HIBE with Short Public Parameters Without Random Oracle

Sanjit Chatterjee and Palash Sarkar

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

Abstract. At Eurocrypt 2005, Waters presented an identity based encryption (IBE) protocol which is secure in the full model without random oracle. In this paper, we extend Waters' IBE protocol to a hierarchical IBE (HIBE) protocol which is secure in the full model without random oracle. The only previous construction in the same setting is due to Waters. Our construction improves upon Waters' HIBE by significantly reducing the number of public parameters.

1 Introduction

The concept of identity based encryption (IBE) was introduced by Shamir in 1984 [17]. An IBE is a type of public key encryption where the public key can be any binary string. The corresponding secret key is generated by a private key generator (PKG) and provided to the legitimate user. The notion of IBE simplifies several applications of public key cryptography. The first efficient implementation and an appropriate security model for IBE was provided by Boneh and Franklin [5].

The PKG issues a private key associated with an identity. The notion of hierarchical identity based encryption (HIBE) was introduced in [14, 13] to reduce the workload of the PKG. An entity in a HIBE structure has an identity which 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 his identity. The security model for IBE was extended to that of HIBE in [14, 13].

The construction of IBE in [5] and of HIBE in [13], was proved to be secure in appropriate models using the *random oracle* heuristic, i.e., the protocols make use of cryptographic hash functions that are modeled as random oracle in the security proof. 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 [3]. Later, Waters [19] presented an efficient construction of an IBE which is secure in the same setting.

An important construction of a HIBE is given by Boneh-Boyen [2]. This paper describes a general framework for constructing a HIBE. For an h-level HIBE, the idea in [2] is to use h functions $\psi_1, \ldots, \psi_h$, where ψ_i is viewed as a hash function which maps the ith component of the identity tuple to an appropriate group element. This framework is instantiated in [2] to obtain a HIBE protocol which can be proved secure in weaker model called the selective-ID (sID) model.

The construction by Waters in [19] can be viewed as another instantiation of a 1-level BB-framework [2]. Identities are considered to be n-bit strings. The construction uses group elements U', $U_1, \ldots, U_n$ (and P, P_1, P_2) as public parameters. A natural extension of this construction to an h-level HIBE is given in [19]. In this extension, for an h-level HIBE, the public parameters will be of the form $U'_1, U_{1,1}, \ldots, U_{1,n}$, $U'_2, U_{2,1}, \ldots, U_{2,n}$, …, $U'_h, U_{h,1}, \ldots, U_{h,n}$. One still requires the parameters P, P_1, P_2, giving rise to $3 + (n+1)h$ many parameters.

Our Contributions: We present a HIBE which can be proved to be secure in the full model assuming the decisional bilinear Diffie-Hellman problem to be hard without using the random oracle heuristic. Our construction can also be viewed as another instantiation of the BB-framework [2]. The public parameters for an h-level HIBE are of the form $U'_1, \ldots, U'_h$, $U_1, \ldots, U_n$. In other words, the parameters $U'_1, \ldots, U'_h$ correspond to the different levels of the HIBE, whereas the parameters $U_1, \ldots, U_n$ are the same for all the levels. These parameters $U_1, \ldots, U_n$ are reused in the key generation procedure. We require $3 + n + h$ parameters compared to $3 + (n+1)h$ parameters in Waters' HIBE.

The reuse of public parameters over the different levels of the HIBE complicates the security proof. A straightforward extension of the independence results and lower bound proofs from [19] is not possible. We provide complete proofs of the required results. The constructed HIBE is proved to be secure under chosen plaintext attack (called CPA-secure). Standard techniques [8, 6] can convert such a HIBE into one which is secure against chosen ciphertext attack (CCA-secure).

Related Work: The first construction of HIBE which is secure in the full model is due to Gentry and Silverberg [13]. The security proof depends on the random oracle heuristic. HIBE constructions which can be proved secure without random oracle are known [2, 4]. However, these are secure in the weaker selective-ID model. A generic transformation converts a selective-ID secure HIBE to a HIBE secure in the full model. Unfortunately, this results in an unacceptable degradation in the security bound. It is also possible to convert it into a HIBE secure in the full model *under* the random oracle hypothesis. As mentioned earlier, Waters [19] HIBE is the only previous indication of directly obtaining a HIBE which is secure in the full model without random oracle. In Table 1 of Section 4, we provide a comparison of our construction with the previous constructions.

An extension of Waters' IBE was independently done by Chatterjee-Sarkar [9] and Naccache [16]. In this extension, the n-bit identities of Waters' IBE are replaced by l strings of length n/l bits each. This reduces the number of public parameters from $3+n$ in Waters' IBE to $3+l$. The trade-off is a further security degradation by a factor of approximately $2^{n/l}$. This can be translated into a trade-off between the size of the public parameters and the efficiency of the protocol (see [9]). The CSN idea of extending Waters' IBE can also be applied to the HIBE we describe.

2 Definitions

In this section, we describe HIBE, security model for HIBE, cryptographic bilinear map and the hardness assumption that will be required in the proof.

2.1 HIBE Protocol

Following [14, 13] a HIBE scheme is specified by four probabilistic algorithms: Setup, Key Generation, Encryption and Decryption. Note that, 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 $1 \leq j \leq h$.

Setup: It takes as input a security parameter and returns the system parameters together with the master key. The system parameters include the public parameters of the PKG, a description of the message space, the ciphertext space and the identity space. These are publicly known while the master key is known only to the PKG.

Each of the algorithms below (Key Generation, Encryption and Decryption) have the system public parameters as an input. We do not mention this explicitly.

Key Generation: It takes as input an identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$, the public parameters of the PKG and the private key $d_{\mathsf{v}|(j-1)}$ corresponding to the identity $(\mathsf{v}_1, \ldots, \mathsf{v}_{j-1})$ and returns a private key d_{v} for v. The identity v is used as the public key while d_{v} is the corresponding private key. If $j = 1$, then the private key is generated by the PKG. It is not difficult to see that any entity which possesses a private key for a prefix of v can generate a private key for v.

Encryption: It takes as input the identity v, the public parameters of the PKG and a message from the message space and produces a ciphertext in the ciphertext space.

Decryption: It takes as input the ciphertext and the private key of the corresponding identity v and returns the message or bad if the ciphertext is not valid.

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 b is chosen and the adversary is provided with C^* which is an encryption of M_b 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 v^* or any of its prefix.

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

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

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 restrict 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 $\mathsf{Adv}^{\mathsf{HIBE}}(t, q) \leq \epsilon$.

As mentioned earlier there are generic techniques [8, 6] for converting a CPA-secure HIBE into a CCA-secure HIBE. In view of these techniques, we will concentrate only on CPA-secure HIBE.

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 $e : G_1 \times G_1 \rightarrow G_2$ 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 [5] and the modified Tate pairing [1, 11] are examples of cryptographic bilinear maps.

Note: 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 [1, 11], it is customary to write G_1 additively and G_2 multiplicatively. On the other hand, some "pure" protocol papers [2, 3, 19] write both G_1 and G_2 multiplicatively though this is not true for the initial protocol papers [15, 5]. Here we follow the first convention as it is closer to the known examples of cryptographic bilinear map.

The decisional bilinear Diffie-Hellman (DBDH) problem in $\langle G_1, G_2, e \rangle$ [5] 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

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

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 $\mathsf{Adv}^{\mathsf{DBDH}}(t)$ denotes the maximum of $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DBDH}}$ where the maximum is taken over all adversaries $\mathcal{B}$ running in time at most t. By the (ϵ, t)-DBDH assumption we mean $\mathsf{Adv}^{\mathsf{DBDH}}(t) \leq \epsilon$.

3 HIBE Construction

The IBE scheme proposed in [19] has some similarities with the 1-level (H)IBE scheme of Boneh-Boyen [2]. Waters in his paper [19], utilized this similarity to build a HIBE in an obvious manner, i.e., for each level we have to generate new parameters. This makes the public parameters quite large – for a HIBE of height h with n-bit identities, the number of public parameters becomes $n \times h$.

Here we present an alternative construction where the public parameters can be significantly reduced. We show that for an h-HIBE it suffices to store $(n + h)$ elements in the public parameter.

The identities are of the type $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$, for $j \in \{1, \ldots, h\}$ where each $\mathsf{v}_k = (\mathsf{v}_{k,1}, \ldots, \mathsf{v}_{k,n})$, $\mathsf{v}_{k,j} \in \{0, 1\}$ for $1 \leq j \leq n$.

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 2.3. The message space is G_2.

Set-Up: The protocol is built from groups G_1, G_2 and a bilinear map e as mentioned above. The public parameters are the following elements: P, $P_1 = \alpha P$, P_2, $U'_1, \ldots, U'_h$, $U_1, \ldots, U_n$, where $G_1 = \langle P \rangle$, α is chosen randomly from $\mathbb{Z}_p$ and the other quantities are chosen randomly from G_1. The master secret is αP_2. (The quantities P_1 and P_2 are not directly required; instead $e(P_1, P_2)$ is required. Hence one may store $e(P_1, P_2)$ as part of the public parameters instead of P_1 and P_2.)

Note that for the jth level of the HIBE, we add a single element, i.e., U'_j in the public parameter while the elements $U_1, \ldots, U_n$ are re-used for each level. This way we are able to shorten the public parameter size.

A shorthand: Let $v = (v_1, \ldots, v_n)$, where each v_i is a bit. For $1 \leq k \leq h$ we define,

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

When v is clear from the context we will write V_k instead of $V_k(v)$. The modularity introduced by this notation allows an easier understanding of the protocol.

Key Generation: Let $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$, $j \leq h$, be the identity for which the private key is required. The private key d_{v} for v is defined to be a tuple $d_{\mathsf{v}} = (d_0, d_1, \ldots, d_j)$ where

$$d_0 = \alpha P_2 + \sum_{k=1}^{j} r_k V_k(\mathsf{v}_k); \text{ and } d_k = r_k P \text{ for } 1 \leq k \leq j.$$

Here $r_1, \ldots, r_j$ are random elements from $\mathbb{Z}_p$.

Such a key can be generated by an entity which possesses a private key for the tuple $(\mathsf{v}_1, \ldots, \mathsf{v}_{j-1})$ in the manner shown in [2]. Suppose $(d'_0, d'_1, \ldots, d'_{j-1})$ is a private key for the identity $(\mathsf{v}_1, \ldots, \mathsf{v}_{j-1})$. To generate a private key for v, first choose a random $r_j \in \mathbb{Z}_p$ and compute $d_{\mathsf{v}} = (d_0, d_1, \ldots, d_j)$ as follows.

$$d_0 = d'_0 + r_j V_j(\mathsf{v}_j); \; d_i = d'_i \text{ for } 1 \leq i \leq j-1; \text{ and } d_j = r_j P.$$

In fact, any prefix of v as well as the PKG can generate a private key d_{v} for v.

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

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

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 v. The decryption steps are as follows.

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 bad, else proceed with further decryption as follows. Compute $V_1(\mathsf{v}_1), \ldots, V_j(\mathsf{v}_j)$. Return

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

It is standard to verify the consistency of decryption.

Chatterjee-Sarkar-Naccache Extension: Following [9, 16], let l be a size parameter which divides n. An identity is a tuple $(\mathsf{v}_1, \ldots, \mathsf{v}_j)$, $j \leq h$, where each v_k, $1 \leq k \leq j$ is represented as $\mathsf{v}_k = (\mathsf{v}_{k,1}, \ldots, \mathsf{v}_{k,l})$ where $\mathsf{v}_{k,i}$ is an (n/l)-bit string considered to be an element of $\mathbb{Z}_{2^{n/l}}$.

The public parameters are P, P_1, P_2, $U_1, \ldots, U_l$ and $U'_1, \ldots, U'_h$. In this case, we change the definition of $V_k()$ to the following: $V_k(v) = U'_k + \sum_{i=1}^{l} v_i U_i$ where each v_i is a bit string of length n/l. Using this modified definition of $V_k()$ for $1 \leq k \leq h$, the key generation, encryption and decryption algorithms of the HIBE described above can be extended to the Chatterjee-Sarkar-Naccache settings.

4 Security

In this section, we state the result on security and discuss its implications. The proof is given in Section 5.

Theorem 1. *The HIBE protocol described in Section 3 is (ϵ_{hibe}, t, q)-CPA secure assuming that the (t', ϵ_{dbdh})-DBDH assumption holds in $\langle G_1, G_2, e\rangle$, where $\epsilon_{hibe} \leq 2\epsilon_{dbdh}/\lambda$; $t' = t + \chi(\epsilon_{hibe})$ 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;
$\lambda = 1/(2(4q(n+1))^h)$.

We further assume $4q(n+1) < p$.

The last assumption is practical and similar assumptions are also made in [19, 9, 16], though not quite so explicitly. Before proceeding to the proof, we discuss the above result. The main point of the theorem is the bound on ϵ_{hibe}. This is given in terms of λ and in turn in terms of q, n and h.

The reduction is not tight; security degrades by a factor of $4(4q(n+1))^h$. The actual value of degradation depends on the value of q, the number of key extraction queries made by the adversary. A value of q used in earlier analysis is $q = 2^{30}$ [12].

$h = 1$: This implies that the HIBE is actually an IBE. This is the situation originally considered by Waters [19] and $\epsilon_{hibe} \leq 16q(n+1)\epsilon_{dbdh} \leq 32nq\epsilon_{dbdh}$.

$h > 1$**:** This corresponds to a proper HIBE and we obtain $\epsilon_{hibe} \leq 4(4q(n+1))^h \epsilon_{dbdh} \leq 4(8nq)^h \epsilon_{dbdh}$. For $n = 160$ (and $q = 2^{30}$), this amounts to $\epsilon_{hibe} \leq 4(10 \times 2^{37})^h \epsilon_{dbdh}$.

In Table 1, we compare the known HIBE protocols which are secure in the full model. We note that HIBE protocols which are secure in the selective-ID model are also secure in the full model with a security degradation of $\approx 2^{nh}$, where h is the number of levels in the HIBE and n is number of bits in the identity. This degradation is far worse than the protocols in Table 1.

Table 1. Comparison of HIBE Protocols.

Protocol	Hardness Assump.	Rnd. Ora.	Sec. Deg.	Pub. Para. sz (elts. of G_1)	Pvt. Key sz (elts. of G_1)	Cprtxt sz (elts. of G_1)	Pairing Enc.	Pairing Dec.
GS [13]	BDH	Yes	$q_H q^h$	2	j	j	1	j
BB [2]	DBDH	Yes	q_H^h	$h+3$	$j+1$	$j+1$	None	$j+1$
Waters [19]	DBDH	No	$4(8nq)^h$	$(n+1)h+3$	$j+1$	$j+1$	None	$j+1$
Our	DBDH	No	$4(8nq)^h$	$h+n+3$	$j+1$	$j+1$	None	$j+1$

The BB-HIBE in Table 1 is obtained through a generic transformation (as mentioned in [2]) of the selective-ID secure BB-HIBE to a HIBE secure in the full model using random oracle. For the GS-HIBE [13] and BB-HIBE, the parameter q_H stands for the total number of random oracle queries and in general $q_H \approx 2^{60} \gg q$ [12]. The parameter j in the private key size, ciphertext size and the encryption and decryption columns of Table 1 represents the number of levels of the identity on which the operations are performed. The parameter h is the maximum number of levels in the HIBE. The construction in this paper requires $(h+n+3)$ many elements of G_1 as public parameters whereas Waters HIBE requires $(n+1)h+3$ many elements. The security degradation remains the same in both cases.

5 Proof of Theorem 1

The security reduction follows along standard lines and develops on the proof given in [19, 9, 16]. We need to lower bound the probability of the simulator aborting on certain queries and in the challenge stage. The details of obtaining this lower bound are given in Section 5.1. In the following proof, we simply use the lower bound. We want to show that the HIBE is (ϵ_{hibe}, t, q)-CPA secure. In the game sequence style of proofs, we start with the adversarial game defining the CPA-security of the protocol against an adversary $\mathcal{A}$ and then obtain a sequence of games as usual. In each of the games, the simulator chooses a bit δ and the adversary makes a guess δ'. By X_i we will denote the event that the bit δ is equal to the bit δ' in the ith game.

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

Game 1: In this game, we setup the protocol from a tuple $\langle P, P_1 = aP, P_2 = bP, P_3 = cP, Z = e(P_1, P_2)^{abc}\rangle$ and answer key extraction queries and generate the challenge. The simulator is assumed to know the values a, b and c. However, the simulator can setup the protocol as well as answer certain private key queries without the knowledge of these values. Also, for certain challenge identities it can generate the challenge ciphertext without the knowledge of a, b and c. In the following, we show how this can be done. If the simulator cannot answer a key extraction query or generate a challenge without using the knowledge of a, b and c, it sets a flag flg to one. The value of flg is initially set to zero.

Note that the simulator is always able to answer the adversary (with or without using a, b and c). The adversary is provided with proper replies to all its queries and is also provided the proper challenge ciphertext. Thus, irrespective of whether flg is set to one, the adversary's view in Game 1 is same as that in Game 0. Hence, we have $\mathsf{Pr}[X_0] = \mathsf{Pr}[X_1]$.

We next show how to setup the protocol and answer the queries based on the tuple $\langle P, P_1 = aP, P_2 = bP, P_3 = cP, Z = e(P_1, P_2)^{abc}\rangle$.

Set-Up: Let m be a prime such that $2q < m < 4q$. Our choice of m is different from that of previous works [19, 9, 16] where m was chosen to be equal to $4q$ and $2q$.

Choose $x'_1, \ldots, x'_h$ and $x_1, \ldots, x_n$ randomly from $\mathbb{Z}_m$; also choose $y'_1, \ldots, y'_h$ and $y_1, \ldots, y_n$ randomly from $\mathbb{Z}_p$. Choose $k_1, \ldots, k_h$ randomly from $\{0, \ldots, n\}$.

For $1 \leq j \leq h$, define $U'_j = (p - mk_j + x'_j)P_2 + y'_jP$ and for $1 \leq i \leq n$ define $U_i = x_iP_2 + y_iP$. The public parameters are $(P, P_1, P_2, U'_1, \ldots, U'_h, U_1, \ldots, U_n)$. The master secret is $aP_2 = abP$. The distribution of the public parameters is as expected by $\mathcal{A}$. In its attack, $\mathcal{A}$ will make some queries, which have to be properly answered by the simulator.

For $1 \leq j \leq h$, we define several functions. Let $v = (v_1, \ldots, v_n)$ where each $v_i \in \{0, 1\}$. We define

$$\left.\begin{array}{r@{}l} F_j(v) &= p - mk_j + x'_j + \sum_{i=1}^{n} x_iv_i \\ J_j(v) &= y'_j + \sum_{i=1}^{n} y_iv_i \\ L_j(v) &= x'_j + \sum_{i=1}^{n} x_iv_i \pmod{m} \\ K_j(v) &= \begin{cases} 0 \text{ if } L_j(v) = 0 \\ 1 \text{ otherwise.} \end{cases} \end{array}\right\} \qquad (2)$$

Recall that we have assumed $4q(n+1) < p$. Let $F_{\min}$ and $F_{\max}$ be the minimum and maximum values of $F_j(v)$. $F_{\min}$ is achieved when k_j is maximum and x'_j and the x_i's are all zero. Thus, $F_{\min} = p - mn$. We have $mn < 4q(n+1)$ and by assumption $4q(n+1) < p$. Hence, $F_{\min} > 0$. Again $F_{\max}$ is achieved when $k_j = 0$ and x'_j and the x_i's and v_i's are equal to their respective maximum values. We get $F_{\max} < p + m(n+1) < p + 4q(n+1) < 2p$. Thus, we have $0 < F_{\min} \leq F_j(v) \leq F_{\max} < 2p$. Consequently, $F_j(v) \equiv 0 \bmod p$ if and only if $F_j(v) = p$ which holds if and only if $-mk_j + x'_j + \sum_{i=1}^{n} x_iv_i = 0$.

Now we describe how the queries made by $\mathcal{A}$ are answered by $\mathcal{B}$. The queries can be made in both Phases 1 and 2 of the adversarial game (subject to the usual restrictions). The manner in which they are answered by the simulator is the same in both the phases.

Key Extraction Query: Suppose $\mathcal{A}$ makes a key extraction query on the identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_j)$. Suppose there is a u with $1 \leq u \leq j$ such that $K_u(\mathsf{v}_u) = 1$. Otherwise set flg to one. In the second case, the simulator uses the value of a to return a proper private key $d_{\mathsf{v}} = (aP_2 + \sum_{i=1}^{j} r_i V_i, r_1 V_1, \ldots, r_j V_j)$. In the first case, the simulator constructs a private key in the following manner.

Choose random $r_1, \ldots, r_j$ from $\mathbb{Z}_p$ and define

$$\left.\begin{aligned} d_{0|u} &= -\tfrac{J_u(\mathsf{v}_u)}{F_u(\mathsf{v}_u)} P_1 + r_u(F_u(\mathsf{v}_u)P_2 + J_u(\mathsf{v}_u)P) \\ d_u &= \tfrac{-1}{F_u(\mathsf{v}_u)} P_1 + r_u P \\ d_k &= r_k P \text{ for } k \neq u \\ d_{\mathsf{v}} &= (d_{0|u} + \textstyle\sum_{k \in \{1,\ldots,j\} \setminus \{u\}} r_k V_k, d_1, \ldots, d_j) \end{aligned}\right\} \quad (3)$$

The quantity d_{v} is a proper private key corresponding to the identity v. The algebraic verification of this fact is similar to that in [2, 19]. This key is provided to $\mathcal{A}$.

Challenge: Let the challenge identity be $\mathsf{v}^* = (\mathsf{v}_1^*, \ldots, \mathsf{v}_{h^*}^*)$, $1 \leq h^* \leq h$ and the messages be M_0 and M_1. Choose a random bit δ. We need to have $F_k(\mathsf{v}_k^*) \equiv 0 \bmod p$ for all $1 \leq k \leq h^*$. If this condition does not hold, then set flg to one. In the second case, the simulator uses the value of c to provide a proper encryption of M_δ to $\mathcal{A}$ by computing $(M_\delta \times e(P_1, P_2)^c, cP, cV_1, \ldots, cV_{h^*})$. In the first case, it constructs a proper encryption of M_δ in the following manner.

$$(M^\delta \times Z, C_1 = P_3, B_1 = J_1(\mathsf{v}_1^*)P_3, \ldots, B_{h^*} = J_{h^*}(\mathsf{v}_{h^*}^*)P_3).$$

We require B_j to be equal to $cV_j(\mathsf{v}_j^*)$ for $1 \leq j \leq h^*$. Recall that the definition of $V_j(v)$ is $V_j(v) = U_j' + \sum_{k=1}^{n} v_k U_k$. Using the definition of U_j' and the U_k's as defined in the setup by the simulator, we obtain, $cV_i = c(F_i(\mathsf{v}_i^*)P_2 + J_i(\mathsf{v}_i^*)P) = J_i(\mathsf{v}_i^*)cP = J_i(\mathsf{v}_i^*)P_3$. Here we use the fact, $F_i(\mathsf{v}_i^*) \equiv 0 \bmod p$. Hence, the quantities $B_1, \ldots, B_{h^*}$ are properly formed.

Guess: The adversary outputs a guess δ' of δ.

Game 2: This is a modification of Game 1 whereby the Z in Game 1 is now chosen to be a random element of G_2. This Z is used to mask the message M_δ in the challenge ciphertext. Since Z is random, the first component of the challenge ciphertext is a random element of G_2 and provides no information to the adversary about δ. Thus, $\mathsf{Pr}[X_2] = \frac{1}{2}$.

We have the following claim.

Claim:

$$|\mathsf{Pr}[X_1] - \mathsf{Pr}[X_2]| \leq \frac{\epsilon_{dbdh}}{\lambda} + \frac{\epsilon_{hibe}}{2}.$$

Proof: The change from Game 1 to Game 2 corresponds to an "indistinguishability" step in Shoup's tutorial [18] on such games. Usually, it is easy to bound the probability difference. In this case, the situation is complicated by the fact that there is a need to abort.

We show that it is possible to obtain an algorithm $\mathcal{B}$ for DBDH by extending Games 1 and 2. The extension of both the games is same and is described as follows. $\mathcal{B}$ takes as input a tuple (P, aP, bP, cP, Z) and sets up the HIBE protocol as in Game 1 (The setup of Games 1 and 2 are the same). The key extraction queries are answered and the challenge ciphertext is generated as in Game 1. If at any point of time flg is set to one by the game, then $\mathcal{B}$ outputs a random bit and aborts. This is because the query cannot be answered or the challenge ciphertext cannot be generated using the input tuple. At the end of the game, the adversary outputs the guess δ'. $\mathcal{B}$ now goes through a separate abort stage as follows.

"Artificial Abort": The probability that $\mathcal{B}$ aborts in the query or challenge phases depends on the adversary's input. The goal of the artificial abort step is to make the probability of abort independent of the adversary's queries by ensuring that in all cases its probability of abort is the maximum possible. This is done by sampling the transcript of adversary's query and in certain cases aborting. The sampling procedure introduces the extra component $O(\epsilon_{hibe}^{-2}\ln(\epsilon_{hibe}^{-1})\lambda^{-1}\ln(\lambda^{-1}))$ into the simulator's runtime. (For details see [19, 16].) Here λ is a lower bound on the probability that $\mathcal{B}$ does not abort before entering the artificial abort stage. The expression for λ is obtained in Proposition 3 of Section 5.1.

Output: If $\mathcal{B}$ has not aborted up to this stage, then it outputs 1 if $\delta = \delta'$; else 0.

Note that if Z is real, then the adversary is playing Game 1 and if Z is random, then the adversary is playing Game 2. The time taken by the simulator in either Game 1 or 2 is clearly $t + \chi(\epsilon_{hibe})$. From this point, standard inequalities and probability calculations establish the claim. □

Now we can complete the proof in the following manner.

$$\begin{aligned}
\epsilon_{hibe} &= \left|\mathsf{Pr}[X_0] - \frac{1}{2}\right| \\
&\leq |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_2]| \\
&\leq |\mathsf{Pr}[X_0] - \mathsf{Pr}[X_1]| + |\mathsf{Pr}[X_1] - \mathsf{Pr}[X_2]| \\
&\leq \frac{\epsilon_{hibe}}{2} + \frac{\epsilon_{dbdh}}{\lambda}.
\end{aligned}$$

Rearranging the inequality gives the desired result. This completes the proof of Theorem 1. □

5.1 Lower Bound on Not Abort

We require the following two independence results in obtaining the required lower bound. Similar independence results have been used in [19, 9, 16] in connection with IBE protocols. The situation for HIBE is more complicated than IBE and especially so since we reuse some of the public parameters over different levels of the HIBE. This makes the proofs more difficult. Our independence results are given in Proposition 1 and 2 and these subsume the results of previous work. We provide complete proofs for these two propositions as well as a complete proof for the lower bound. The probability calculation for the lower bound is also more complicated compared to the IBE case.

Proposition 1. *Let m be a prime and $L(\cdot)$ be as defined in (2). Let $\mathsf{v}_1, \ldots, \mathsf{v}_j$ be identities, i.e., each $\mathsf{v}_i = (\mathsf{v}_{i,1}, \ldots, \mathsf{v}_{i,n})$, is an n-bit string. Then*

$$\mathsf{Pr}\left[\bigwedge_{k=1}^{j} (L_k(\mathsf{v}_k) = 0)\right] = \frac{1}{m^j}.$$

The probability is over independent and uniform random choices of $x'_1, \ldots, x'_j$, $x_1, \ldots, x_n$ from $\mathbb{Z}_m$. Consequently, for any $\theta \in \{1, \ldots, j\}$, we have

$$\mathsf{Pr}\left[L_\theta(\mathsf{v}_\theta) = 0 \,\middle|\, \bigwedge_{k=1, k\neq\theta}^{j} (L_k(\mathsf{v}_k) = 0)\right] = \frac{1}{m}.$$

Proof: Since $\mathbb{Z}_m$ forms a field, we can do linear algebra with vector spaces over $\mathbb{Z}_m$. The condition $\bigwedge_{k=1}^{j} (L_j(\mathsf{v}_j) = 0)$ is equivalent to the following system of equations over $\mathbb{Z}_m$.

$$\begin{array}{cccccc}
x'_1 & + \mathsf{v}_{1,1}x_1 & + \cdots + & \mathsf{v}_{1,n}x_n & = 0 \\
x'_2 & + \mathsf{v}_{2,1}x_1 & + \cdots + & \mathsf{v}_{2,n}x_n & = 0 \\
\cdots . & \cdots & . \cdots . & \cdots & . \ . \\
x'_j & + \mathsf{v}_{j,1}x_1 & + \cdots + & \mathsf{v}_{j,n}x_n & = 0
\end{array}$$

This can be rewritten as

$$(x'_1, \ldots, x'_j, x_1, \ldots, x_n)A_{(j+n)\times(j+n)} = (0, \ldots, 0)_{1\times(j+n)}$$

where

$$A = \begin{bmatrix} I_j & O_{j\times n} \\ \mathsf{V}_{n\times j} & O_{n\times n} \end{bmatrix} \text{ and } \mathsf{V}_{n\times j} = \begin{bmatrix} \mathsf{v}_{1,1} & \cdots & \mathsf{v}_{j,1} \\ \cdots & \cdots & \cdots \\ \mathsf{v}_{1,n} & \cdots & \mathsf{v}_{j,n} \end{bmatrix};$$

I_j is the identity matrix of order j; O is the all zero matrix of the specified order. The rank of A is clearly j and hence the dimension of the solution space is n. Hence, there are m^n solutions in $(x'_1, \ldots, x'_j, x_1, \ldots, x_n)$ to the above system of linear equations. Since the variables $x'_1, \ldots, x'_j, x_1, \ldots, x_n$ are chosen independently and uniformly at random, the probability that the system of linear equations is satisfied for a particular choice of these variables is $m^n/m^{n+j} = 1/m^j$. This proves the first part of the result.

For the second part, note that we may assume $\theta = j$ by renaming the x''s if required. Then

$$\mathsf{Pr}\left[L_j(\mathsf{v}_j) = 0 \,\middle|\, \bigwedge_{k=1}^{j-1} (L_k(\mathsf{v}_k) = 0)\right] = \frac{\mathsf{Pr}\left[\bigwedge_{k=1}^{j} (L_k(\mathsf{v}_k) = 0)\right]}{\mathsf{Pr}\left[\bigwedge_{k=1}^{j-1} (L_k(\mathsf{v}_k) = 0)\right]} = \frac{m^{j-1}}{m^j} = \frac{1}{m}.$$

□

Proposition 2. *Let m be a prime and $L(\cdot)$ be as defined in (2). Let $\mathsf{v}_1, \ldots, \mathsf{v}_j$ be identities, i.e., each $\mathsf{v}_i = (\mathsf{v}_{i,1}, \ldots, \mathsf{v}_{i,n})$, is an n-bit string. Let $\theta \in \{1, \ldots, j\}$ and let v'_θ be an identity such that $\mathsf{v}'_\theta \neq \mathsf{v}_\theta$. Then*

$$\mathsf{Pr}\left[(L_\theta(\mathsf{v}'_\theta) = 0) \wedge \bigwedge_{k=1}^{j} (L_k(\mathsf{v}_k) = 0)\right] = \frac{1}{m^{j+1}}.$$

The probability is over independent and uniform random choices of $x'_1, \ldots, x'_j$, $x_1, \ldots, x_n$ from $\mathbb{Z}_m$. Consequently, we have

$$\mathsf{Pr}\left[L_\theta(\mathsf{v}'_\theta) = 0 \,\middle|\, \bigwedge_{k=1}^{j} (L_k(\mathsf{v}_k) = 0)\right] = \frac{1}{m}.$$

Proof: The proof is similar to the proof of Proposition 1. Without loss of generality, we may assume that $\theta = j$, since otherwise we may rename variables to achieve this. The condition $(L_\theta(\mathsf{v}'_\theta) = 0) \wedge \bigwedge_{k=1}^{j} (L_k(\mathsf{v}_k) = 0)$ is equivalent to a system of linear equations $xA = 0$ over $\mathbb{Z}_m$. In this case, the form of A is the following.

$$A = \begin{bmatrix} I_j & c^T & O_{j\times n} \\ V_{n\times j} & (\mathsf{v}'_j)^T & O_{n\times} \end{bmatrix}$$

where $c = (0, \ldots, 0, 1)$; c^T denotes the transpose of c and $(\mathsf{v}'_j)^T$ is the transpose of v'_j. The first j columns of A are linearly independent. The $(j+1)$th column of A is clearly linearly independent of the first $(j-1)$ columns. We have $\mathsf{v}_j \neq \mathsf{v}'_j$ and $m > 2$, hence $\mathsf{v}_j \not\equiv \mathsf{v}'_j \bmod m$. Using this, it is not difficult to see that the first $(j+1)$ columns of A are linearly independent and hence the rank of A is $(j+1)$. Consequently, the dimension of the solution space is $n-1$ and there are m^{n-1} solutions in $(x'_1, \ldots, x'_j, x_1, \ldots, x_n)$ to the system of linear equations. Since the x''s and the x's are chosen independently and uniformly at random from $\mathbb{Z}_m$, the probability of getting a solution is $m^{n-1}/m^{n+j} = 1/m^{j+1}$. This proves the first part of the result. The proof of the second part is similar to that of Proposition 1. □

Proposition 3. *The probability that the simulator in the proof of Theorem 1 does not abort before the artificial abort stage is at least $\lambda = \frac{1}{2(4q(n+1))^h}$.*

Proof: We consider the simulator in the proof of Theorem 1. Up to the artificial abort stage, the simulator could abort on either a key extraction query or in the challenge stage. Let abort be the event that the simulator aborts before the artificial abort stage. For $1 \leq i \leq q$, let E_i denote the event that the simulator does not abort on the ith key extraction query and let C be the event that the simulator does not abort in the challenge stage. We have

$$\begin{aligned}
\mathsf{Pr}[\overline{\mathsf{abort}}] &= \mathsf{Pr}\left[\left(\bigwedge_{i=1}^{q} E_i\right) \wedge C\right] \\
&= \mathsf{Pr}\left[\left(\bigwedge_{i=1}^{q} E_i\right) |C\right] \mathsf{Pr}[C] \\
&= \left(1 - \mathsf{Pr}\left[\left(\bigvee_{i=1}^{q} \neg E_i\right) |C\right]\right) \mathsf{Pr}[C] \\
&\geq \left(1 - \sum_{i=1}^{q} \mathsf{Pr}\left[\neg E_i \,|C\right]\right) \mathsf{Pr}[C].
\end{aligned}$$

We first consider the event C. Let the challenge identity be $\mathsf{v}^* = (\mathsf{v}_1^*, \ldots, \mathsf{v}_{h^*}^*)$. Event C holds if and only if $F_j(\mathsf{v}_j^*) \equiv 0 \bmod p$ for $1 \leq j \leq h^*$. Recall that by choice of p, we can assume $F_j(\mathsf{v}_j^*) \equiv 0 \bmod p$ if and only if $x'_j + \sum_{k=1}^{n} x_k \mathsf{v}_{j,k} = mk_j$. Hence,

$$\mathsf{Pr}[C] = \mathsf{Pr}\left[\bigwedge_{j=1}^{h^*}\left(x'_j + \sum_{k=1}^{n} x_k \mathsf{v}_{j,k} = mk_j\right)\right]. \tag{4}$$

For $1 \leq j \leq h^*$ and $0 \leq i \leq n$, denote the event $x'_j + \sum_{k=1}^{n} x_k \mathsf{v}_{j,k} = m_i$ by $A_{j,i}$ and the event $k_j = i$ by $B_{j,i}$. Also, let $C_{j,i}$ be the event $A_{j,i} \wedge B_{j,i}$.

Note that the event $\bigvee_{i=0}^{n} A_{j,i}$ is equivalent to $x'_j + \sum_{k=1}^{n} x_k \mathsf{v}_{j,k} \equiv 0 \bmod m$ and hence equivalent to the condition $L_j(\mathsf{v}_j) = 0$. Since k_j is chosen uniformly at random from the set $\{0, \ldots, n\}$, we have $\mathsf{Pr}[B_{j,i}] = 1/(1+n)$ for all j and i. The events $B_{j,i}$'s are independent of each other and also independent of the $A_{j,i}$'s. We have

$$\begin{aligned}
\mathsf{Pr}\left[\bigwedge_{j=1}^{h^*}\left(x'_j + \sum_{k=1}^{n} x_k \mathsf{v}_{j,k} = mk_j\right)\right] &= \mathsf{Pr}\left[\bigwedge_{j=1}^{h^*}\left(\bigvee_{i=0}^{n} C_{j,i}\right)\right] \\
&= \frac{1}{(1+n)^{h^*}} \mathsf{Pr}\left[\bigwedge_{j=1}^{h^*}\left(\bigvee_{i=0}^{n} A_{j,i}\right)\right] \\
&= \frac{1}{(1+n)^{h^*}} \mathsf{Pr}\left[\bigwedge_{j=1}^{h^*} (L_j(\mathsf{v}_j) = 0)\right] \\
&= \frac{1}{(m(1+n))^{h^*}}
\end{aligned}$$

The last equality follows from Proposition 1.

Now we turn to bounding $\mathsf{Pr}[\neg E_i|C]$. For simplicity of notation, we will drop the subscript i from E_i and consider the event E that the simulator does not abort on a particular key extraction query on an identity $(\mathsf{v}_1,\ldots,\mathsf{v}_j)$. By the simulation, the event $\neg E$ implies that $L_i(\mathsf{v}_i)=0$ for all $1\leq i\leq j$. This holds even when the event is conditioned under C. Thus, we have $\mathsf{Pr}[\neg E|C]\leq \mathsf{Pr}[\wedge_{i=1}^{j} L_i(\mathsf{v}_i)=0|C]$. The number of components in the challenge identity is h^* and now two cases can happen:

$j\leq h^*$: By the protocol constraint (a prefix of the challenge identity cannot be queried to the key extraction oracle), we must have a θ with $1\leq\theta\leq j$ such that $\mathsf{v}_\theta\neq\mathsf{v}_\theta^*$.

$j>h^*$: In this case, we choose $\theta=h^*+1$.

Now we have $\mathsf{Pr}[\neg E|C]\leq \mathsf{Pr}\left[\bigwedge_{i=1}^{j} L_i(\mathsf{v}_i)=0|C\right]\leq \mathsf{Pr}[L_\theta(\mathsf{v}_\theta)=0|C]=1/m.$

The last equality follows from an application of either Proposition 1 or Proposition 2 according as whether $j>h^*$ or $j\leq h^*$. Substituting this in the bound for $\mathsf{Pr}[\overline{\mathsf{abort}}]$ we obtain

$$\begin{aligned}\mathsf{Pr}[\overline{\mathsf{abort}}] &\geq \left(1-\sum_{i=1}^{q}\mathsf{Pr}\left[\neg E_i\,|C\right]\right)\mathsf{Pr}[C].\\ &\geq \left(1-\frac{q}{m}\right)\frac{1}{(m(n+1))^h}\geq\frac{1}{2}\times\frac{1}{(4q(n+1))^h}.\end{aligned}$$

We use $h\geq h^*$ and $2q<m<4q$ to obtain the inequalities. This completes the proof. □

6 Conclusion

Waters presented a construction of IBE [19] which significantly improves upon the previous construction of Boneh-Boyen [3]. In his paper, Waters also described a method to extend his IBE to a HIBE. The problem with this construction is that it increases the number public parameters. In this paper, we have presented a construction of a HIBE which builds upon the previous (H)IBE protocols. The number of public parameters is significantly less compared to Waters' HIBE. The main open problem in the construction of HIBE protocols is to avoid or control the security degradation which is exponential in the number of levels of the HIBE.

References

1. 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.
2. Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Cachin and Camenisch [7], pages 223–238.

3. Dan Boneh and Xavier Boyen. Secure Identity Based Encryption Without Random Oracles. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, 2004.
4. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Cramer [10], pages 440–456. Full version available at Cryptology ePrint Archive; Report 2005/015.
5. 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.
6. 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.
7. 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.
8. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In Cachin and Camenisch [7], pages 207–222.
9. 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.
10. 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.
11. 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.
12. David Galindo. Boneh-Franklin Identity Based Encryption Revisited. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 791–802. Springer, 2005.
13. 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.
14. 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.
15. Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004. Earlier version appeared in the proceedings of ANTS-IV.
16. David Naccache. Secure and Practical Identity-Based Encryption. Cryptology ePrint Archive, Report 2005/369, 2005. http://eprint.iacr.org/.
17. 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.
18. Victor Shoup. Sequences of Games: a Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/.
19. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [10], pages 114–127.