# Transforming a CPA-Secure HIBE Protocol into a CCA-Secure HIBE Protocol Without Loss of Security


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 consider the problem of constructing a HIBE protocol which is secure in the full model against chosen ciphertext attacks without using random oracle. Known techniques (generic as well as non-generic) convert an $(h+1)$-level CPA-secure HIBE protocol into an $h$-level CCA-secure HIBE protocol. Applied to known constructions, these result in an $h$-level CCA-secure HIBE protocol whose security degradation is exponential in $(h+1)$. In this paper, we modify a recent construction of a CPA-secure HIBE to obtain a HIBE protocol in the KEM/DEM framework which is CCA-secure in the full model and without the random oracle assumption. The main feature of our construction is that the security degradation for an $h$-level HIBE protocol is exponential in $h$. The reduction of the exponent of security degradation from $(h+1)$ to $h$ can be significant in reducing the size of the underlying groups for practical applications. The security of the new protocol is based on the hardness of the decisional bilinear Diffie-Hellman (DBDH) problem.


## 1 Introduction

Identity based encryption [25, 7] 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) [20, 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 [20, 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 [27] presented an efficient construction of an IBE which is secure in the same setting. An extension of Waters' construction has been independently described in [12] and [24]. This leads to a controllable trade-off between the size of the public parameters and the efficiency of the protocol (see [12] for details).

A construction of a HIBE secure in the full model without using the random oracle heuristic is given in [27]. A recent work [13], describes a HIBE which builds on the suggestion in [27] by reducing the number of public parameters. The constructed HIBE is secure against chosen plaintext attacks (CPA-secure).

### 1.1 The Problem

We consider the problem of constructing a HIBE which is secure in the full model against adaptive chosen ciphertext attacks (CCA-secure). By full model, we mean the model introduced by Boneh-

Franklin [7] and not the weaker selective-ID model introduced later. The security is based on the conjectured computational hardness of the decisional bilinear Diffie-Hellman (DBDH) problem. Also the security proof does not use the random oracle assumption. Construction of a full model CCA-secure HIBE protocol without the random oracle assumption and based on the DBDH problem is a basic problem in the area of identity based encryption.

### 1.2 Known Solutions

The construction in [19] is based on the random oracle assumption and does not constitute a solution to the above problem. Two generic techniques [11, 8] are known which convert 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. Currently, there are only two known HIBE protocols which are CPA-secure in the full model without the random oracle assumption and are based on the DBDH problem – the Waters HIBE [27] and its improvement, the Chatterjee-Sarkar HIBE [13]. Both these protocols have a security degradation which is exponential in $h$, the maximum number of levels in the HIBE protocol. Applying the generic techniques to these CPA-secure HIBE protocols, one obtains an $h$-level CCA-secure HIBE whose security degradation is exponential in $(h+1)$. There is also a non-generic method of achieving CCA-security for a HIBE protocol [9]. This method also results in an $h$-level CCA-secure HIBE whose security degradation is exponential in $(h+1)$.

### 1.3 Our Contributions

We present two solutions to the problem in Section 1.1. Both of these are obtained by modifying the CPA-secure HIBE protocol given in [13]. The first construction is a CCA-secure hierarchical identity based key encapsulation mechanism (HIB-KEM). For an identity tuple having $j$ components, the decapsulation algorithm of this construction uses $j$ pairings to verify the well-formedness of the encapsulated key. In our second construction, we do away with these pairing computations while retaining the KEM/DEM framework. The separation of the KEM/DEM boundary in this case requires us to extend the notion of Tag-KEM/DEM framework [1] to the hierarchical identity based setting. The first construction is conceptually easier to understand where as the second construction is more efficient. We consider the second construction to be the main contribution of this paper. The first construction is to be considered as a conceptual step in the development of the second construction.

### 1.4 Is Our Solution Significant?

We first note that in the current state of knowledge, there is no known HIBE protocol which is secure in the full model and which can avoid security degradation exponential in $h$. This is true irrespective of whether the random oracle assumption is used or security is based on some (possibly weaker) variant of the DBDH problem. Avoiding such security degradation is a major open problem in HIBE construction and this paper does not present a solution to this problem.

On the other hand, the important feature of both our constructions is the fact that the security degradation of an $h$-level HIBE is exponential in $h$ (and not $(h+1)$ as could be obtained previously). Thus, we have prevented a loss of security in transforming from CPA-security to CCA-security. To the best of our knowledge, none of the existing techniques in the literature can be put together to obtain a solution to the problem in Section 1.1 and which is better than our second construction.

In more concrete terms, the security degradation reduces from approximately $q^{h+1}$ to $q^h$. In an asymptotic sense this is not significant. In cryptography, however, it has become customary to interpret security reductions in terms of what is called concrete security. This is usually done by substituting concrete values for the different parameters of the protocol. A typical value of $q$ is $2^{30}$. With this value of $q$, the security degradation reduces from $2^{30(h+1)}$ to $2^{30h}$. Thus, we can prevent an extra 30-bit security degradation which would be incurred by known techniques for attaining CCA-security. This improvement comes at no extra cost in efficiency of the protocol, i.e., our protocol (from the second construction) is as efficient as any protocol that can be obtained using other known techniques.

We consider the result of reducing security degradation at no extra cost in efficiency to be an advancement on understanding of HIBE protocols. On the other hand, we do note that the main open problem in construction of HIBE protocols is to obtain a (CPA-secure) HIBE protocol which does not have an exponential security degradation. The current paper does not provide a solution to this problem.

### 1.5 Our techniques

For the most part, we combine techniques from different papers to attain our result. The only new technique that we introduce is to incorporate information about the length of the identity into the ciphertext. Our main claim to technical novelty is in making different ideas fit together properly to obtain a correct and efficient construction. More details on our techniques are discussed below.

In the security proof for any CCA-secure (H)IBE protocols, there are two types of queries that need to be handled by a simulator – key extraction queries and decryption queries. The techniques for handling these queries are separate and the protocol usually incorporates different features for tackling these two issues. In fact, these issues are sufficiently separate that one can actually "add-on" the features for handling decryption queries to a CPA-secure protocol which already incorporates features for handling key extraction queries. This is clearly the case for generic conversions from CPA-secure protocol to CCA-secure protocol. Though less evident, this is also true for the non-generic conversion technique given by [9].

As mentioned earlier, we modify the CPA-secure protocol from [13] to obtain a a CCA-secure protocol. This CPA-secure protocol includes features for handling key extraction queries and these features are a development of algebraic techniques introduced in [4] and [27]. CCA-security is attained by adding "something more" to the protocol in [13].

Our technique for attaining CCA-security is based on the technique of [9] which in turn uses algebraic ideas from the construction of IBE given in [4]. We make several small but important changes to prevent the loss of one level of the HIBE in moving from CPA-security to CCA-security. This results in a reduced security degradation for our protocol.

The well-formedness of the ciphertext in the first construction is ensured through several pairing based verifications. These are computationally costly. The second construction does away with these pairings by using a MAC based technique to ensure well formedness. We use ideas from the Tag-KEM/DEM technique of [1] for this. Basically, the Tag-KEM/DEM construction of [1] can be considered to be made up of two parts. One part consists of algebraic techniques based on the Kurosawa-Desmedt [23] protocol (which in turn is based on the Cramer-Shoup [16] protocol). The other part consists of several components – DEM, MAC, KDF, CRHF. At a high level, our second construction can be seen replacement of the algebraic part of the Tag-KEM/DEM construction of [1] by the pairing based HIBE technique, where as the other part remains more or less unchanged.

### 1.6 Related Work

Apart from the references mentioned above, [22] also consider the problem of non-generic conversion of a CPA-secure IBE based on [27] to a CCA-secure IBE. The main focus of the work is on IBE and they make a passing remark on how to modify the IBE construction to obtain a HIBE. This remark is worked out in details in the recent eprint report [3].

We would like to point out that the HIB-KEM construction suggested in [22] and spelled out in [3] is incorrect. The problem is that in their protocol if we obtain a ciphertext corresponding to a $j$-level identity, then it is easy to obtain a valid ciphertext for its $(j-1)$-level prefix by simply discarding the component of the ciphertext corresponding to the last level of the identity. This leads to an easy attack on the protocol. More details are given in Section 3.1.

The attack mentioned above is not our observation. An earlier version of this paper, had the same problem as that of [22, 3]. This problem was pointed out to us by an anonymous reviewer of the paper (for either PKC 2006 or Crypto 2006) who considered it difficult to fix. Fortunately, we have been able to fix the problem using a simple technique to ensure that the length of the identity tuple enters the ciphertext.

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 [14]. The work [21] also considers the same problem. Since the hardness assumptions of this paper and that of [14, 21] are different, we do not compare the current protocol with that given in these two papers.

## 2 Preliminaries

In this section, we present the basic definitions of HIBE and HIB-KEM as well as construction of CPA-secure HIBE protocol from [13]. Due to lack of space, the security model, bilinear map, hardness assumption and components such as DEM, MAC, KDF and CRHF are defined in Appendix A.

### 2.1 HIBE Protocol

*Definition:* Following [20, 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 $\mathsf{v}$ 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 $\mathsf{v}$, the secret $d_{\mathsf{v}|_{j-1}}$ corresponding to its $(j-1)$-level prefix and $pk$ and returns as output $d_{\mathsf{v}}$, the secret key corresponding to $\mathsf{v}$. 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 $\mathsf{v}$, the message $M$ and $pk$, and returns $C$, the ciphertext obtained by encrypting $M$ under $\mathsf{v}$ and $pk$.
– $\mathsf{HIBE.Decrypt}(\mathsf{v}, d_{\mathsf{v}}, C, pk)$: Takes as input $\mathsf{v}$, the secret key $d_{\mathsf{v}}$ corresponding to $\mathsf{v}$, a ciphertext $C$ and $pk$. Returns either $\mathsf{bad}$ 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 $\mathsf{v}$, $d_{\mathsf{v}}$, $C$, $pk$, $sk$ and $M$ associated by the above four algorithms.

*HIB-KEM:* A KEM is used to construct a hybrid encryption scheme. In such a scheme, the actual encryption is done using a symmetric key algorithm and the secret key of the symmetric encryption is encapsulated using a public key procedure. The extension of the notion of KEM towards the construction of a hybrid HIBE is quite straightforward. Only the encryption and decryption algorithms of HIBE are respectively changed to the following encapsulation and decapsulation algorithms. Let $\mathcal{K}_D$ be the key space of a suitable symmetric encryption algorithm.

– $\mathsf{HIBE.Encap}(\mathsf{v}, pk)$: Takes as input $\mathsf{v}$ and $pk$, and returns $(\omega, dk)$, where $dk \in \mathcal{K}_D$ and $\omega$ is an encapsulation of $dk$ under $\mathsf{v}$.
– $\mathsf{HIBE.Decap}(\mathsf{v}, d_{\mathsf{v}}, \omega, pk)$: Takes as input $\mathsf{v}$, the secret key $d_{\mathsf{v}}$ corresponding to $\mathsf{v}$, an encapsulation $\omega$ and $pk$. Returns either $\mathsf{bad}$ or $dk$, the secret key of the symmetric encryption algorithm.

### 2.2 CPA-Secure HIBE Construction from [13]

We describe the CPA-secure HIBE protocol given in [13]. Later we modify this to obtain a CCA-secure HIB-KEM protocol.

Let $G_1$ and $G_2$ be cyclic groups having the same prime order $p$. We use a cryptographic bilinear map $e : G_1 \times G_1 \to G_2$ the definition of which is given in Section A.3.

*Set-Up:*

*Depth.* The maximum depth of the HIBE is $h$.
*Identity.* An identity $\mathsf{v}$ is a tuple $(\mathsf{v}_1, \dots, \mathsf{v}_j)$ where $j \in \{1, \dots, h\}$ with each $\mathsf{v}_k = (\mathsf{v}_1^{(k)}, \dots, \mathsf{v}_l^{(k)})$ and $\mathsf{v}_i^{(k)}$ is an $(n/l)$-bit string which will also be considered to be an integer in the range $\{0, \dots, 2^{n/l} - 1\}$. Choosing $l = n$, gives $\mathsf{v}_k$ to be an $n$-bit string as considered by Waters [27]. We set $N = 2^n$.
*Public Parameters.* The public parameters are the following elements: $P$, $P_1 = \alpha P$, $P_2$, $U'_1, \dots, U'_h$, $U_1 \dots, U_l$, where $G_1 = \langle P \rangle$, $\alpha$ is chosen randomly from $\mathbb{Z}_p$ and the other quantities are chosen randomly from $G_1$.
*Master Secret.* The master secret is $\alpha P_2$.

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

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

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 $V_k(v)$.

*Key Generation:* Let $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$, $1 \leq j \leq h$, be the identity for which the private key is required. Choose $r_1, \ldots, r_j$ randomly from $\mathbb{Z}_p$ and define $d_{\mathsf{v}} = (d_0, d_1, \ldots, d_j)$ where $d_0 = \alpha P_2 + \sum_{k=1}^{j} r_k V_k(\mathsf{v}_k)$ and $d_k = r_k P$ for $1 \leq k \leq j$. This defines a private key corresponding to an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$. The requirement of HIBE protocol is to be able to delegate keys. The delegation technique is essentially based on the Boneh-Boyen [4] technique and details can be found in [13].

*Encryption:* Let $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$ be the identity under which a message $M \in G_2$ is to be encrypted. Choose $t$ to be a random element of $\mathbb{Z}_p$. The ciphertext is

$$(C_0 = M \times e(P_1, P_2)^t, C_1 = tP, B_1 = tV_1(\mathsf{v}_1), \ldots, B_j = tV_j(\mathsf{v}_j)). \tag{2}$$

*Decryption:* Let $C = (C_0, C_1, B_1, \ldots, B_j)$ be a ciphertext and the corresponding identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$. Let $(d_0, d_1, \ldots, d_j)$ be the decryption key corresponding to the identity $\mathsf{v}$. The decryption steps are as follows.

*Verification.* Verify whether $C_0$ is in $G_2$, $C_1$ and the $B_i$'s are in $G_1$. If any of these verifications fail, then return $\mathsf{bad}$, else proceed with further decryption as follows.

*Return* $C_0 \times \frac{\prod_{k=1}^{j} e(B_i, d_i)}{e(d_0, C_1)}$.

## 3 CCA-Secure HIB-KEM

In this section, we modify the CPA-secure HIBE protocol in Section 2.2 to obtain a CCA-secure HIB-KEM protocol. The modification consists of certain additions to the set-up procedure as well as modification of the encryption and the decryption algorithm to obtain the encapsulation and decapsulation algorithms respectively. *No changes are required in the key generation algorithm and hence we do not include it below.* The additional changes 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).

*Set-Up:* In addition to the set-up for the HIBE protocol of Section 2.2 the following are required.

- A collision resistant hash function $H$ is randomly chosen from a CRHF family $\{H_k\}_{k \in \mathcal{K}}$, where each $H_k : \{1, \ldots, h\} \times G_1 \to \mathbb{Z}_p$.
- A random element $W \in G_1$.

*Key Generation:* This is the same as the key generation of the protocol in 2.2.

*Key Encapsulation:* Let $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$ be the identity for which a key encapsulation is to be done.

- Choose $t$ to be a random element of $\mathbb{Z}_p$.
- The secret key of the symmetric encryption algorithm is (a suitable hash of) $e(P_1, P_2)^t$.
- The encapsulated key is formed as follows:
  Compute $tP$; $\gamma = H(j, tP)$; and $W_\gamma = W + \gamma P_1$.
  The encapsulated key is
  $(C_1 = tP, C_2 = tW_\gamma, B_1 = tV_1(\mathsf{v}_1), \ldots, B_j = tV_j(\mathsf{v}_j))$.

Compared to (2), the encapsulated key in the current protocol has one extra component ($C_2$) but does not have the encryption of the message.

*Key Decapsulation:* Let $C = (C_1, C_2, B_1, \ldots, B_j)$ be an encapsulated key corresponding to an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$. The decapsulation steps are as follows.

– Compute $V_1(\mathsf{v}_1), \ldots, V_j(\mathsf{v}_j)$. (These can be precomputed.)
– Compute $\gamma = H(j, C_1)$ and $W_\gamma = W + \gamma P_1$.
– Perform the following verifications:
  Verify that each component of $C$ is an element of $G_1$;
  verify $e(C_1, W_\gamma) = e(P, C_2)$; and
  for $1 \leq i \leq j$, verify $e(C_1, V_i) = e(P, B_i)$.
  Note that all the verifications can be done publicly and does not require the secret key.
– If any of the verifications fail, then return $\mathsf{bad}$ else return $\frac{e(d_0,C_1)}{\prod_{k=1}^{j} e(B_i,d_i)}$.

The purpose of the public verification tests is to ensure that any decapsulation query is of the form $(tP, tW_\gamma, tV_1(\mathsf{v}_1), \ldots, tV_j(\mathsf{v}_j))$ for some $t$. This ensures the well-formedness of the ciphertext. The construction of HIB-KEM can be viewed as consisting of two structures – a HIBE and a selective-ID secure BB-IBE. The public parameters of the IBE consists of $(P, P_1, P_2, W)$. The encryption consists of encrypting the message twice – once for the HIBE and the second time for the IBE under the identity derived from the randomizer $tP$ and the depth $j$ of the identity. This second encryption is not actually used in the protocol. It is used in the security proof, where the simulator derives the secret key corresponding to the identity obtained from $tP$ and $j$ and then decrypts the message. This technique for handling decryption queries is essentially due to Boyen-Mei-Waters [9].

*An Alternative Verification Idea.* The well-formedness of the ciphertext is ensured by the comparisons "for $1 \leq i \leq j$, verify $e(C_1, V_i) = e(P, B_i)$". This requires $2j$ many pairing computations which is quite costly. An alternative (suggested by a reviewer for Eurocrypt 2007) is the following. Randomly choose $s_1, \ldots, s_j$ from $\mathbb{Z}_p$. Replace the computation within quotes by the following computation: Verify whether $e(C_1, \sum_{i=1}^{j} s_i V_i) = e(P, \sum_{i=1}^{j} s_i B_i)$. This removes the $2j$ pairing computations from the verification. However, it introduces $2j$ scalar multiplications for computing $s_1 V_1, \ldots, s_j V_j, s_1 B_1, \ldots, s_j B_j$. While a scalar multiplication is certainly cheaper than a pairing computation, by itself it is quite costly. In the worst case, $j = h$ and there is an overhead of $2h$ scalar multiplication for verifying the well-formedness of the ciphertext using this approach. In our second construction (given in Section 4.3) we show that the pairing verifications can be done away with at essentially no extra cost. In particular, no additional scalar multiplications are required.

### 3.1 An Incorrect Construction

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 HIB-KEM protocol above, $H()$ 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 encapsulation for a $(j+1)$-level identity can be converted to an encapsulation 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.) We consider this in more details.

Suppose that $H()$ is a map from $G_1$ to $\mathbb{Z}_p$ and the rest of the protocol is unchanged. Now suppose that $(C_1, C_2, B_1, \ldots, B_j)$ is a ciphertext obtained for an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$, with

$j > 1$. Then it is easy to see that $(C_1, C_2, B_1, \ldots, B_{j-1})$ is a valid ciphertext corresponding to the identity $(\mathsf{v}_1, \ldots, \mathsf{v}_{j-1})$. Not only that, both these encapsulate the same secret symmetric key.

This leads to an easy attack on the corresponding protocol. Let the challenge identity be $\mathsf{v}$ and let $(C_1, C_2, B_1, \ldots, B_j)$ be the obtained challenge ciphertext. In the second phase, ask the decryption oracle for the decryption of $(C_1, C_2, B_1, \ldots, B_{j-1})$ under $(\mathsf{v}_1, \ldots, \mathsf{v}_{j-1})$. Under the rules of the game, this is an allowed query. The answer will be the symmetric key which is encapsulated by the challenge ciphertext. As mentioned in the introduction, the HIB-KEM protocol suggested in [22] and described in [3] suffers from the above problem.

In our protocol, the domain of $H()$ is $\{1, \ldots, h\} \times G_1$. This ensures that the component $C_2$ of the ciphertext incorporates information about the length of the identity. With this, the simple mauling of the ciphertext outlined above no longer works and one can obtain a proof of security as described below.

### 3.2 Security Statement

The protocol in Section 2.2 is proved to be secure in [13] and the proof is based on ideas introduced in [27, 4]. Basically, the simulator is unable to answer certain types of key extraction queries and also unable to provide challenge ciphertext for certain challenge identities. This leads the simulator to abort and output a random bit. The main analysis is to lower bound the probability of not abort. Also, as introduced in [27] and increasingly used in different papers, the technique of "artificial abort" is used to ensure that the probability of not abort is almost the same for all adversarial queries. This introduces an extra component in the runtime of the simulator.

Since the new protocol is based on the protocol in Section 2.2, it inherits all the features of the earlier protocol. The exact security statement for the new protocol is given below.

**Theorem 1.** *The HIB-KEM protocol described above is $(\epsilon_{hib\text{-}kem}, t, q_{\mathsf{ID}}, q_C)$-CCA secure assuming that the $(t', \epsilon_{dbdh})$-DBDH assumption holds in $\langle G_1, G_2, e\rangle$, where*

$$\epsilon_{hib\text{-}kem} \leq 2\epsilon_{crhf} + \frac{\epsilon_{dbdh}}{\lambda};$$

$t' = t + \chi(\epsilon_{hibe})$*; and*

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

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

The proof of the Theorem is given in Section B. The statement of Theorem 1 is almost the same as that of Theorem 1 in [13] with two differences.

1. The above theorem states CCA-security where as [13] proves CPA-security.
2. The value of $\lambda$ 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 [13].

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

$$\epsilon_{hib\text{-}kem} \leq 2\epsilon_{crhf} + 2h(4lq_{\mathsf{ID}}2^{n/l})^h\epsilon_{dbdh}.$$

The corresponding bound in [13] is $2(4lq_{\mathsf{ID}}2^{n/l})^h\epsilon_{dbdh}$. Thus, we get an additional security degradation of $\epsilon_{dbdh}$ 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.

The statement of Theorem 1 is a little complicated. The complexity is actually inherited from the corresponding security statement in [13] for the protocol in Section 2.2. These arise from the requirement of tackling key extraction queries and providing challenge ciphertexts. In particular, $\lambda$ 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 [13], 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.

*What have we gained?* The currently known techniques 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 [13] (see Section 2.2), 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 $h$. The net effect is a substantial gain in controlling security degradation. The effect of security degradation on the size of the groups can be very pronounced as has been analyzed in [12]. Any significant gain in reducing security degradation has a considerable effect on the time required for performing scalar multiplication and pairing in the underlying groups.

*What have we lost?* The reduction in security degradation comes at a cost of increasing the total number of pairings in the decapsulation. Here, we emphasize that the *number* of pairings go up. However, when one chooses a larger size group to compensate for the larger security degradation, the total time required for executing the entire decapsulation algorithm may actually go up even though the total number of pairings is less. This is because each operation will then be performed on larger size groups. Settling this point needs a careful comparison of the total time of the two protocols in different size groups. Here, we do not perform such a comparison. The reason is that we are actually able to do away with almost all the pairing verifications during decapsulation.

## 4 Hierarchical ID-Based Tag-KEM

The CCA-secure HIB-KEM in Section 3 performs several pairing based verifications during decapsulation. There are $j$ pairing verifications for an identity of $j$ levels. The aim of these is to ensure the well-formedness of the encapsulated key. These pairings are quite costly and will take up the major time for decapsulation.

In this section, we describe a method to do away with the pairing verifications. Basically the following is done. The secret key in the encapsulation algorithm is $e(P_1, P_2)^t$. From this we produce two keys $(mk, dk)$ using a key derivation function (KDF). The key $mk$ is the secret key of a MAC algorithm, where as $dk$ is the secret key of a symmetric encryption algorithm. The actual message is encrypted using $dk$ to produce a ciphertext $\mathsf{cpr}$. A MAC $\mathsf{chksum}$ of this $\mathsf{cpr}$ is computed under $mk$. The $\mathsf{chksum}$ is sent along with the encrypted message. The idea is that if the adversary changes the public key part, then the keys $(mk, dk)$ will change and the $\mathsf{chksum}$ will not be verified at the receiving end. Thus, we can do away with the pairing verifications.

The above approach does not separate between the public key and the symmetric encryption algorithm. It is certainly convenient to separate the two parts, as then one can separately reason about the security requirements of the two parts. To obtain such a separation, we have to divide the encapsulation algorithm into two phases. In the first phase, the keys $(mk, dk)$ are produced. The input to the first phase are $\mathsf{v}$ and $pk$ as usual. The ciphertext produced by the DEM is input to the second phase, which then produces a MAC on it. While looking at the KEM part, we want to remain oblivious of the DEM. We do this by saying that the second phase of the encapsulation algorithm takes a tag as input. Thus, what we are doing is really drawing the abstraction boundary between the KEM and the DEM parts a little differently.

This approach has been recently adopted in the context of PKE [1] where the KEM part has been called a Tag-KEM. A generic composition result proved in [1] shows that it is possible to combine a CCA-secure Tag-KEM with a one-time secure DEM to obtain a CCA-secure PKE. The notion of Tag-KEM can be easily extended to the HIBE setting. In this section, we briefly summarize this extension. In doing so, we will closely follow the notation used in [1]. In the following, the set $\mathcal{K}_D$ denotes the key space of a suitable symmetric encryption algorithm.

*Definition:* HIB-Tag-KEM ($\mathsf{HTKEM}$) is defined by five algorithms.

- $\mathsf{HTKEM.Setup}$: 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 tags, encapsulated keys and identities.
- $\mathsf{HTKEM.KeyGen}(\mathsf{v}, d_{\mathsf{v}|_{j-1}}, pk)$: This is the same as that for HIBE defined in Section 2.1.
- $\mathsf{HTKEM.Key}(\mathsf{v}, pk)$: Takes as input $pk$ and $\mathsf{v}$ and returns $(\omega, dk)$ as output, where $dk \in \mathcal{K}_D$ and $\omega$ is state information.
- $\mathsf{HTKEM.Enc}(\omega, \mathsf{cpr}, \mathsf{v})$: Here $\mathsf{cpr}$ is the tag. Outputs $\psi$.
- $\mathsf{HTKEM.Decap}(\mathsf{v}, d_{\mathsf{v}}, \psi, \mathsf{cpr}, pk)$: Outputs $dk$.

As usual, for soundness, we require that $\mathsf{HTKEM.Decap}(\mathsf{v}, d_{\mathsf{v}}, \psi, \tau, pk) = dk$ must hold for all $\mathsf{v}$, $d_{\mathsf{v}}$, $pk$, $sk$, $dk$, $\psi$, $\tau$ associated by the above five algorithms. For more details on the interpretation of the above model in the PKE setting see [1]. Much of these also hold in the identity based setting.

### 4.1 Security Model for HIB-Tag-KEM

The adversary $\mathcal{A}^{\mathcal{O}_D, \mathcal{O}_P}$ is a probabilistic algorithm with access to the two oracles $\mathcal{O}_D$ and $\mathcal{O}_P$. The adversarial game is defined as follows.

1. generate $(pk, sk)$ using $\mathsf{HTKEM.Setup}$.
2. $(\mathsf{st}_1, \mathsf{v}^*) \leftarrow \mathcal{A}^{\mathcal{O}_D, \mathcal{O}_P}(pk)$.
3. $(\omega^*, dk_1) \leftarrow \mathsf{HTKEM.Key}(\mathsf{v}^*, pk)$;
   choose $dk_0$ randomly from $\mathcal{K}_D$; choose $\delta$ to be a random bit.
4. $(\mathsf{cpr}^*, \mathsf{st}_2) \leftarrow \mathcal{A}^{\mathcal{O}_D, \mathcal{O}_P}(\mathsf{st}_1, dk_\delta)$.
5. $\psi^* \leftarrow \mathsf{HTKEM.Enc}(\omega^*, \mathsf{cpr}^*, \mathsf{v}^*)$.
6. $\delta' \leftarrow \mathcal{A}^{\mathcal{O}_D, \mathcal{O}_P}(\mathsf{st}_2, \psi)$.

The variables $\mathsf{st}_1, \mathsf{st}_2$ are state variables that the adversary uses to carry information from one phase to another. There are several natural restrictions on the use of the oracles by the adversary. In Step 2, the adversary outputs an identity $\mathsf{v}^*$; the adversary must not query $\mathcal{O}_P$ with $\mathsf{v}^*$ or any of its prefixes in either Steps 2, 4 or 6. In Step 6, the adversary is not allowed to query $\mathcal{O}_D$ with $(\mathsf{v}^*, \psi^*, \tau^*)$. Additionally, certain queries are useless for the adversary and we will assume that the adversary does not make such queries. If the adversary knows the secret key corresponding to an identity $\mathsf{v}$ (by querying $\mathcal{O}_P$), then he does not query $\mathcal{O}_D$ on $\mathsf{v}$ or any identity of which $\mathsf{v}$ is a prefix. The advantage of the adversary in winning this game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{HTKEM}} = \left|\Pr[\delta = \delta'] - 1/2\right|.$$

As in the case of HIB-KEM, we define the resource bounded versions of the above advantage in an analogous manner.

### 4.2 Generic Construction of Hybrid CCA-Secure HIBE

As mentioned earlier, the importance of considering HTKEM is that one can generically combine a CCA-secure HTKEM with a one-time secure DEM to obtain a CCA-secure HIBE. This parallels a similar construction in the PKE setting as shown in [1].

**Set-Up:** Invoke $\mathsf{HTKEM.Setup}$ to obtain $(pk, sk)$.
**Key Extraction:** Given $\mathsf{v}$ return $\mathsf{HTKEM.KeyGen}(\mathsf{v}, d_{\mathsf{v}|_{j-1}}, pk)$.
**Encryption:** A message $M$ is encrypted using an identity $\mathsf{v}$ in the following manner.

1. $(\omega, dk) \leftarrow \mathsf{HTKEM.Key}(\mathsf{v}, pk)$;
2. $\mathsf{cpr} \leftarrow \mathsf{DEM.Enc}_{dk}(M)$;
3. $\psi \leftarrow \mathsf{HTKEM.Enc}(\omega, \mathsf{cpr}, \mathsf{v})$;
4. output $c = (\psi, \mathsf{cpr})$ and $\mathsf{v}$.

**Decryption:** Given $(c, \mathsf{v})$, the decryption is as follows.

1. $(\psi, \mathsf{cpr}) \leftarrow c$;
2. $dk \leftarrow \mathsf{HTKEM.Dec}(\mathsf{v}, d_{\mathsf{v}}, \psi, \mathsf{cpr}, pk)$;
3. $M \leftarrow \mathsf{DEM.Dec}_{dk}(\chi)$;
4. output $M$.

The above construction yields a CCA-secure (hybrid) HIBE. The corresponding result for PKE was proved in [1]. The same proof also works for the identity based protocol, with the obvious modification that any private key query by the adversary attacking HIBE is answered by the simulator by querying the key extraction oracle of the HTKEM. The rest of the proof goes through without any other change.

**Theorem 2.** *If HTKEM is $(\epsilon_{htkem}, t)$-CCA secure and DEM is $(\epsilon_{dem}, t)$-secure, then HIBE is $(\epsilon_{hibe}, t)$-CCA secure, where $\epsilon_{hibe} \leq 2\epsilon_{htkem} + \epsilon_{dem}$.*

Informally, we say that if HTKEM is CCA-secure and DEM is one-time secure, then the above hybrid HIBE is CCA-secure.

### 4.3 Construction of HIB-Tag-KEM

The main contribution of this paper is a modification of the HIB-KEM protocol of Section 3 to obtain a HIB-Tag-KEM protocol. This is a non-generic construction. (A generic construction of HIB-Tag-KEM along the lines of a generic construction of a Tag-KEM in [1] is possible but is of less interest.)

$\mathsf{HTKEM.Setup}$ *and* $\mathsf{HTKEM.KeyGen}(\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j), d_{\mathsf{v}|_{j-1}}, pk)$*.* These two are same as the HIB-KEM protocol in Section 3.

$\mathsf{HTKEM.Key}(\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j), pk)$*:*

1. Choose $t$ randomly from $\mathbb{Z}_p$;
2. Define $\phi = (C_1 = tP, C_2 = tW_\gamma, B_1 = tV_1(\mathsf{v}_1), \ldots, B_j = tV_j(\mathsf{v}_j))$, where $W_\gamma = W + \gamma P_1$ and $\gamma = H(j, C_1)$;
3. Set $K = e(P_1, P_2)^t$ and $(dk, mk) = \mathsf{KDF}(K)$;
4. output $(dk, \omega = (mk, \phi))$.

$\mathsf{HTKEM.Enc}(\omega, \mathsf{cpr}, \mathsf{v})$*:*

1. $(mk, \phi) = \omega$;
2. $\mathsf{chksum} = \mathsf{MAC.Sign}_{mk}(\mathsf{cpr})$;
3. output $\psi = (\phi, \mathsf{chksum})$.

$\mathsf{HTKEM.Dec}(\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j), d_{\mathsf{v}} = (d_0, d_1, \ldots, d_j), \psi, \mathsf{cpr}, pk)$*:*

1. $(\phi, \mathsf{chksum}) = \psi$ where $\phi = (C_1, C_2, B_1, \ldots, B_j)$;
2. Compute $W_\gamma = W + \gamma P_1$ where $\gamma = H(j, C_1)$;
3. if $e(C_1, W_\gamma) \neq e(P, C_2)$, return $\mathsf{bad}$;
4. $K = e(d_0, C_1)/(\prod_{k=1}^{j} e(B_i, d_i))$;
5. $(mk, dk) = \mathsf{KDF}(K)$;
6. if $\mathsf{MAC.Ver}_{mk}(\mathsf{cpr}, \mathsf{chksum}) \neq 1$, return $\mathsf{bad}$;
7. else return $dk$.

**Theorem 3.** *If $\{H_k\}_{k \in \mathcal{K}}$ is $(\epsilon_{crhf}, t)$-secure, MAC is $(\epsilon_{mac}, t')$-secure, KDF is $(\epsilon_{kdf}, t')$-secure and the $(\epsilon_{dbdh}, t + \chi(\epsilon_{htkem}))$-DBDH assumption holds in $\langle G_1, G_2, e\rangle$, then HTKEM is $(\epsilon_{htkem}, t, q_{\mathsf{ID}}, q_C)$-CCA secure, where,*

$$2\epsilon_{htkem} \leq 2\epsilon_{crhf} + \epsilon_{dbdh}/\lambda + 4\epsilon_{kdf} + 4hq_C(\epsilon_{kdf} + \epsilon_{mac}).$$

*Here $h$ is the maximum number of levels in the HIBE; $\chi()$ and $\lambda$ are as defined in Theorem 1. Further, $t' = t + O(\tau(q_{\mathsf{ID}} + q_C))$.*

We present the proof in Section C.

*Discussion:* The above construction is more efficient than that of Section 3 in the sense that we can avoid all but one pairing computations for the verification of well-formedness of the ciphertext. The technique to achieve CCA-security in [9] requires pairing verifications as in Section 3. On the other hand, the transformations in [11, 8] do not perform pairing computations to test well-formedness. Applying the MAC-based transformation of [8] to the protocol in Section 2.2 results in CCA-secure HIBE where the number of operations is approximately same as the number of operation in the protocol of this section.

CCA-secure protocols resulting from applying the previously known techniques yield an $h$-level HIBE whose exponent of security degradation is $h + 1$. In contrast, for both the protocols in Section 3 and this section, the exponent of security degradation for an $h$-level HIBE is $h$. This reduction in security degradation, while retaining the efficiency of MAC based approach, is the main contribution of this work.

## 5 Conclusion

We have presented two constructions of CCA-secure HIBE in the KEM/DEM framework. Both of these protocols are secure in the full model without random oracle and are obtained by modifying a recent construction of CPA-secure HIBE [13]. The first construction uses a number of expensive pairings to verify the well-formedness of the encapsulated key. The second construction removes these pairings by properly using a MAC algorithm. This requires us to extend the notion of Tag-KEM/DEM framework to the hierarchical identity based setting. The main point of both our constructions is that the security degradation for an $h$-level HIBE is exponential in $h$. Applying previous constructions [11, 8, 9] to the CPA-secure protocol in [13] yields a CCA-secure HIBE protocol where the security degradation for an $h$-level HIBE is exponential in $(h+1)$. This reduction in the exponent of security degradation from $(h+1)$ to $h$ is significant in choosing smaller size groups for practical implementations.

## 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 [15], 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 [15], pages 440–456. Full version available at Cryptology ePrint Archive; Report 2005/015.
7. 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.
8. Dan Boneh and Jonathan Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.
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. 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.
13. 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/.
14. 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.

15. 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.
16. Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.
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. 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.
21. 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/.
22. 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.
23. Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In Franklin [17], pages 426–442.
24. David Naccache. Secure and Practical Identity-Based Encryption. Cryptology ePrint Archive, Report 2005/369, 2005. http://eprint.iacr.org/.
25. 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.
26. Victor Shoup. Sequences of Games: a Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/.
27. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [15], pages 114–127.

## Appendix

## A Definitions

### A.1 Security Model

We describe the full security model for HIB-KEM which is a minor variant of the full security model for HIBE. The adversary $\mathcal{A}^{\mathcal{O}_D,\mathcal{O}_P}$ is a probabilistic algorithm with access to two oracles $\mathcal{O}_D$ (the decapsulation oracle) and $\mathcal{O}_P$ (the private key extraction oracle). On querying $\mathcal{O}_P$ with $\mathsf{v}$, the adversary obtains $d_{\mathsf{v}}$ the secret key corresponding to $\mathsf{v}$. Similarly, on querying $\mathcal{O}_D$ with $(\mathsf{v}, \omega)$, the adversary obtains either $dk$ or $\mathsf{bad}$.

The adversarial game is defined as follows.

1. generate $(pk, sk)$ using $\mathsf{HIBE.Setup}$.
2. $(\mathsf{state}, \mathsf{v}^*) \leftarrow \mathcal{A}^{\mathcal{O}_D,\mathcal{O}_P}(pk)$.
3. $(\omega^*, dk_1) \leftarrow \mathsf{HIBE.Encap}(\mathsf{v}^*, pk)$;
   choose $dk_0$ randomly from $\mathcal{K}_D$; choose $\delta$ to be a random bit.
4. $\delta' \leftarrow \mathcal{A}^{\mathcal{O}_D,\mathcal{O}_P}(\mathsf{state}, \omega^*, dk_\delta)$.

The variable $\mathsf{state}$ is used by the adversary to carry information from one phase to another. Step 2 correspond to the first phase of the game, whereby the adversary interacts with the oracles and produces a challenge identity $\mathsf{v}^*$. In Step 3, the challenge step, the adversary is given $\omega^*$ and either the secret key corresponding to $\omega^*$ or a random secret key according as $\delta$ is 1 or 0. The second phase is Step 4, where the adversary guesses the value of $\delta$. There are several natural restrictions on the use of the oracles by the adversary. In Steps 2 and 4, the adversary cannot ask $\mathcal{O}_P$ for the secret key of $\mathsf{v}^*$ or any of its prefixes. Similarly, in Step 4, it cannot ask $\mathcal{O}_D$ for the decapsulation of $(\mathsf{v}^*, \omega^*)$. Additionally, certain queries are useless for the adversary and we will assume that the adversary does not make such queries. If the adversary knows the secret key corresponding to an identity $\mathsf{v}$ (by querying $\mathcal{O}_P$), then he does not query $\mathcal{O}_D$ using $\mathsf{v}$ or any identity of which $\mathsf{v}$ is a prefix. The advantage of the adversary in winning this game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{HIB-KEM}} = \left|\mathsf{Pr}[\delta = \delta'] - 1/2\right|. \tag{3}$$

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

### A.2 CPA-Security

In the adversarial game, we can restrict the adversary $\mathcal{A}$ from querying the decryption oracle. $\mathsf{Adv}^{\text{HIB-KEM}}(t, q)$ 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 $(\epsilon, t, q)$-CPA secure, if $\epsilon = \mathsf{Adv}^{\text{HIB-KEM}}(t, q)$.

### A.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 [7] 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, 27] write both $G_1$ and $G_2$ multiplicatively though this is not true of the initial protocol papers [7, 19]. Here we follow the first convention as it is closer to the known examples.

### A.4 Hardness Assumption

The decisional bilinear Diffie-Hellman (DBDH) problem in $\langle G_1, G_2, e \rangle$ [7] 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

$$\mathsf{Adv}_{\mathcal{B}}^{\mathrm{DBDH}} = |\mathsf{Pr}[\mathcal{B}(P, aP, bP, cP, Z) = 1|\mathsf{Z\ is\ real}] - \mathsf{Pr}[\mathcal{B}(P, aP, bP, cP, Z) = 1|\ \mathsf{Z\ is\ random}]| \quad (4)$$

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 $t$. We have the $(\epsilon, t)$-DBDH assumption, if $\epsilon = \mathsf{Adv}^{\mathrm{DBDH}}(t)$.

### A.5 Components (DEM, MAC, KDF, CRHF)

In the following, we will require different components such as data encapsulation mechanism (DEM), message authentication code (MAC) and key derivation function (KDF). We briefly introduce and state the security notions for DEM, MAC and KDF.

A DEM is a symmetric encryption scheme that consists of two algorithms $\mathsf{DEM.Enc}$ and $\mathsf{DEM.Dec}$. We require a $\mathsf{DEM}$ to satisfy a notion of security called one-time security which is the following. An adversary chooses two equal length messages; one of this is randomly selected and the adversary is given a ciphertext for this message under a randomly chosen key; the adversary has to determine which message has been encrypted.

A MAC has two algorithms $\mathsf{MAC.Sign}$ (the signing algorithm) and $\mathsf{MAC.Ver}$ (the verification algorithm). Both the algorithms use a common secret key. The security notion required of the MAC scheme is the following. The adversary chooses a message $\mathsf{msg}$; a $\mathsf{tag}$ is produced under a randomly chosen secret key $mk$ by computing $\mathsf{tag} = \mathsf{MAC.Sign}_{mk}(\mathsf{msg})$; the adversary is given $\mathsf{tag}$; the adversary now has to produce a message tag pair $(\mathsf{msg}', \mathsf{tag}')$ such that $\mathsf{msg} \neq \mathsf{msg}'$ and $\mathsf{MAC.Ver}_{mk}(\mathsf{msg}', \mathsf{tag}')$ is true.

A KDF is a function $\mathsf{KDF}()$ which takes an input $K$ and produces two keys $(dk, mk)$ as output, where $dk$ (resp. $mk$) is the secret key for the DEM (resp. MAC). The security notion for KDF is the following. For a randomly chosen $K$, the adversary has to distinguish between $\mathsf{KDF}(K)$ and a randomly chosen $(dk, mk)$.

A function family $\{H_k\}_{k\in\mathcal{K}}$ is said to be a collision resistant hash function (CRHF) family if the following adversarial task is difficult. The adversary is given a randomly chosen $k \in \mathcal{K}$ and has to find $x \neq x'$ in the domain of the family such that $H_k(x) = H_k(x')$. We say that the family is $(\epsilon, t)$-secure if the maximum probability of an adversary running in time $t$ and of finding a collision is $\epsilon$.

*Notation:* We say that a DEM (resp. MAC, KDF, CRHF) is $(\epsilon, t)$-secure if the maximum advantage of an adversary running in time $t$ of breaking the DEM (resp. MAC, KEM) is $\epsilon$. Similarly, we say that a HIBE (resp. HIB-KEM) is $(\epsilon, t, q_{\mathsf{ID}}, q_C)$-CCA secure if the maximum advantage of an adversary running in time $t$ and making $q_{\mathsf{ID}}$ key extraction queries and $q_C$ decryption (resp. decapsulation) queries of breaking the HIBE (resp. HIB-KEM) is $\epsilon$. Lastly, we say that the $(\epsilon, t)$-DBDH assumption holds if the maximum advantage of an adversary running in time $t$ for solving DBDH is $\epsilon$. By $\epsilon_{xxx}$ we will denote the advantage corresponding to XXX, where XXX is one of DBDH, HIBE, HIB-KEM, DEM, MAC, KDF, CRHF.

## B Proof of Theorem 1

The construction of CCA-secure HIB-KEM in Section 3 is built on the construction of CPA-secure HIBE given in [13] (see Section 2.2). The proof of security given in [13] shows how to answer key-extraction queries and generate the challenge ciphertext. Our proof of Theorem 1 incorporates these aspects of the proof in [13]. Additionally, the proof shows how to answer decapsulation queries. Further, the protocol in Section 3 produces a ciphertext which has one more component than the protocol in [13]. During challenge generation, we show how to generate this component as well.

It might appear that to understand the proof of Theorem 1, it is essential for a reader to have detailed knowledge of the proof given in [13]. This is actually not the case. Since the techniques for handling key extraction queries and decapsulation queries are separate, in the proof below we focus only on the decapsulation queries and gloss over the details of key extraction queries which have already been provided in [13]. The parts of the game which are new to this paper (in particular the simulation of decapsulation queries and the relevant part on challenge generation) can be understood without reading [13].

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

**Game 0:** This is the usual adversarial game for defining CCA-security of HIB-KEM protocols. We assume that the adversary's runtime is $t$, it makes $q_{\mathsf{ID}}$ key-extraction queries and $q_{\mathsf{C}}$ decapsulation queries. Also, we assume that the adversary maximizes the advantage among all adversaries with similar resources. Thus, we have $\epsilon_{hib\text{-}kem} = \left|\mathsf{Pr}[X_0] - \frac{1}{2}\right|$.

**Game 1:** This is the same as Game 0, with the following change. If the adversary ever submits two decapsulation queries of the forms $(C_1, C_2, B_1, \ldots, B_j)$ and $(C'_1, C'_2, B'_1, \ldots, B'_{j'})$, with $(j, C_1) \neq (j', C'_1)$ and $H(j, C_1) = H(j, C'_1)$, then the simulator rejects the second query. Let $F_1$ be the event that a decapsulation query is rejected only by this check. It is easy to see that $\mathsf{Pr}[F_1] \leq \epsilon_{crhf}$. If $F_1$ does not occur, then Game 0 and Game 1 are identical. Using the difference lemma (as named in [26]), we obtain

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

**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 decapsulation 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 $\mathsf{flg}$ 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 $bP_2 = abP$ which is not known to the simulator.

The public parameters $(U'_1, \ldots, U'_h, U_1, \ldots, U_l)$ are required to handle key extraction queries. They have no role in answering decapsulation queries. The proper construction of these parameters are given in [13] and we do not include these here.

The parameter $W$ is required for answering decapsulation queries (and is not present in [13]). We show how to define $W$. Randomly choose a $j_\theta$ from $\{1, \ldots, h\}$ and compute $\gamma = H(j_\theta, P_3)$; choose $\beta$ randomly from $\mathbb{Z}_p$ and define $W = -\gamma P_1 + \beta P$. The choice of $j_\theta$ 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 are described in detail in [13] and hence we do not provide these here. We only note that answering certain queries require the use of the values $a$ or $b$. In all such cases, the simulator set $\mathsf{flg}$ to one.

*Decapsulation Query:* Suppose $C = (C_1, C_2, B_1, \ldots, B_j)$ is a decapsulation query for the identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$. Compute $\gamma' = H(j, C_1)$ and $W_{\gamma'} = W + \gamma' P_1$. The simulator then runs the public verification tests (i.e., $e(C_1, W'_\gamma) = e(P, C_2)$ and for $1 \leq i \leq j$, $e(C_1, V_i) = e(P, B_i)$) from the decapsulation procedure and proceeds if the test succeeds. If the test fails, it returns $\mathsf{bad}$ to $\mathcal{A}$. Note that, at this point, since we have already 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 $W = -\gamma P_1 + \beta P$.

$$\left.\begin{aligned} E_{\gamma'} &= \tfrac{-\beta}{\gamma'-\gamma} P_2 + r((\gamma' - \gamma)P_1 + \beta P) \\ &= aP_2 + \left(r - \tfrac{b}{\gamma'-\gamma}\right)(\gamma' P_1 + W) \\ &= aP_2 + \widetilde{r} W_{\gamma'} \\ d_{\gamma'} &= rP - \tfrac{1}{\gamma'-\gamma} P_2 \\ &= \widetilde{r} P. \end{aligned}\right\} \tag{5}$$

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 decapsulation can now be performed as follows.

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

Note that any decapsulation query can be answered without using the values of $a$, $b$ or $c$. Thus, $\mathsf{flg}$ is never set to 1 during this step.

*Challenge:* Let the challenge identity be $(\mathsf{v}_1^*, \ldots, \mathsf{v}_{j^*}^*)$. The challenge ciphertext is of the form $(C_1^*, C_2^*, B_1^*, \ldots, B_j^*)$. We set $C_1^* = cP$. The components $B_1^*$ to $B_j^*$ are also present in the protocol in [13] and the procedure to generate these elements are given in [13]. The same procedure is used in the current proof and hence we do not repeat the details here. Again, as in the case of key extraction queries, we note that for certain challenge identities, the generation of $B_1^*, \ldots, B_j^*$ require the use of the value of $c$. In this case, $\mathsf{flg}$ is set to one.

The component $C_2^*$ is new to this protocol and we show how to generate it. If $j^* \neq j_\theta$, then set $\mathsf{flg}$ to 1. 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 $\gamma' = H(j_\theta, C_1) = H(j^*, cP) = \gamma$. Set $Z_0 = Z$ and $Z_1$ to be a random element of $G_2$. Choose a random bit $\delta$ and return $(K^*, Z_\delta)$ to the adversary.

**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, $\mathsf{flg}$ is set to one. If $\mathsf{flg}$ 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 [13]. Also, a detailed analysis of the probability that $\mathsf{flg}$ remains 0 throughout the game has been carried out in [13]. All these analysis also hold for the current proof essentially due to the fact that answering decapsulation queries introduce no new abort condition. The only new abort condition is during challenge generation, when $j^* \neq j_\theta$. Since $1 \le j^*, j_\theta \le h$ and $j_\theta$ is chosen randomly from $\{1, \ldots, h\}$, the probability of this new abort is $1/h$.

With this small change, the analysis of [13] shows that

$$|\mathsf{Pr}[X_2] - \mathsf{Pr}[X_3]| \le \frac{\epsilon_{dbdh}}{2\lambda} + \frac{\epsilon_{hib\text{-}kem}}{2} \quad (6)$$

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.

In Game 3, the $Z$ is random and independent of the rest of the ciphertext. This provides the adversary with no information and hence $\mathsf{Pr}[X_3] = 1/2$. Combining the several relations we have,

$$\begin{aligned}
\epsilon_{hib\text{-}kem} &= \left|\mathsf{Pr}[X_0] - \frac{1}{2}\right| \\
&= |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_3]| \\
&= |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1]| + |\mathsf{Pr}[X_1] - \mathsf{Pr}[X_2]| + |\mathsf{Pr}[X_2] - \mathsf{Pr}[X_3]| \\
&= \epsilon_{crhf} + \frac{\epsilon_{dbdh}}{2\lambda} + \frac{\epsilon_{hib\text{-}kem}}{2}
\end{aligned}$$

Rearranging the expression provides the necessary result. □

## C Proof of Theorem 3

This proof is also via a sequence of games. The first three games are almost the same as the first three games in the proof of Theorem 1. Essentially, these three games show how to construct a DBDH-solver from an adversary which is able to break the protocol.

The main difference between the HIB-KEM and the HIB-Tag-KEM protocols is in the verification of well-formedness of ciphertexts. In the HIB-KEM protocol, this is done using several pairings, where as in the HIB-Tag-KEM protocol this is done using the MAC and KDF components. In the proof below, from Game 4 onward, the task is to show that attacking the protocol amounts to breaking either MAC or KDF. These games are very similar to the corresponding games in [1]. The reason is that the use of MAC and KDF in the HIB-Tag-KEM protocol is similar to the use of MAC and KDF in the Tag-KEM protocol of [1].

As in Section B, by $X_i$ we will denote the event that the bit $\delta$ is equal to the bit $\delta'$ in the $i$th game. We want to show that HTKEM is $(\epsilon_{htkem}, t, q_{\mathsf{ID}}, q_{\mathsf{C}})$-CCA secure.

**Game 0:** This is the usual adversarial game used in defining CCA-secure HIB-Tag-KEM. We assume that the adversary's runtime is $t$ and it makes $q_{\mathsf{ID}}$ key extraction queries and $q_{\mathsf{C}}$ decapsulation queries. Also, we assume that the adversary maximizes the advantage among all adversaries with similar resources. Thus, we have $\epsilon_{htkem} = \left|\mathsf{Pr}[X_0] - \frac{1}{2}\right|$.

**Game 1:** This is obtained by modifying Game 0 in the same manner as in the proof of Theorem 1. As before, we have, $|\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1]| \leq \epsilon_{crhf}$.

**Game 2:** This game is similar to the Game 2 in the proof of Theorem 1 with some obvious modifications for adjusting the game from the HIB-KEM format to the HIB-Tag-KEM format. Note that as part of these modifications, the pairing based public verification tests are no longer performed as these are not part of the HIB-Tag-KEM protocol.

**Game 3:** In this game, $Z$ is taken to be a random element of $G_2$. In a manner similar to that of Theorem 1, we have

$$|\mathsf{Pr}[X_2] - \mathsf{Pr}[X_3]| \leq \frac{\epsilon_{dbdh}}{2\lambda} + \frac{\epsilon_{htkem}}{2}. \tag{7}$$

**Game 4:** We now want to tackle the adversary's strategy of attacking either $\mathsf{KDF}$ or $\mathsf{MAC}$. We will assume that $u'_j$ and $u_i$ are known to the simulator such that $U'_j = u'_jP$ and $U_i = u_iP$. 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 $V_i = w_iP$. The adversary may submit a decryption query with $C_1 = tP$ and for some $i$, $B_i = t_1V_i$ with $t \neq t_1$. The knowledge of $w_i$ allows the simulator to test for this in the following manner: If $e(C_1, V_i) \neq e(w_iC_1, P)$, 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 either $\mathsf{KDF}$ or $\mathsf{MAC}$.

As in [1], let **Rejection Rule 0** be the normal protocol rejection rule and **Rejection Rule 1** be the rejection rule as 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 decapsulation queries. The adversary makes a total of $q_C$ decapsulation queries. We will use the superscript $(j)$ to denote the quantities related to the $j$th decryption query. For example, $K^{(j)}$ denotes the input to $\mathsf{KDF}()$ in the $j$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 **Rule 0** does not apply to the $\jmath$th query but **Rule 1** does apply. Then $\mathsf{Pr}[F_4] \leq h \times q_C \times \Pr[F'_4]$ and we have

$$|\mathsf{Pr}[X_3] - \mathsf{Pr}[X_4]| \leq \mathsf{Pr}[F_4] \leq h \times q_{\mathsf{C}} \times \mathsf{Pr}[F'_4]. \tag{8}$$

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 $V$.

**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 $\mathsf{v}^*$ 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

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

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* $\mathsf{v}^* \neq \mathsf{v}^{(\jmath)}$*:* In this case, the adversary can ask for the secret key for $\mathsf{v}^{(\jmath)}$ but not before making the malformed decryption query. If the adversary knows the secret key for $\mathsf{v}^{(\jmath)}$, then he can decrypt any ciphertext encrypted using $\mathsf{v}^{(\jmath)}$. Thus, it is useless for him to query the decryption oracle using $\mathsf{v}^{(\jmath)}$ *after* obtaining the secret key for $\mathsf{v}^{(\jmath)}$. 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 decapsulation query with $\mathsf{v}^{(\jmath)}$ and a later private key extraction query on $\mathsf{v}^{(\jmath)}$. With the private key $d_{\mathsf{v}^{(\jmath)}}$ returned to him, the adversary can decrypt his own earlier decapsulation query. Consistency requires that the output given to him on his decapsulation 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 $K^{(\jmath)}$, since this will be easily ensured.

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

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

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

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

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

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

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

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

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

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

Define $F'_5$ in a manner similar to $F'_4$. Then we have

$$\mathsf{Pr}[X_4] = \mathsf{Pr}[X_5] \text{ and } \mathsf{Pr}[F'_4] = \mathsf{Pr}[F'_5]. \tag{9}$$

**Game 6:** This is obtained from Game 5 by the following modification. In Game 5, the keys $(dk^*, mk^*)$ and $(dk^{(\jmath)}, mk^{(\jmath)})$ 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

$$|\mathsf{Pr}[X_5] - \mathsf{Pr}[X_6]| \leq 2\epsilon_{kdf} \text{ and } |\mathsf{Pr}[F'_5] - \mathsf{Pr}[F'_6]| \leq 2\epsilon_{kdf}. \tag{10}$$

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

Further, $\mathsf{Pr}[X_6] = 1/2$ since irrespective of the value of $b$ chosen by the simulator the adversary gets to see only a random string. Also, $\mathsf{Pr}[F'_6] \leq 2\epsilon_{mac}$. The factor of two again comes due to the fact that the adversary can forge one out of above two applications of MAC verification. (We note that the modifications done to Game 5 to obtain Game 6 are the same as the modifications done in [1] to Game 4 to obtain Game 5.) Finally, combining all the inequalities, we obtain

$$\begin{aligned}
\epsilon_{htkem} &= \left|\mathsf{Pr}[X_0] - \frac{1}{2}\right| \\
&= |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_6]| \\
&\leq |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1]| + |\mathsf{Pr}[X_2] - \mathsf{Pr}[X_3]| + |\mathsf{Pr}[X_3] - \mathsf{Pr}[X_4]| + |\mathsf{Pr}[X_5] - \mathsf{Pr}[X_6]| \\
&\leq \epsilon_{crhf} + \frac{\epsilon_{dbdh}}{2\lambda} + \frac{\epsilon_{htkem}}{2} + 2\epsilon_{kdf} + hq_{\mathsf{C}}(2\epsilon_{kdf} + 2\epsilon_{mac}).
\end{aligned}$$

Rearranging the inequality gives the desired relationship. □