $e()$ also satisfies the symmetry property. The modified Weil pairing [8] and Tate pairing [2, 18] are examples of cryptographic bilinear maps.

Known examples of $e()$ have G_1 to be a group of Elliptic Curve (EC) points and G_2 to be a subgroup of a multiplicative group of a finite field. Hence, in papers on pairing implementations [2, 18], it is customary to write G_1 additively and G_2 multiplicatively. On the other hand, some "pure" protocol papers such as [5, 30] write both G_1 and G_2 multiplicatively though this is not true of the initial protocol papers [8, 19]. Here we follow the first convention as it is closer to the known examples.

2.4 Hardness Assumption

The decisional bilinear Diffie-Hellman (DBDH) problem in $\langle G_1, G_2, e \rangle$ [8] is as follows: Given a tuple $\langle P, aP, bP, cP, Z \rangle$, where $Z \in G_2$, decide whether $Z = e(P, P)^{abc}$ (which we denote as Z is real) or Z is random. The advantage of a probabilistic algorithm $\mathcal{B}$, which takes as input a tuple $\langle P, aP, bP, cP, Z \rangle$ and outputs a bit, in solving the DBDH problem is defined as

$$\mathsf{Adv}_{\mathcal{B}}^{\mathrm{DBDH}} = |\mathsf{Pr}[\mathcal{B}(P, aP, bP, cP, Z) = 1 | \mathsf{Z\ is\ real}] - \mathsf{Pr}[\mathcal{B}(P, aP, bP, cP, Z) = 1 | \mathsf{\ Z\ is\ random}]| \quad (1)$$

where the probability is calculated over the random choices of $a, b, c \in \mathbb{Z}_p$ as well as the random bits used by $\mathcal{B}$. The quantity $\mathsf{Adv}^{\mathrm{DBDH}}(t)$ denotes the maximum of $\mathsf{Adv}_{\mathcal{B}}^{\mathrm{DBDH}}$ where the maximum is taken over all adversaries $\mathcal{B}$ running in time at most t. By the (ϵ, t)-DBDH assumption we mean $\mathsf{Adv}^{\mathrm{DBDH}}(t) \leq \epsilon$.

2.5 Components (AE, KDF, UOWHF)

We briefly introduce and state the security notions for AE, KDF and UOWHF.

An AE protocol consists of two deterministic algorithms – $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$. Both of these use a common secret key k. The $\mathsf{Encrypt}_k$ algorithm takes as input a nonce IV and a message M and returns (C, tag), where C is the ciphertext corresponding to M (and is usually of the same length as M). The $\mathsf{Decrypt}_k$ algorithm takes as input IV and a pair (C, tag) and returns either the message M or $\perp$ (indicating invalid ciphertext).

An AE algorithm possesses two security properties – privacy and authenticity. For privacy, the adversarial game is the following. The adversary $\mathcal{A}$ is given access to an oracle which is either the encryption oracle instantiated with a random key k or is an oracle which simply returns random strings of length equal to its input. After interacting with the oracle the adversary ultimately outputs a bit. The advantage of $\mathcal{A}$ is defined to be

$$|\mathsf{Prob}[\mathcal{A} = 1 | \text{real oracle}] - \mathsf{Prob}[\mathcal{A} = 1 | \text{random oracle}]|.$$

In the above game, the adversary is assumed to be nonce-respecting, in that it does not repeat a nonce. The requirement that IV is a nonce can be replaced by the requirement that IV is chosen randomly. This leads to an additive quadratic degradation in the advantage.

The security notion defined above is that of pseudorandom permutation. This provides the privacy of an AE protocol. In particular, it implies the following notion of one-time security. The adversary submits two equal length messages M_0 and M_1. A random (IV^*, k^*) pair is chosen and a random bit δ is chosen. The adversary is given (C^*, tag^*) which is the encryption of M_δ using IV^* and k^*. The adversary then outputs δ' and its advantage is

$$\left|\mathsf{Prob}[\delta = \delta'] - \frac{1}{2}\right|.$$

We say that an AE protocol satisfies (ϵ, t) one-time encryption security if the maximum advantage of any adversary running in time t in the above game is ϵ.

The authenticity property of an AE protocol is defined through the following game. A nonce respecting adversary $\mathcal{A}$ is given access to an encryption oracle instantiated by a secret key k. It submits messages to the oracle and receives as output ciphertext-tag pairs. Finally, it outputs a "new" ciphertext-tag pair and a nonce, which can be equal to a previous nonce. The advantage of $\mathcal{A}$ in this game is the probability that the forgery is valid, i.e., it will be accepted as a valid ciphertext.

As before, we can replace the requirement that IV be a nonce by the requirement that IV is random without significant loss of security. By an (ϵ, t)-secure authentication of an AE protocol we mean that the maximum advantage of any adversary running in time t in the above game is ϵ.

A KDF is a function $\mathsf{KDF}()$ which takes an input K and produces (IV, dk) as output. The security notion for KDF is the following. For a randomly chosen K, the adversary has to distinguish between $\mathsf{KDF}(K)$ from a randomly chosen (IV, dk).

A function family $\{H_k\}_{k \in \mathcal{K}}$ is said to be a universal one-way hash family if the following adversarial task is difficult. The adversary outputs an x; is then given a randomly chosen $k \in \mathcal{K}$ and has to find $x' \neq x$ such that $H_k(x) = H_k(x')$. We say that the family is (ϵ, t)-secure if the maximum advantage (probability) of an adversary running in time t and winning the above game is ϵ.

3 CCA-Secure HIBE Protocol

In this section, we modify the CPA-secure HIBE protocol in [14] to obtain a CCA-secure HIBE protocol. We provide an explicit hybrid protocol. This allows us to improve the decryption efficiency as we explain later. The modification consists of certain additions to the set-up procedure as well as modifications of the encryption and the decryption algorithms. *No changes are required in the key generation algorithm.*

The additions are based on the technique used by Boyen-Mei-Waters [9] and are also based on the IBE construction by Boneh-Boyen [4] (BB-IBE). Some new ideas – incorporating length of the identity into the ciphertext and using symmetric key authentication to verify ciphertext well formedness – are introduced. Also, an AE protocol is used to combine the two tasks of symmetric key encryption and authentication.

A Useful Notation: Let $v = (v_1, \ldots, v_l)$, where each v_i is an (n/l)-bit string (where l divides n) and is considered to be an element of $\mathbb{Z}_{2^{n/l}}$. For $1 \leq k \leq h$ we define,

$$V_k(v) = U'_k + \sum_{i=1}^{l} v_i U_i. \qquad (2)$$

The modularity introduced by this notation allows an easier understanding of the protocol, since one does not need to bother about the exact value of l. When v is clear from the context we will write V_k instead of $V_k(v)$.

Fig. 1. CCA-secure HIBE.

1. Maximum depth of the HIBE is h.
2. Identities are of the form $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$, $j \in \{1, \ldots, h\}$, $\mathsf{v}_k = (\mathsf{v}_1^{(k)}, \ldots, \mathsf{v}_l^{(k)})$ and $\mathsf{v}_i^{(k)}$ is an (n/l)-bit string.
3. $\langle G_1, G_2, e \rangle$ is as defined in Section 2.3.
4. The notation $V_k()$ is given in (2).
5. The standard way to avoid the computation of $e(P_1, P_2)$ in HIBE.Encrypt is to replace P_2 with $e(P_1, P_2)$ in the public parameters.
6. Key generation is essentially the same as in [30, 14].

<u>HIBE.SetUp</u> 1. Choose α randomly from $\mathbb{Z}_p$. 2. Set $P_1 = \alpha P$. 3. Choose $P_2, U'_1, \ldots, U'_h, U_1, \ldots, U_l$ randomly from G_1. 4. **Choose $\mathbf{W}$ randomly from $\mathbf{G_1}$.** 5. **Let $\mathbf{H_s} : \{\mathbf{1}, \ldots, \mathbf{h}\} \times \mathbf{G_1} \rightarrow \mathbb{Z}_\mathbf{p}$ be chosen from a UOWHF and made public.** 6. Public parameters: $P, P_1, P_2, U'_1, \ldots, U'_h, U_1, \ldots, U_l$ and $\mathbf{W}$. 7. Master secret key: αP_2.	<u>HIBE.KeyGen</u>: Identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$. 1. Choose $r_1, \ldots, r_j$ randomly from $\mathbb{Z}_p$. 2. $d_0 = \alpha P_2 + \sum_{k=1}^{j} r_k V_k(\mathsf{v}_k)$. 3. $d_k = r_k P$ for $k = 1, \ldots, j$. 4. Output $d_\mathsf{v} = (d_0, d_1, \ldots, d_j)$. (Key delegation, i.e., generating d_v from $d_{\mathsf{v}\vert_{j-1}}$ can be done in the standard manner as shown in [30, 14].)
<u>HIBE.Encrypt</u>: Identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$; message M. 1. Choose t randomly from $\mathbb{Z}_p$. 2. $C_1 = tP$, $B_1 = tV_1(\mathsf{v}_1), \ldots, B_j = tV_j(\mathsf{v}_j)$. 3. $K = e(P_1, P_2)^t$. 4. $(\mathsf{IV}, dk) = \mathsf{KDF}(K)$. 5. $(\mathsf{cpr}, \mathsf{tag}) = \mathsf{AE.Encrypt}_{dk}(\mathsf{IV}, M)$. 6. $\gamma = \mathbf{H_s}(\mathbf{j}, \mathbf{C_1})$; $\mathbf{W}_\gamma = \mathbf{W} + \gamma \mathbf{P_1}$; $\mathbf{C_2} = \mathbf{t}\mathbf{W}_\gamma$. 7. Output $(C_1, \mathbf{C_2}, B_1, \ldots, B_j, \mathsf{cpr}, \mathsf{tag})$.	<u>HIBE.Decrypt</u>: Identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$; ciphertext $(C_1, \mathbf{C_2}, B_1, \ldots, B_j, \mathsf{cpr}, \mathsf{tag})$; decryption key $d_\mathsf{v} = (d_0, d_1, \ldots, d_j)$. 1. $\gamma = \mathbf{H_s}(\mathbf{j}, \mathbf{C_1})$; $\mathbf{W}_\gamma = \mathbf{W} + \gamma \mathbf{P_1}$. 2. **If $\mathbf{e}(\mathbf{C_1}, \mathbf{W}_\gamma) \neq \mathbf{e}(\mathbf{P}, \mathbf{C_2})$ return $\perp$.** 3. $K = e(d_0, C_1) / \prod_{k=1}^{j} e(B_k, d_k)$. 4. $(\mathsf{IV}, dk) = \mathsf{KDF}(K)$. 5. $M = \mathsf{AE.Decrypt}_{dk}(\mathsf{IV}, C, \mathsf{tag})$. (This may abort and return $\perp$). 6. Output M.

The description of the construction is given in Figure 1. The bold portions of Figure 1 provide the additional points required over the CPA-secure HIBE construction from [14]. We provide some intuition of how decryption queries are answered. (Key extraction can be answered using the technique from [14] which is built on the work of Waters [30].) First, let us consider what happens if we attempt to simulate decryption queries by key extraction queries. The idea is that we use a key extraction query to derive the private key of the identity which is provided as part of the decryption query. Then this private key is used to decrypt the ciphertext. This idea works fine except for the situation where a decryption query is made on a prefix of the challenge identity. Since, it is not allowed to query the key extraction oracle on prefixes of the challenge identity, the above simulation technique will not work. We need an additional mechanism to answer such decryption queries.

The mechanism that we have used is primarily based on the BMW technique. The parameter W along with P and P_1 define an instance of a BB-IBE protocol. During encryption, an "identity" $\gamma = H_s(j, C_1)$ for this protocol is generated from the randomizer $C_1 = tP$ and the length j of the identity tuple. Using this identity, a separate encapsulation of the key $e(P_1, P_2)^t$ is made. This encapsulation consists of the element C_2 (and C_1). In the security proof, if a decryption query is made on the challenge identity, then this encapsulation is used to obtain the private key of γ and answer the decryption query.

The use of the function $H()$ is different from its use in [9]. In [9], the function $H()$ maps G_1 to $\mathbb{Z}_p$. On the other hand, in the HIBE protocol in Figure 1, $H()$ maps $\{1, \ldots, h\} \times G_1$ to $\mathbb{Z}_p$. Our aim is to include information about the length of the identity into the output of $H()$. Without this information, an encryption for a $(j+1)$-level identity can be converted to an encryption for its j-level prefix by simply dropping the term corresponding to the last component in the identity. (This was pointed out by a reviewer of an earlier version of this work, who, however, did not provide the solution described here.)

The other aspect is that of checking for the well formedness of the ciphertext. A well formed ciphertext requires verifying that $C_1 = tP$, $C_2 = tW_\gamma$ and $B_1 = tV_1(\mathsf{v}_1), \ldots, B_j = tV_j(\mathsf{v}_j)$. In other words, we need to verify the following.

$$\log_P C_1 = \log_{W_\gamma} C_2 \text{ and } \log_P C_1 = \log_{V_1(\mathsf{v}_1)} B_1 = \cdots = \log_{V_j(\mathsf{v}_j)} B_j.$$

In Figure 1, the first equality is explicitly verified, whereas the second equality is not. The idea is that if the second equality does not hold, then the key K that will be reconstructed will be improper and indistinguishable from random (to the adversary). Correspondingly, the quantities (IV, dk) will also be indistinguishable from random and symmetric authentication with this pair will fail (otherwise the adversary has broken the authentication of the AE protocol). Thus, instead of using j pairings for verifying the second equality, we use symmetric authentication to reject invalid ciphertext. This leads to a more efficient decryption algorithm. Note that the use of hybrid encryption is very crucial in the current context. This is similar to the Kurosawa-Desmedt PKE, which provides improved efficiency over the Cramer-Shoup protocol for hybrid encryption.

The additional requirements of group elements and operations for attaining CCA-security compared to the protocol in [14] consists of the following.

1. One extra group element in the public parameters.
2. Two additional scalar multiplications during encryption.
3. One additional scalar multiplication and one pairing based verification during decryption.

3.1 Security Statement

The security statement for the new protocol is given below.

Theorem 1. *The HIBE protocol described in Figure 1 is $(\epsilon_{hibe}, t, q_{\mathsf{ID}}, q_C)$-CCA secure assuming that the (t', ϵ_{dbdh})-DBDH assumption holds in $\langle G_1, G_2, e\rangle$; H_s is an (ϵ_{uowhf}, t)-UOWHF;* KDF *is (ϵ_{kdf}, t)-secure; and the AE protocol possesses (ϵ_{auth}, t)-authorization security and (ϵ_{enc}, t) one-time encryption security; where*

$$\epsilon_{hibe} \leq 2\epsilon_{uowhf} + \frac{\epsilon_{dbdh}}{\lambda} + 4\epsilon_{kdf} + 2\epsilon_{enc} + 2hq_C\epsilon_{auth}. \tag{3}$$

where $t' = t + O(\tau q) + \chi(\epsilon_{hibe})$ and

$\chi(\epsilon) = O(\tau q + O(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1})))$;
τ is the time required for one scalar multiplication in G_1;
$\lambda = 1/(2h(2\sigma(\mu_l + 1))^h)$ *with* $\mu_l = l(2^{n/l} - 1)$, $\sigma = \max(2q, 2^{n/l})$ *and* $q = q_{\mathsf{ID}} + q_C$.

We further assume $2\sigma(1 + \mu_l) < p$.

The proof of the Theorem is given in Section A. The statement of Theorem 1 is almost the same as that of Theorem 1 in [14] with the following differences.

1. The above theorem states CCA-security where as [14] proves CPA-security.
2. The value of λ is equal to $1/(2h(2\sigma(\mu_l + 1))^h)$ in the above statement where as it is equal to $1/(2(2\sigma(\mu_l + 1))^h)$ in [14], i.e., there is an additional degradation by a factor of h.
3. The value of q in the expression for σ is the sum of q_{ID} and q_C whereas in [14] it is only q_{ID}. The reason for having q_C as part of q is that it may be required to simulate decryption queries using key extraction queries.

For $2q \geq 2^{n/l}$ (typically l would be chosen to ensure this), we have

$$\epsilon_{hibe} \leq 2\epsilon_{uowhf} + 2h(4lq2^{n/l})^h\epsilon_{dbdh} + 4\epsilon_{kdf} + 2\epsilon_{enc} + 2hq_C\epsilon_{auth}.$$

The corresponding bound on ϵ_{dbdh} in [14] is $2(4lq_{\mathsf{ID}}2^{n/l})^h\epsilon_{dbdh}$. Thus, we get an additional security degradation of ϵ_{dbdh} by a factor of h while attaining CCA-security. Since h is the maximum number of levels in the HIBE, its value is small and the degradation is not significant. Also, q in the present case includes both key extraction and decryption queries.

The statement of Theorem 1 is a little complicated. The complexity is inherited from the corresponding security statement in [14]. These arise from the requirement of tackling key extraction queries and providing challenge ciphertexts. In particular, λ is a lower bound on the probability of not abort by the simulator and $O(\epsilon^{-2}\ln((\epsilon^{-1}\lambda^{-1}\ln(\lambda^{-1})))$ is the extra runtime introduced due to the artificial abort requirement. In [14], the security degradation is worked out in more details and much of these also hold for Theorem 1. Hence, we do not repeat the analysis in this paper.

Techniques for showing security against chosen plaintext attacks is from [14] and is based on the works of Waters [30] and Boneh-Boyen [4]. Since these details are already given in [14], we do not repeat them in the proof of Theorem 1. The proof technique for answering decryption queries is based on the work of Boyen-Mei-Waters [9]. Also relevant is the work of Kiltz-Galindo [24]. The basic idea of using symmetric authentication to verify ciphertext well formedness is taken from the paper by Kurosawa-Desmedt [25]. A proof of the KD protocol using the so called method of "deferred analysis" is given in [1]. This proof is in the PKE setting which we had to adapt to fit the (H)IBE framework. The proof of Theorem 1 is given in Section A.

3.2 Comparison to Previous Work

The construction in Figure 1 can be specialized to obtain CCA-secure PKE and IBE as special cases. We show that when specialized to PKE, the protocol in Figure 1 simplifies to yield the BMW construction. On the other hand, when specialized to IBE, we obtain a more efficient (actually the decryption algorithm is more efficient) IBE protocol compared to the previously best known construction of Kiltz-Galindo [24].

Public Key Encryption. In this case there are no identities and no PKG. It is possible to make the following simplifications.

SetUp:

1. The elements $U'_1, \ldots, U'_l, U_1, \ldots, U_l$ are no longer required.
2. The UOWHF H_s can be replaced by an injective embedding from G_1 to $\mathbb{Z}_p$.
3. A random w in $\mathbb{Z}_p$ is chosen and W is set to be equal to wP.
4. The secret key is set to be equal to $(\alpha P_2, \alpha, w)$.
5. The AE protocol can be replaced with a one-time secure data encapsulation mechanism (DEM).

KeyGen: This is not required at all.

Encrypt:

1. The elements $B_1, \ldots, B_j$ are not required.
2. Encryption with a DEM will not produce a tag.

Decrypt:

1. The purpose of the pairing verification $e(C_1, W_\gamma) = e(P, C_2)$ is to ensure that $C_1 = tP$ and $C_2 = tW_\gamma$, where $W_\gamma = W + \gamma P_1$. With the knowledge of w and α, this can be done as follows. Compute $w' = w + \gamma\alpha$ and verify whether $w'C_1 = C_2$. This requires only one scalar multiplication as opposed to one pairing verification.
2. The value of K is reconstructed as $K = e(C_1, \alpha P_2)$.
3. Since the AE protocol is replaced with a DEM, symmetric authentication will not be done.

With these simplifications, the protocol becomes the BMW protocol.

Identity Based Encryption. In this case $h = 1$. The protocol in Figure 1 remains unchanged except for one simplification. In a HIBE, the length of the identity tuple can vary from 1 to h. For an IBE, the length is always one. Hence, in this case, we can restrict the domain of H_s to be G_1. Since, G_1 has cardinality p, the domain and range of H_s are the same and we can also take H_s to be an injective embedding from G_1 to $\mathbb{Z}_p$ as has been done in the BMW construction.

Let us now compare the resulting IBE construction with the previous construction of Kiltz-Galindo [24]. In both cases, the public key portion of the ciphertext is of the form (C_1, C_2, B_1). During decryption, KG protocol verifies that $C_1 = tP$, $C_2 = tW_\gamma$ and $B_1 = tV_1(\mathsf{v}_1)$. This requires two pairing based verifications of the type $e(P, C_2) = e(C_1, W_\gamma)$ and $e(P, B_1) = e(C_1, V_1(\mathsf{v}_1))$. The cost of one such verification is less than the cost of two pairing operations. By $[VP]$ we denote the cost of one such verification. Also, let $[P]$, $[SM]$, $[m]$ and $[i]$ respectively denote the costs of one pairing operation, one scalar multiplication in G_1, one multiplication in G_2 and one inversion in G_1. The total cost of decryption in the KG protocol with the pairing based verification technique is $1[SM] + 2[VP] + 2[P] + 1[i] + 1[m]$.

KG suggests a method of implicit rejection, whereby this cost can be reduced. In this case, the pairing based verifications are not done; instead two random elements r_1 and r_2 are chosen and K is computed as

$$\frac{e(C_1, d_0 + r_1 W_\gamma + r_2 V_1(\mathsf{v}_1))}{e(B_1, d_1 + r_2 P)e(r_1 P, C_2)}.$$

If the ciphertext is proper, then the proper K is computed, while if the ciphertext is improper, then a random K is computed. *Note that implicit rejection cannot explicitly detect an invalid ciphertext.* If the KG-KEM with implicit rejection is composed with a one-time secure DEM, then invalid ciphertexts will *not* be rejected. Thus, implicit rejection does not seem to provide the full functionality of a CCA-secure KEM. The cost of decryption with implicit rejection is $5[SM] + 3[P] + 1[i] + 1[m]$.

In contrast, the cost of verification in our case is $1[SM] + 1[VP] + 2[P] + 1[i] + 1[m]$. This is significantly lower than the KG protocol with explicit pairing based verification. Compared to the implicit rejection technique, our cost will be lower when $1[VP] < 1[P] + 4[SM]$. Based on the current status of efficient pairing based algorithms, this seems to be a reasonable condition.

The reason for obtaining this lower cost is that we do not verify $e(P, B_1) = e(C_1, V_1(\mathsf{v}_1))$ either explicitly or implicitly. In other words, we do not verify whether $\log_P C_1 = \log_{V_1(\mathsf{v}_1)} B_1$. If this does not hold, then an incorrect session key will be generated and ultimately the authentication of the AE protocol will fail. In a sense, this is also an implicit verification, but the verification is done using the symmetric component which reduces the total cost of decryption. Also, an invalid ciphertext will always be rejected.

In summary, the IBE version of the protocol in Figure 1 is the currently known most efficient CCA-secure IBE protocol in the full model without the random oracle heuristic and based on the DBDH assumption.

Hierarchical Identity Based Encryption. Based on the work by BMW [9], the KG paper [24] sketches a construction of a HIBE. The details are worked out in [3]. Compared to this approach, there are several advantages of our protocol. First, the ciphertext verification procedure in this approach requires the verification of $\log_P C_1 = \log_{V_1(\mathsf{v}_1)} B_1 = \cdots = \log_{V_1(\mathsf{v}_1)} B_1$ either explicitly using pairing based verifications or implicitly as suggested by Kiltz-Galindo. On the other hand, our approach does not require these verifications. If any of these equalities do not hold, then an improper value of K will be obtained and as a result the authentication of the AE protocol will fail. This significantly reduces the cost of the decryption algorithm. Second, we use an AE algorithm to perform simultaneous encryption and authentication which can be twice as fast as separate encryption and authentication.

An earlier work [11, 7] showed a generic construction for converting an $(h+1)$-level CPA-secure HIBE into an h-level CCA-secure HIBE. The construction used one-time signatures, which make it quite inefficient. It was suggested (without details) in [11] that a MAC based construction can be used to remove the inefficiency of the one-time signature based approach. Also, the efficiency of the resulting protocol is less than that of Figure 1. Another problem with the approach in [11] is that the identity components of the CCA-secure HIBE are prepended with a bit to obtain identity components of the underlying CPA-secure HIBE. This can cause difficulties in implementation.

The currently known techniques (both generic and non-generic) for converting a CPA-secure HIBE protocol to a CCA-secure HIBE protocol, starts with an $(h+1)$-level CPA-secure HIBE and then converts it to an h-level CCA-secure HIBE. The security degradation thus correspond to the

$(h+1)$-level HIBE. If we apply this technique to the protocol in [14], then the security degradation for the obtained h-level CCA-secure HIBE will be $2(4lq2^{n/l})^{h+1}$. Compared to this, the security degradation given by Theorem 1 is $2h(4lq2^{n/l})^{h}$. In other words, we have managed to reduce the exponent from $(h+1)$ to h and have introduced a multiplicative factor of h. From the viewpoint of concrete security analysis, a typical value of q is 2^{30}. Assuming this value of q, we are able to prevent approximately a 30-bit security degradation compared to previous work.

4 Conclusion

In this paper, we have provided a construction of a hybrid HIBE protocol. The protocol is secure against adaptive adversaries (making both key extraction and decryption queries) without using the random oracle hypothesis. Security is reduced from the computational hardness of the DBDH problem. To the best of our knowledge, in this setting, the HIBE protocol described in this paper is the currently known most efficient construction for the setting described in this paper.

References

1. 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 Cramer [16], pages 128–146.
2. Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002.
3. James Birkett, Alexander W. Dent, Gregory Neven, and Jacob Schuldt. Identity based key encapsulation with wildcards. Cryptology ePrint Archive, Report 2006/377, 2006. http://eprint.iacr.org/.
4. Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Cachin and Camenisch [10], pages 223–238.
5. Dan Boneh and Xavier Boyen. Secure Identity Based Encryption Without Random Oracles. In Franklin [17], pages 443–459.
6. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Cramer [16], pages 440–456. Full version available at Cryptology ePrint Archive; Report 2005/015.
7. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *SIAM J. of Computing*, 36(5):915–942, 2006.
8. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. Earlier version appeared in the proceedings of CRYPTO 2001.
9. Xavier Boyen, Qixiang Mei, and Brent Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 320–329. ACM, 2005.
10. Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.
11. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In Cachin and Camenisch [10], pages 207–222.
12. Debrup Chakraborty and Palash Sarkar. A General Construction of Tweakable Block Ciphers and Different Modes of Operations. In Helger Lipmaa, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 4318 of *Lecture Notes in Computer Science*, pages 88–102. Springer, 2006.
13. Sanjit Chatterjee and Palash Sarkar. Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model. In Dong Ho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 424–440. Springer, 2005.
14. Sanjit Chatterjee and Palash Sarkar. HIBE with Short Public Parameters Without Random Oracle. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 145–160. Springer, 2006. see also Cryptology ePrint Archive, Report 2006/279, http://eprint.iacr.org/.

15. Sanjit Chatterjee and Palash Sarkar. New Constructions of Constant Size Ciphertext HIBE Without Random Oracle. In M.S. Rhee and B. Lee, editors, *ICISC*, volume 4296 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2006.
16. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.
17. Matthew K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology-Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004.
18. Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the Tate Pairing. In Claus Fieker and David R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.
19. Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.
20. Virgil D. Gligor and Pompiliu Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2001.
21. Jeremy Horwitz and Ben Lynn. Toward Hierarchical Identity-Based Encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.
22. Charanjit S. Jutla. Encryption Modes with Almost Free Message Integrity. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, 2001.
23. Eike Kiltz. Chosen-ciphertext secure identity-based encryption in the standard model with short ciphertexts. Cryptology ePrint Archive, Report 2006/122, 2006. http://eprint.iacr.org/.
24. Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2006. full version available at http://eprint.iacr.org/2006/034.
25. Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In Franklin [17], pages 426–442.
26. David Naccache. Secure and Practical Identity-Based Encryption. Cryptology ePrint Archive, Report 2005/369, 2005. http://eprint.iacr.org/.
27. Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.
28. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.
29. Victor Shoup. Sequences of Games: a Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/.
30. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [16], pages 114–127.

Appendix

A Proof of Theorem 1

The construction of CCA-secure HIBE in Figure 1 is built on the construction of CPA-secure HIBE given in [14]. The proof in [14] shows how to set-up the protocol, answer key-extraction queries and generate the challenge ciphertext. The proof of Theorem 1 incorporates these aspects of the proof in [14]. Additionally, we have the following considerations.

1. Definition of W during set-up.
2. Generation of C_2 during challenge generation as well as generation of a proper ciphertext using the AE protocol.
3. Properly answering decryption queries.

The proof of Theorem 1 is given as a sequence of games. In each game a bit δ is chosen randomly and the adversary makes a guess δ'. By X_i we denote the event that $\delta = \delta'$ in the ith game.

Game 0: This is the usual adversarial game for defining CCA-security of HIBE protocols. We assume that the adversary's runtime is t, it makes q_{ID} key-extraction queries and q_{C} decryption queries. Also, we assume that the adversary maximizes the advantage among all adversaries with similar resources. Thus, we have

$$\epsilon_{hibe} = \left|\mathsf{Pr}[X_0] - \frac{1}{2}\right|.$$

The group element C_1^* provided to the adversary during the challenge generation does not depend on the adversary's input. We will assume that this is randomly chosen during setup. Also, we will assume that during set-up an integer j_θ is chosen uniformly at random from $\{1, \ldots, h\}$. The significance of j_θ will become clear later. We will denote the quantities corresponding to the challenge by a superscript $*$.

Game 1: This is the same as Game 0, with the following change. If the adversary ever submits a decryption query of the form $(C_1, C_2, B_1, \ldots, B_j)$ with $(j, C_1) \neq (j_\theta, C_1^*)$ and $H(j, C_1) = H(j_\theta, C_1^*)$, then the simulator rejects the query. Let F_1 be the event that a decryption query is rejected only by this check. It is easy to see that $\mathsf{Pr}[F_1] \leq \epsilon_{uowhf}$. If F_1 does not occur, then Game 0 and Game 1 are identical. Using the difference lemma (as named in [29]), we obtain

$$|\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1]| \leq \mathsf{Pr}[F_1] \leq \epsilon_{uowhf}.$$

Game 2: This game is the main non-trivial game of the proof. The protocol is setup from a tuple $(P, P_1 = aP, P_2 = bP, P_3 = cP, Z = e(P, P)^{abc})$, where we assume that a, b and c are known to the simulator. There are four parts to this game – setup; simulation of key-extraction queries; simulation of decryption queries; and challenge generation.

For certain queries as well as for certain challenge identities, the simulator is unable to answer without using the values of a, b or c. In such cases, it sets a flag flg to 1 (which is initially set to 0). However, it always answers the adversary's queries properly and hence the adversary's view remains unchanged from the previous game. Thus, we have $\mathsf{Pr}[X_1] = \mathsf{Pr}[X_2]$.

Set-Up: Set $P_1 = aP$ and $P_2 = bP$. The secret key is $bP_2 = abP$. Also, set $C_1^* = cP$ and j_θ is chosen during set-up as mentioned in Game 0.

The public parameters $(U'_1, \ldots, U'_h, U_1, \ldots, U_l)$ are required to handle key extraction queries. The proper construction of these parameters are given in [14].

The parameter W is required for answering decryption queries (and is not present in [14]). We show how to define W. Compute $\gamma = H(j_\theta, P_3)$; choose β randomly from $\mathbb{Z}_p$ and define $W = -\gamma P_1 + \beta P$. The choice of j_θ corresponds to the fact that at this point we are guessing the length of the challenge identity.

Key Extraction Query: The technique for answering such queries is described in details in [14]. Here we only note that answering certain queries require the use of the values a or b. In all such cases, the simulator sets flg to one.

Decryption Query: Suppose $C = (C_1, C_2, B_1, \ldots, B_j)$ is a decryption query for the identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$. There are several cases to consider.

Case $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$ *is not a prefix of* $(\mathsf{v}_1^*, \ldots, \mathsf{v}_{j_\theta}^*)$*:* In this case, a private key d_v for v is obtained using the technique for simulating key extraction query. This d_v is used to decrypt the ciphertext. In the process of key extraction, the variable flg might have to be set to one.

Case $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$ *is a prefix of* $(\mathsf{v}_1^*, \ldots, \mathsf{v}_{j_\theta}^*)$*:* If either $j < j_\theta$ or $C_1 \neq C_1^*$, then by Game 1, we can assume that $H(j, C_1) \neq H(j_\theta, C_1^*)$. So suppose that $(j, C_1) = (j_\theta, C_1^*)$. We assume that this happens only in Phase 2. (In Phase 1, the randomly chosen C_1^* is not available to the adversary and hence the event $C_1 = C_1^*$ can occur only with negligible probability of q/p.) Using $j = j_\theta$, we have $(\mathsf{v}_1, \ldots, \mathsf{v}_j) = (\mathsf{v}_1^*, \ldots, \mathsf{v}_{j_\theta}^*)$. This and $C_1 = C_1^*$ implies that $(B_1, \ldots, B_j) = (B_1^*, \ldots, B_j^*)$. Otherwise the query is invalid and will be rejected. Also, using $(j, C_1) = (j_\theta, C_1^*)$ it is easy to verify that $C_2 = C_2^*$. Thus, we have $(C_1, C_2, B_1, \ldots, B_j) = (C_1^*, C_2^*, B_1^*, \ldots, B_j^*)$. In other words, the decryption query is on the challenge ciphertext, which is not allowed in the game. Hence, we cannot have $(j, C_1) = (j_\theta, C_1^*)$ and so $H(j, C_1) \neq H(j_\theta, C_1^*)$.

Let $\gamma' = H(j, C_1)$ and $W_{\gamma'} = W + \gamma' P_1$. The simulator verifies whether $e(C_1, W'_\gamma) = e(P, C_2)$ proceeds if the test succeeds. If the test fails, it returns $\perp$ to $\mathcal{A}$. Note that, at this point, since we have verified that $e(C_1, W'_\gamma) = e(P, C_2)$, we can write $C_1 = tP$ and $C_2 = tW_{\gamma'}$ for some t in $\mathbb{Z}_p$.

Choose r randomly from $\mathbb{Z}_p$ and compute $E_{\gamma'}$ and $d_{\gamma'}$ in the following manner. Recall that $\gamma = H(j_\theta, P_3) = H(j_\theta, cP)$ and $W = -\gamma P_1 + \beta P$. Since, $\gamma' = H(j, C_1) \neq H(j_\theta, C_1^*) = \gamma$, the inverse of $(\gamma' - \gamma)$ exists.

$$\left.\begin{array}{rl} E_{\gamma'} &= \frac{-\beta}{\gamma'-\gamma} P_2 + r((\gamma' - \gamma)P_1 + \beta P) \\ &= aP_2 + \left(r - \frac{b}{\gamma'-\gamma}\right)(\gamma' P_1 + W) \\ &= aP_2 + \tilde{r} W_{\gamma'} \\ d_{\gamma'} &= rP - \frac{1}{\gamma'-\gamma} P_2 \\ &= \tilde{r} P. \end{array}\right\} \tag{4}$$

This technique is essentially based on [9] which is in turn based on the technique of [4]. The verification of the above computation is quite routine – in particular the second equality can be easily seen by substituting $W = -\gamma P_1 + \beta P$ and noting that $P_2 = bP$.

The decryption can now be performed as follows.

$$\frac{e(E_{\gamma'}, C_1)}{e(d_{\gamma'}, C_2)} = \frac{e(aP_2 + \tilde{r} W_{\gamma'}, tP)}{e(\tilde{r} P, tW_{\gamma'})} = e(P_1, P_2)^t.$$

Note that any such decryption query can be answered without using the values of a, b or c. Thus, flg is never set to 1 during this step.

The simulation of the decryption query makes the role of W clear. The protocol uses $K = e(P_1, P_2)^t$ to be the secret key and creates two encapsulations of it. The first encapsulation is using the HIBE protocol of [14], where as the second encapsulation is using the BB-IBE protocol from [4]. In the actual protocol, the second encapsulation is never used (apart from verifying its correctness). It is used in the simulation to obtain K and answer a decryption query if the identity of the decryption query is a prefix of the challenge identity. The advantage is that the BB-IBE protocol is only required to be selective-ID secure and hence the "challenge identity" γ for the BB-IBE protocol can be generated during set-up.

Challenge: The adversary submits equal length messages M_0 and M_1 and a challenge identity $(\mathsf{v}_1^*, \ldots, \mathsf{v}_{j^*}^*)$. The challenge ciphertext is of the form $(C_1^*, C_2^*, B_1^*, \ldots, B_j^*)$, where we have already chosen $C_1^* = cP$ during set-up. The components B_1^* to B_j^* are generated as in [14].

The component C_2^* is new to this protocol and we show how to generate it. If $j^* \neq j_\theta$, then set flg to 1, i.e., the random guess of the length of the challenge identity during the set-up turns out to be incorrect. In this case, the simulator uses a, b and c to generate the challenge and answer the adversary. Otherwise, set $C_2 = \beta P_3$. This C_2 is properly formed since $C_2 = cW_{\gamma'} = c(\gamma' P_1 + W) = c(\gamma' P_1 - \gamma P_1 + \beta P) = c\beta P = \beta P_3$. We use $\gamma' = H(j_\theta, C_1) = H(j^*, cP) = \gamma$.

Choose a random bit δ. Set $K^* = Z$ and then apply the rest of the encryption procedure to complete the encryption for the message M_δ.

Game 3: This game is the same as Game 2, with the only difference that the Z in Game 2 is now replaced by a random element of G_2. The difference in the two games can be used to obtain an algorithm to solve DBDH problem. The basic idea is the following.

Suppose we are given a tuple (P, aP, bP, cP, Z) where Z is either $e(P,P)^{abc}$ or Z is random. The algorithm for solving DBDH can be described as follows. We play an adversarial game based on the given tuple as described above. If $Z = e(P,P)^{abc}$, then we are playing Game 2 and if Z is random, then we are playing Game 3. The problem is that in certain cases in these two games, we need to use the values of a, b or c, which are of course not known to us when we are trying to solve the DBDH problem. In all such cases, flg is set to one. If flg is set to one, then the algorithm to solve DBDH aborts and outputs a random bit. Details of how to obtain a DBDH solver from the two games are given in [14]. Also, a detailed analysis of the probability that flg remains 0 throughout the game has been carried out in [14]. The technique and the proof is based on the work of Waters and Boneh-Boyen. This analysis also hold for the current proof. The only new abort condition is during challenge generation, when $j^* \neq j_\theta$. Since $1 \leq j^*, j_\theta \leq h$ and j_θ is chosen randomly from $\{1, \ldots, h\}$, the probability of this new abort is $1/h$.

With this small change, the analysis of [14] shows that

$$|\mathsf{Pr}[X_2] - \mathsf{Pr}[X_3]| \leq \frac{\epsilon_{dbdh}}{2\lambda} + \frac{\epsilon_{hibe}}{2} \tag{5}$$

where $\lambda = 1/(2h(2\sigma(\mu_l + 1))^h)$. The factor $1/h$ in this expression is due to the new kind of abort.

Game 4: At this point, we have K^* to be random. Since we are assuming that a, b and c are known to the simulator, we can also assume that u'_j and u_i are known to the simulator such that $U'_j = u'_j P$ and $U_i = u_i P$. (This follows easily from the definition of U_i and U'_j given in [14].) This does not disturb adversary's view of the game. On the other hand, with this knowledge, we can assume that for any V_i, the simulator is able to compute w_i such that $V_i = w_i P$. The adversary may submit a decryption query with $C_1 = tP$ and for some i, $B_i = t_1 V_i$ with $t \neq t_1$. The knowledge of w_i allows the simulator to test for this in the following manner: If $e(C_1, V_i) \neq e(w_i C_1, P)$, then $t_1 \neq t$ and the query is malformed. The simulator can now detect and reject such a query. Note that this checking is not done in the actual protocol. So, we would like to be assured that the chance of getting to this checking stage is small. In other words, we would like to be assured that if the query is malformed as above and the protocol does not reject it, then the adversary has broken the authentication property of the AE protocol.

Let **Rejection Rule 0** be the rule whereby a ciphertext is rejected based on the failure of the authentication property of the AE protocol. Let **Rejection Rule 1** be the rejection rule mentioned above. Let F_4 be the event that a malformed query is rejected by **Rule 1** but not by **Rule 0**. Our aim is to show that the chance of this happening is low. Note that if no query is rejected by **Rule 1**, then Games 3 and 4 are identical.

From this point onwards, we will only be considering decryption queries. The adversary makes a total of q_C decryption queries. We will use the superscript $(\jmath)$ to denote the quantities related to the $\jmath$th decryption query. For example, $K^{(\jmath)}$ denotes the input to $\mathsf{KDF}()$ in the $\jmath$th decryption query.

We now employ a "plug and pray" technique used in [1] and assume that the $\imath$th component of the $\jmath$th query is malformed, i.e., $C_1^{(\jmath)} = tP$ and $B_\imath^{(\jmath)} = t_1P$ with $t \neq t_1$. Note that the "plug and pray" here also extends over the levels of the HIBE, a feature which is not required in [1]. Let F_4' be the event that the query is not rejected by **Rule 0** but the $\imath$th component of the $\jmath$th query fails **Rule 1**. Then $\mathsf{Pr}[F_4] \leq h \times q_C \times \Pr[F_4']$ and we have

$$|\mathsf{Pr}[X_3] - \mathsf{Pr}[X_4]| \leq \mathsf{Pr}[F_4] \leq h \times q_{\mathsf{C}} \times \mathsf{Pr}[F_4']. \tag{6}$$

We would like to upper bound $\mathsf{Pr}[F_4']$. For this we use the deferred analysis technique of [1]. Also, since we have done a "plug and pray" over the levels of the HIBE, henceforth we will assume that there is only one level in the HIBE, i.e., we are considering an IBE protocol. This will simplify the notation as this will result in only one B which is of the form tV.

Game 5: We modify Game 4 in the following manner. If the $\jmath$th decryption query is detected to be malformed using **Rule 1**, then we set $K^{(\jmath)}$ to be a random element of G_2. We now have to argue that this does not change the adversary's point of view. In effect, we are setting both K^* and $K^{(\jmath)}$ to be independent random elements and have to argue that this is what the adversary can expect to see.

A similar argument is also required in [1]. This is done by initially having some extra randomness in the setup and later adjusting the setup parameters such that these randomness can be transferred to the challenge ciphertext and the malformed query. The situation in the identity based setting is different. In the identity based setting, the adversary can ask for the private key corresponding to an identity; such a thing is not possible in the public key encryption setting. On the other hand, the on line probabilistic generation of the secret key for an identity allows an extra source of randomness.

Let us now analyze the relationship between the identity v^* for the challenge ciphertext and the identity $\mathsf{v}^{(\jmath)}$ for the malformed query. There are two cases to consider.

Case $\mathsf{v}^* = \mathsf{v}^{(\jmath)}$*:* In this case, the adversary cannot ask for the private key of $\mathsf{v}^{(\jmath)}$. Let the secret key corresponding to $\mathsf{v}^{(\jmath)}$ be $(aP_2 + rV^{(\jmath)}, rP)$, where r is a random element of $\mathbb{Z}_p$. Then the adversary expects $K^{(\jmath)}$ of the malformed query to be

$$K^{(\jmath)} = \frac{e(aP_2 + rV^{(\jmath)}, tP)}{e(t_1V^{(\jmath)}, rP)} = e(P_1, P_2)^t \times e(P, P)^{rw(t-t_1)}.$$

Since $t \neq t_1$ (as the query is malformed) and r is random, $K^{(\jmath)}$ is also random. On the other hand, the adversary expects K^* to be $e(P_1, P_2)^{t^*}$ where t^* is random. Hence, the adversary expects K^* to be random. Further, the randomness of $K^{(\jmath)}$ and K^* depends on the randomness of r and t^* which are independent. Hence, the adversary also expects $K^{(\jmath)}$ and K^* to be independent random quantities as provided to the adversary.

Case $\mathsf{v}^* \neq \mathsf{v}^{(j)}$*:* In this case, the adversary can ask for the secret key for $\mathsf{v}^{(j)}$ but not before making the malformed decryption query. If the adversary knows the secret key for $\mathsf{v}^{(j)}$, then he can decrypt any ciphertext encrypted using $\mathsf{v}^{(j)}$. Thus, it is useless for him to query the decryption oracle using $\mathsf{v}^{(j)}$ *after* obtaining the secret key for $\mathsf{v}^{(j)}$. Recall that we had disallowed such useless queries.

The adversary can first ask for the decryption of a malformed query and then ask for the private key for the same identity. We have to ensure that the answers to the decryption and private key queries are consistent. (This situation does not arise in public key encryption scheme.) By consistency we mean the following. Suppose the adversary makes a decryption query with $\mathsf{v}^{(j)}$ and later a private key extraction query on $\mathsf{v}^{(j)}$. With the private key $d_{\mathsf{v}^{(j)}}$ returned to him, the adversary can decrypt his own earlier decryption query. Consistency requires that the output given to him on his decryption query should be equal to what he computes for himself. The next modification ensures this consistency. Note that in this case, we do not have to bother about the independence of K^* and $K^{(j)}$, since this will be easily ensured.

Let the jth query be of the form $(t^{(j)}P, t_1^{(j)}V)$. Suppose the simulator returns $K^{(j)} = e(P_1, P_2)^{t_2^{(j)}}$. On a later private key query on $\mathsf{v}^{(j)}$, the simulator has to return $(aP_2 + r^{(j)}V, r^{(j)}P)$ for some random $r^{(j)} \in \mathbb{Z}_p$. The consistency requirement is satisfied if

$$K^{(j)} = \frac{e(aP_2 + r^{(j)}V, t^{(j)}P)}{e(t_1^{(j)}V, r^{(j)}P)}.$$

As mentioned before, the simulator can compute a w such that $V = wP$ for some $w \in \mathbb{Z}_p$. Also $P_1 = aP$ and $P_2 = bP$, where we assume at this point that the quantities a and b are known to the simulator. The above consistency condition can be written as

$$t_2^{(j)} = t^{(j)} + \frac{wr^{(j)}(t^{(j)} - t_1^{(j)})}{ab}.$$

Note that the simulator does not know $t^{(j)}$ and $t_1^{(j)}$.

The jth malformed query is answered in the following manner. The simulator chooses an $r^{(j)}$ (required for answering a possible future key extraction query on $\mathsf{v}^{(j)}$) randomly. It then computes $A = e(P, abt^{(j)}P) = e(P, P)^{abt^{(j)}}$. This can be done since the simulator knows a, b, P and $t^{(j)}P$. It then computes

$$B = \frac{e(t^{(j)}P, r^{(j)}V^{(j)})}{e(t_1^{(j)}V^{(j)}, r^{(j)}P)} = e(P, P)^{r^{(j)}w(t^{(j)} - t_1^{(j)})}.$$

Note that both numerator and denominator is computable from what is known to the simulator. Then the simulator computes

$$K^{(j)} = (A \times B)^{1/(ab)} = e(P_1, P_2)^{t_2^{(j)}}.$$

This value $K^{(j)}$ is returned to the adversary. Since $r^{(j)}$ is random, so is $t_2^{(j)}$ and hence $K^{(j)}$ is random. Later if the adversary asks for the private key for $\mathsf{v}^{(j)}$, then the simulator uses $r^{(j)}$ to construct the private key and answer the adversary.

Define F_5' in a manner similar to F_4'. Then we have

$$\mathsf{Pr}[X_4] = \mathsf{Pr}[X_5] \text{ and } \mathsf{Pr}[F_4'] = \mathsf{Pr}[F_5']. \tag{7}$$

Game 6: This is obtained from Game 5 by the following modification. In Game 5, the keys $(\mathsf{IV},^{*} dk^{*})$ and $(\mathsf{IV}^{(\jmath)}, dk^{(\jmath)})$ are obtained by applying KDF to K^* and $K^{(\jmath)}$ respectively. In Game 6, these are generated randomly. Define F'_6 in a manner similar to that of F'_4. Then we have

$$|\mathsf{Pr}[X_5] - \mathsf{Pr}[X_6]| \leq 2\epsilon_{kdf} \text{ and } |\mathsf{Pr}[F'_5] = \mathsf{Pr}[F'_6]|. \tag{8}$$

The factor of two comes due to the fact that the adversary can break one out of these two invocations of KDF.

In Game 6, the secret K^* (and also (IV, dk)) is random and independent of the elements $C_1^*, C_2^*, B_1^*, \ldots, B_{j^*}^*$. The message M_δ is encrypted using the random (IV^*, dk^*). If the adversary is able to correctly guess δ, then the one-time security of the AE protocol is broken. Hence, $|\mathsf{Pr}[X_6] - 1/2| \leq \epsilon_{enc}$.

Now, we turn to bounding the probability of the event F'_6. Recall that the occurrence of the event F'_6 implies that the query has passed the authentication of the underlying AE protocol. At this point, we have $K^\jmath$ to be random and hence using the security of KDF $(\mathsf{IV}^\jmath, dk^\jmath)$ is also random (and unknown to the adversary). Thus, the adversary has been able to obtain a forgery for the AE protocol under a random unknown key. This violates the authentication property of the AE protocol and hence $\mathsf{Pr}[F'_6] \leq \epsilon_{auth}$. Finally, combining all the inequalities, we obtain

$$\begin{aligned}
\epsilon_{hibe} &= \left|\mathsf{Pr}[X_0] - \frac{1}{2}\right| \\
&\leq |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1]| + |\mathsf{Pr}[X_2] - \mathsf{Pr}[X_3]| + |\mathsf{Pr}[X_3] - \mathsf{Pr}[X_4]| \\
&\quad + |\mathsf{Pr}[X_5] - \mathsf{Pr}[X_6]| + |\mathsf{Pr}[X_6] - 1/2| \\
&\leq \epsilon_{uowhf} + \frac{\epsilon_{dbdh}}{2\lambda} + \frac{\epsilon_{hibe}}{2} + hq_{\mathsf{C}}\epsilon_{auth} + 2\epsilon_{kdf} + \epsilon_{enc}.
\end{aligned}$$

Rearranging the inequality gives the desired relationship. □