A New Practical Identity-Based Encryption System

Jong Hwan Park* Dong Hoon Lee†

Abstract

We present a new practical Identity-Based Encryption (IBE) system that can be another candidate for standard IBE techniques. Our construction is based on a new framework for realizing an IBE trapdoor from pairing-based groups, which is motivated from the 'two equation' revocation technique suggested by Lewko, Sahai, and Waters. The new framework enables our IBE system to achieve a tight security reduction to the Decision Bilinear Diffie-Hellman assumption. Due to its the tightness, our system can take as input the shorter size of security parameters than the previous practical BF, SK, and BB_1 systems, which provides better efficiency to our system in terms of computational cost. With appropriate parametrization at 80-bit security level (considering security loss), our IBE system can obtain 11 times faster decryption than the previous ones and 77 times faster encryption than the BF system. We prove that our system is fully secure against chosen ciphertext attacks in the random oracle model. From computational variant of Naor's observation, we can also suggest a new signature scheme that features a tight security reduction to the Computational Diffie-Hellman assumption and provides strong unforgeability simultaneously.

Keywords: Identity-based encryption, Bilinear maps.

1 Introduction

Identity-Based Encryption (IBE) [48] is a special type of public key encryption where a public key can be any string that carries its own meaning to a user's identity, such as an e-mail address. As such a meaningful string can be naturally associated with a user, an IBE system does not need a certifying mechanism to ensure that a public key (as the meaningful string) is bound to a user (as the owner of the public key). As opposed to an IBE system, a traditional public key encryption system needs the certifying mechanism to securely distribute public keys, and indeed it must run on a complex architecture called 'Public Key Infrastructure'.

By virtue of the advantage over the public key encryption, IBE had received considerable interest to cryptographic researchers since Shamir [48] posed the initial question about the existence of such an IBE system. In 2001, Boneh and Franklin [15] proposed the first practical IBE system based on groups with efficiently computable bilinear maps (i.e., paring), along with a formal security definition of IBE. Since then, a large body of work [46, 24, 25, 10, 49, 32, 41, 42, 50] has been devoted to constructing pairing-based IBE systems to improve in terms of security and efficiency. Among the previous IBE systems, three of them have been perceived as practical constructions, which are works by Boneh-Franklin [15], Sakai-Kasahara [46, 24, 25], and Boneh-Boyen[1] [10], and thereafter they have been submitted to the IEEE P1363.3 standard for "Identity-Based Cryptographic Techniques using Pairings".

*Korea University, Seoul, Korea. Email: `decartian@korea.ac.kr`
†Korea University, Seoul, Korea. Email: `donghlee@korea.ac.kr`

[1]This is denoted as 'BB_1'.

1.1 Our Contribution

In this work we present a new practical IBE system that can be another candidate for standard IBE techniques. Our IBE system results from a new framework for realizing the IBE trapdoor, which is motivated by the 'two equation' technique recently suggested by Lewko, Sahai, and Waters [43]. One notable advantage of the new framework is that our construction is also pairing-based like BF, SK, and BB_1 systems, yet it has a *tight* security reduction to the Decision Bilinear Diffie-Hellman (DBDH) assumption. In order to encrypt arbitrary-length messages, we also suggest a new Identity-Based Key Encapsulation Mechanism (IBKEM) combined with one-time symmetric-key encryption. Our IBE systems are all proven to be fully secure against chosen ciphertext attacks in the random oracle model. In particular, one-time symmetric-key encryption secure against passive attacks is sufficient for the latter IBE system without the need of the 'encrypt-then-MAC' or 'authenticated symmetric encryption' paradigm.

Prior to our result, none of the three practical BF, SK, and BB_1 systems provided tightness in their respective security analysis, and in fact there existed significant security gaps between security assumptions and their IBE systems. One might wonder what the benefit from such tightness in security reduction is. The benefit is that we can achieve security of our system straightforwardly from that of the underlying DBDH assumption at the same security level. This means that if we want to instantiate our IBE system at current 80-bit security level, we can use a DBDH-hard group *at the same security level*. However, this is not the case in BF, SK, and BB_1 systems where security is loosely reduced to each security assumption by a factor of (at least) 2^{50} if we consider a reasonable number of adversarial hash queries as 2^{50}. The loose security reduction forces us to choose a larger security parameter (regarding DBDH-hard groups) than the 80-bit one even if we want to instantiate them at the 80-bit security level. Importantly, the larger security parameter tends to have an unfavorable effect on computational cost of resulting IBE systems [19]. For instance, when comparing BF, SK, and BB_1 systems at 128-bit security level with our system at 80-bit security level, ours has about (at least) 11 times faster decryption than the three systems, and about 77 times faster encryption than the BF system. To add credence to this result, we give more concrete comparison results in terms of security reduction and efficiency in Section 5.

From variant of Naor's observation (stated in [15]), our new framework gives rise to a new public-key signature scheme whose security relies on the Computational Diffie-Hellman (CDH) assumption in the random oracle model. Two favorable features of our signature scheme are that (1) it has a tight security reduction to the CDH assumption and (2) it is secure in the sense of strong unforgeability. BF, SK, and BB_1 systems also gave rise to signature schemes [17, 11, 49] derived from each one, but none of them have the two properties at the same time. Until now, it has been known that the Katz-Wang [36, 40] technique (combined with BF key generation framework) is only the one for achieving the two goals, but it is worth mentioning that our result can give an alternative method.

1.2 Overview of Our New Technique

BF, SK, and BB_1 systems have their unique frameworks to realize IBE trapdoors from paring-based groups, respectively. Following Boyen's naming in [19], each framework is called 'full-domain-hash' (for BF), 'exponent-inversion' (for SK), and 'commutative-blinding' (for BB_1). Each framework determines both the distinct structure of a private key and different kinds of security assumptions. Also, most of the subsequent paring-based IBE systems fall into one of the three paradigms.

To build our new IBE system, we also come up with a new framework for realizing the IBE trapdoor. As we mentioned before, our framework is motivated by the two equation technique of Lewko, Sahai, and Waters [43]. Roughly speaking, the original LSW technique is to use private key elements $(g^r, (g_1 u^{\mathsf{ID}})^r)$ and ciphertext elements $(g^s, (g_1 u^{\mathsf{ID}'})^s)$ to compute a pairing value $e(g,u)^{sr}$. Here, g, g_1, and u are from public parameters. The value $e(g,u)^{sr}$ is then used to recover a message blinding factor $e(g,g)^{\alpha s}$ by pairing g^s with an additional key element $g^\alpha u^r$. The point is that such a pairing value can be obtained only if $\mathsf{ID} \neq \mathsf{ID}'$, and this gives the revocation system [43] where only users whose identities are different from ID' are able to compute the pairing value. On the other hand, our two equation technique is slightly changed in a way that computing the value $e(g,u)^{sr}$ is possible only if $\mathsf{ID} = \mathsf{ID}'$. This can be done by setting private key elements as $(g^r, (\mathcal{H}(\mathsf{ID})u^{\mathrm{tag}_k})^r)$ and ciphertext elements as $(g^s, (\mathcal{H}(\mathsf{ID}')u^{\mathrm{tag}_c})^s)$, where $\mathcal{H}$ is a cryptographic hash function and (as we explain below) the probability that $\mathrm{tag}_k = \mathrm{tag}_c$ is negligible.

As in the BF system, our framework requires a cryptographic hash function $\mathcal{H}$ that maps an identity string $\mathsf{ID} \in \{0,1\}^*$ to a group element $\mathcal{H}(\mathsf{ID})$, but unlike the BF system a private key for an identity ID is not uniquely determined. A private key $\mathsf{sk}_{\mathsf{ID}}$ consists of three groups $(g^\alpha u^r,\ g^r,\ (\mathcal{H}(\mathsf{ID})u^{\mathrm{tag}_k})^r)$, which differs by a randomly-chosen exponent r in $\mathbb{Z}_p$. Here, α is the master secret key known only to a key generation center. At this moment, one may wonder how the value tag_k is decided. Indeed, we have that $\mathrm{tag}_k = h(g^\alpha u^r, g^r) \in \mathbb{Z}_p$ by introducing another cryptographic hash function h. Similarly, when encrypting a message M, a ciphertext under ID is constructed as $\big(Me(g,g)^{\alpha s},\ g^s,\ (\mathcal{H}(\mathsf{ID})u^{\mathrm{tag}_c})^s\big)$ using the hash value $\mathrm{tag}_c = h(Me(g,g)^{\alpha s}, g^s)$. In case of our IBKEM, an arbitrary length message M is encrypted as $\big(\mathcal{E}_K(M),\ g^s,\ (\mathcal{H}(\mathsf{ID})u^{\mathrm{tag}_c})^s\big)$ using a one-time symmetric-key encryption algorithm $\mathcal{E}$, where $K = e(g,g)^{\alpha s}$ and $\mathrm{tag}_c = h(\mathcal{E}_K(M), g^s)$. If we use a collision-resistant hash function h, the correctness error caused by the equality $\mathrm{tag}_k = \mathrm{tag}_c$ becomes acceptable in practice. Another characteristic of our framework is to use the hash function h to protect the ciphertext element $Me(g,g)^{\alpha s}$ or $\mathcal{E}_K(M)$ related to M. Indeed, the distinct usage of h enables our system to directly obtain chosen ciphertext security without resorting to other methods such as 'encrypt-then-MAC' or 'authenticated symmetric encryption'.

We now explain how our IBE system can achieve a tight security reduction under the DBDH assumption. In our security proofs, the two hash functions $\mathcal{H}$ and h are modeled as random oracles. Somewhat surprisingly, being able to use two hash functions in generating one group element enables our reduction algorithm to generate private keys for *all* identities and use *any* identity as a challenge identity ID^*. Nevertheless, a private key for ID^* is not helpful to decrypt the challenge ciphertext (that can be constructed under ID^*), which is necessary for solving the so-called 'self-decryption' paradox. Notice that similar reductions can be found in [32, 50] that provided full security without random oracles. Let (g, g^a, g^b, g^c, T) be a DBDH instance. Given an identity ID_i, the random oracle $\mathcal{H}$ outputs $\mathcal{H}(\mathsf{ID}_i) = (g^a)^{\gamma_i} g^{\pi_i}$ for two randomly-chosen exponents γ_i and π_i in $\mathbb{Z}_p$. The important point is that the value γ_i per each identity can be information-theoretically hidden from an adversary's view and later used for an output value of another random oracle h. When creating a private key for ID_i, our reduction algorithm is able to generate the key as $\big(g^\alpha u^{\tilde{r}},\ g^{\tilde{r}},\ (\mathcal{H}(\mathsf{ID}_i)u^{\mathrm{tag}_k})^{\tilde{r}}\big)$ by setting $\mathrm{tag}_k = h(g^\alpha u^{\tilde{r}}, g^{\tilde{r}}) = \gamma_i$ and $\tilde{r} = b + r$ for a random r in $\mathbb{Z}_p$. The validity of the private key is checked under the condition that $\alpha = ab$ and $u = g^{-a} g^\delta$ for a random $\delta \in \mathbb{Z}_p$. A similar manipulation is taken when generating the challenge ciphertext under ID^*. As $\mathsf{sk}_{\mathsf{ID}^*}$ must not be asked, the value γ^* embedded into $\mathcal{H}(\mathsf{ID}^*)$ will be hidden until the challenge phase (with overwhelming probability) and thus can be reserved for setting $\mathrm{tag}_c = \gamma^*$ (as well as $s = c$ for the DBDH problem). In case when trying to decrypt the challenge ciphertext using $\mathsf{sk}_{\mathsf{ID}^*}$, it should be the case that $\mathrm{tag}_k = \gamma^*$ that was already embedded into $\mathcal{H}(\mathsf{ID}^*)$. Therefore, the decryption is not possible because $\mathrm{tag}_k = \mathrm{tag}_c$, and this explains how the self-decryption paradox can be solved.

To achieve the chosen ciphertext security, our reduction algorithm needs to deal with adversarial decryption queries. In our security analysis, this is not a big problem as private keys for all identities can be generated and ill-formed ciphertexts are detected via consistency check using pairing. The only problem is that in the event that $\mathrm{tag}_k = \mathrm{tag}_c$ happens, normal decryption cannot be performed. However, as an output of h as a random oracle is determined by choosing a random value in $\mathbb{Z}_p$ and p is exponentially large (e.g., p is a 160-bit prime), our reduction can avoid such a troublesome case in all decryption queries with overwhelming probability.

1.3 Related Work

Boneh and Franklin [15] presented the first practical IBE system based on groups with efficiently computable pairings and defined the formal security notion for IBE known as full security against chosen ciphertext attacks. Most of the subsequent IBE systems followed the notion depending on different kinds of security assumptions. Until now, BF[15], SK[46, 24, 25], and BB_1 [10] systems have been considered as practical constructions and their security was all demonstrated in the random oracle model.

In an attempt to prove security without random oracles, Canetti et al. [21] suggested a weaker security notion for IBE, known as selective-ID security. Following the weaker notion, Boneh and Boyen [10] proposed two efficient IBE systems, one of which was the basis for BB_1. Many IBE systems [49, 32, 41, 42, 50] were later suggested to achieve full security without random oracles, but they all become less efficient than the random oracle constructions in practical aspects such as public parameter size or achieving chosen ciphertext security.

Regarding tight security reduction, Attrapadung et al.[5] proposed a Katz-Wang variant of the BF system whose security is tightly reduced to the DBDH assumption. Their construction is fully secure against chosen ciphertext attacks in the random oracle model, but impractical especially in terms of both encryption and decryption costs. On the other hand, Gentry IBE [32] achieved the full security without random oracles. Tightness in its security reduction was achieved by relying on a (non-standard) q-type assumption where q depends on the number of private key queries that an adversary makes.

The notion of IBE has been extended in two flavors. In a vertical (and hierarchical) extension, IBE can provide a 'delegation' mechanism [39, 35] by which private keys for lower-level identities are created from an upper-level identity but the reverse is not possible. Many works [35, 10, 12, 49, 20, 47, 33, 50] have been suggested to realize such a delegation mechanism, also known as Hierarchical IBE (HIBE). In a horizontal extension, IBE becomes the special case of the Attribute-Based Encryption (ABE) [45, 38, 9], where attributes (instead of a single identity) are associated with private keys and ciphertexts, respectively, and decryption works only if attributes satisfy a function depending on each ABE system. Furthermore, when attributes (an identity) embedded into ciphertexts are encrypted, ABE (IBE) can also be extended for providing searchable techniques [14, 1] on encrypted data. Recently, the horizontal extensions are conceptually united under the notion of Functional Encryption (FE) [18].

Finally, we notice that there exist other approaches to build IBE trapdoors without pairings. Cocks [27] and Boneh et al. [16] constructed IBE systems based on the quadratic-residuosity problem and Gentry et al. [34] demonstrated how to build an IBE system based on lattice. Recently, lattice-based IBE can also be extended toward HIBE [23, 2, 3] and FE [4] constructions.

2 Preliminaries

2.1 Identity-Based Encryption

An Identity-Based Encryption (IBE) system consists of four algorithms:

- **Setup**(k) takes a security parameter k as input and outputs a public parameter PP and a master secret key msk.

- **KeyGen**(msk, PP, ID) takes a master secret key msk, a public parameter PP and an identity $\mathsf{ID} \in \mathcal{ID}$ as inputs, where $\mathcal{ID}$ is an identity space. It outputs $\mathsf{sk}_{\mathsf{ID}}$, a private key for ID.
- **Encrypt**(PP, M, ID') takes a public parameter PP, a message $M \in \mathcal{M}$, and an identity $\mathsf{ID} \in \mathcal{ID}$ as inputs, where $\mathcal{M}$ is a message space. It outputs CT under ID, a ciphertext under ID.
- **Decrypt**(CT, PP, $\mathsf{sk}_{\mathsf{ID}}$) takes a ciphertext CT under ID', a public parameter PP, and a private key $\mathsf{sk}_{\mathsf{ID}}$ as inputs. It outputs a message M or $\perp$.

Correctness. For all $\mathsf{ID}, \mathsf{ID}' \in \mathcal{ID}$ and all $M \in \mathcal{M}$, let $(\mathsf{PP}, \mathsf{msk}) \leftarrow$ **Setup**(k), $\mathsf{sk}_{\mathsf{ID}} \leftarrow$ **KeyGen**$(\mathsf{msk}, \mathsf{PP}, \mathsf{ID})$, $\mathsf{CT} \leftarrow$ **Encrypt**$(\mathsf{PP}, M, \mathsf{ID}')$. We have $M \leftarrow$ **Decrypt**$(\mathsf{sk}_{\mathsf{ID}}, \mathsf{PP}, \mathsf{CT})$.

We next define the chosen ciphertext security [15] of an IBE system, which is commonly accepted. The security is defined via the following game interacted by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- **Setup**: $\mathcal{C}$ runs the setup algorithm to obtain a public parameter PP and a master secret key msk. $\mathcal{C}$ gives PP to $\mathcal{A}$.
- **Query Phase 1**: $\mathcal{A}$ adaptively issues a number of queries where each query is one of:
  - Private key query on ID: $\mathcal{C}$ runs the key generation algorithm to obtain a private key for ID and gives the key $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$.
  - Decryption query on $(\mathsf{CT}, \mathsf{ID})$: $\mathcal{C}$ runs the key generation algorithm to obtain $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$ and then runs the decryption algorithm using $\mathsf{CT}_{\mathsf{ID}}$ and $\mathsf{sk}_{\mathsf{ID}}$. It gives the resulting message to $\mathcal{A}$.
- **Challenge**: $\mathcal{A}$ outputs two equal-length messages M_0, M_1 and an identity ID^* on which it wishes to be challenged. The only restriction is that ID is not queried in Query Phase 1. $\mathcal{C}$ flips a coin $\sigma \in \{0,1\}$. $\mathcal{C}$ gives $\mathsf{CT}^* \leftarrow$ **Encrypt**$(\mathsf{PP}, M_\sigma, \mathsf{ID}^*)$ as a challenge ciphertext to $\mathcal{A}$.
- **Query Phase 2**: $\mathcal{A}$ adaptively issues a number of additional queries where each query is one of:
  - Private key query on ID, where $\mathsf{ID} \neq \mathsf{ID}^*$: $\mathcal{C}$ responds as in Query Phase 1.
  - Decryption query on $(\mathsf{CT}, \mathsf{ID})$, where $(\mathsf{CT}, \mathsf{ID}) \neq (\mathsf{CT}^*, \mathsf{ID}^*)$: $\mathcal{C}$ responds as in Query Phase 1.
- **Guess**: $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$. $\mathcal{A}$ wins if $\sigma' = \sigma$.

The advantage of the adversary $\mathcal{A}$ in breaking the chosen ciphertext security of an IBE system $\mathcal{IBE}$ is defined as $\mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{IBE},\mathcal{A}} = \left|\Pr[b' = b] - 1/2\right|$.

Definition 1. *We say that an IBE system is $(t, \varepsilon, q_K, q_D)$-IND-ID-CCA secure if for any polynomial time adversary $\mathcal{A}$ that runs in time at most t, issues at most q_K private key queries and at most q_D decryption queries in chosen ciphertext security games, we have that $\mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{IBE},\mathcal{A}} < \varepsilon$.*

2.2 One-time Symmetric-key Encryption

A one-time symmetric-key encryption scheme consists of two algorithms: a deterministic encryption algorithm $\mathcal{E}$ takes a message $M \in \{0,1\}^*$ and a key $K \in \mathcal{K}$ as inputs and outputs a ciphertext $C = \mathcal{E}_K(M)$. Here, $\mathcal{K}$ is a key space that is determined by a security parameter $k \in \mathbb{Z}^+$. Another deterministic algorithm $\mathcal{D}$ is a decryption algorithm that takes a ciphertext C and a key K as inputs and outputs a message $M = \mathcal{D}_K(C)$.

We define security for a one-time symmetric-key encryption scheme $\mathcal{SKE} = (\mathcal{E}, \mathcal{D})$, which is security against *passive* attacks [28]. The security is defined via the following game interacted by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- **Setup**: $\mathcal{C}$ chooses a random key K in key space $\mathcal{K}(k)$.
- **Challenge**: $\mathcal{A}$ outputs two equal-length messages M_0 and M_1. $\mathcal{C}$ flips a coin $\sigma \in \{0,1\}$ and gives $C^* \leftarrow \mathcal{E}_K(M_\sigma)$ as a challenge ciphertext to $\mathcal{A}$.
- **Guess**: $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$. $\mathcal{A}$ wins if $\sigma' = \sigma$.

The advantage of the adversary $\mathcal{A}$ in breaking the passive security of a one-time symmetric-key encryption scheme $\mathcal{SKE}$ is defined as $\mathbf{Adv}^{\text{OT-IND}}_{\mathcal{SKE},\mathcal{A}} = \left|\Pr[b' = b] - 1/2\right|$.

Definition 2. *We say that a one-time symmetric-key encryption scheme is (t,ε)-secure against passive attacks if for any polynomial time adversary $\mathcal{A}$ that runs in time at most t in passive attack games, we have that $\mathbf{Adv}^{\textit{OT-IND}}_{\mathcal{SKE},\mathcal{A}} < \varepsilon$.*

2.3 Bilinear Maps and Complexity Assumptions

We briefly review bilinear maps and the complexity assumptions. Here, we simply consider symmetric pairings in prime-order groups.

Bilinear Maps: We follow the standard notation in [15, 10]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of prime order p. We assume that g is a generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a function that has the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u,v)^{ab}$.
2. Non-degenerate: $e(g,g) \neq 1$.
3. Computable: there is an efficient algorithm to compute the map e.

Then, we say that $\mathbb{G}$ is a bilinear group and the map e is a bilinear pairing in $\mathbb{G}$. Note that $e(,)$ is symmetric since $e(g^a, g^b) = e(g,g)^{ab} = e(g^b, g^a)$.

The Decisional Bilinear Diffie-Hellman (DBDH) Problem: The DBDH problem [15] is defined as follows: given $(g, g^a, g^b, g^c, T) \in \mathbb{G}^4 \times \mathbb{G}_T$ as input, output 1 if $T = e(g,g)^{abc}$ and 0 otherwise. We say that an algorithm $\mathcal{A}$ that outputs $\sigma \in \{0,1\}$ has an advantage $\mathbf{Adv}^{\text{DBDH}}_{\mathbb{G},\mathcal{A}} = \varepsilon$ in solving the DBDH problem in $\mathbb{G}$ if

$$\left|\Pr\left[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0\right] - \Pr\left[\mathcal{A}(g, g^a, g^b, g^c, R) = 0\right]\right| \geq \varepsilon,$$

where the probability is taken over the random choice of a, b, $c \in \mathbb{Z}_p$, the random choice of $R \in \mathbb{G}_T$, and the random bits used by $\mathcal{A}$.

Definition 3. *We say that the (t,ε)-DBDH assumption holds in $\mathbb{G}$ if no polynomial time adversary $\mathcal{A}$ that runs in time at most t has at least advantage ε in solving the DBDH problem in $\mathbb{G}$.*

3 Our IBE System

3.1 Construction

Setup(k): Given a security parameter $k \in \mathbb{Z}^+$, the setup algorithm runs $\mathcal{G}(k)$ to obtain a tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$. The algorithm selects a random generator $g \in \mathbb{G}$, a random group element $u \in \mathbb{G}$, and a random exponent $\alpha \in \mathbb{Z}_p$. The algorithm sets $A = e(g,g)^\alpha$ and chooses two cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$

and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$. The public parameters PP (with the description of $(p, \mathbb{G}, \mathbb{G}_T, e)$) and the master secret key msk are generated as

$$\mathsf{PP} = \big(g, u, A, H_1, H_2\big), \quad \mathsf{msk} = \alpha.$$

KeyGen(msk, PP, ID)**:** To create a private key $\mathsf{sk}_{\mathsf{ID}}$ for an identity $\mathsf{ID} \in \mathcal{ID}$, the key generation algorithm does the following:

1. Pick a random exponent $r \in \mathbb{Z}_p$.
2. Compute $d_0 = g^{\alpha} u^r \in \mathbb{G}$, $d_1 = g^r \in \mathbb{G}$, and $\mathrm{tag}_k = H_2(d_0, d_1) \in \mathbb{Z}_p$.
3. Compute $d_2 = \big(H_1(\mathsf{ID}) u^{\mathrm{tag}_k}\big)^r \in \mathbb{G}$.
4. Output a private key $\mathsf{sk}_{\mathsf{ID}} = (d_0, d_1, d_2, \mathrm{tag}_k) \in \mathbb{G}^3 \times \mathbb{Z}_p$.

Encrypt(PP, ID, M)**:** To encrypt a message $M \in \mathbb{G}_T$ under an identity $\mathsf{ID} \in \mathcal{ID}$, the encryption algorithm does as follows:

1. Pick a random exponent $s \in \mathbb{Z}_p$.
2. Compute $C_0 = A^s M \in \mathbb{G}_T$, $C_1 = g^s \in \mathbb{G}$, and $\mathrm{tag}_c = H_2(C_0, C_1) \in \mathbb{Z}_p$.
3. Compute $C_2 = \big(H_1(\mathsf{ID}) u^{\mathrm{tag}_c}\big)^s \in \mathbb{G}$.
4. Output a ciphertext $\mathsf{CT} = (C_0, C_1, C_2) \in \mathbb{G}_T \times \mathbb{G}^2$.

Decrypt(PP, CT, $\mathsf{sk}_{\mathsf{ID}}$)**:** To decrypt a ciphertext $\mathsf{CT} = (C_0, C_1, C_2)$ using a private key $\mathsf{sk}_{\mathsf{ID}} = (d_0, d_1, d_2, \mathrm{tag}_k)$ for ID, the decryption algorithm does as follows:

1. Compute $\mathrm{tag}_c = H_2(C_0, C_1) \in \mathbb{Z}_p$ and check if $e\big(H_1(\mathsf{ID}) u^{\mathrm{tag}_c},\ C_1\big) \stackrel{?}{=} e(C_2, g)$.
2. If the equality above fails, output $\perp$.
3. Otherwise, check if $\mathrm{tag}_c \stackrel{?}{=} \mathrm{tag}_k$ in $\mathbb{Z}_p$.
4. If the equality above holds, output $\perp$.
5. Otherwise, compute

$$M = C_0 \Big/ e(d_0, C_1) \cdot \left(\frac{e(C_2, d_1)}{e(d_2, C_1)} \right)^{\frac{-1}{\mathrm{tag}_c - \mathrm{tag}_k}}. \tag{1}$$

Correctness. If $\mathrm{tag}_c = \mathrm{tag}_k$ in $\mathbb{Z}_p$, the decryption algorithm does not work, so that it has the correctness error $1/p$ on each decryption. Otherwise, we can verify that the decryption algorithm works correctly for well-formed ciphertexts as follows:

$$\begin{aligned} e(d_0, C_1) \cdot \left(\frac{e(C_2, d_1)}{e(d_2, C_1)} \right)^{\frac{-1}{\mathrm{tag}_c - \mathrm{tag}_k}} &= e\big(g^{\alpha} u^r,\ g^s\big) \cdot \left(\frac{e\big((H_1(\mathsf{ID}) u^{\mathrm{tag}_c})^s,\ g^r\big)}{e\big((H_1(\mathsf{ID}) u^{\mathrm{tag}_k})^r,\ g^s\big)} \right)^{\frac{-1}{\mathrm{tag}_c - \mathrm{tag}_k}} \\ &= e(g, g^{\alpha})^s e(u, g)^{sr} \cdot e\big(u^{\mathrm{tag}_c - \mathrm{tag}_k},\ g^{sr}\big)^{\frac{-1}{\mathrm{tag}_c - \mathrm{tag}_k}} \\ &= A^s. \end{aligned}$$

Encryption and Decryption Costs. In encryption, the three exponentiations A^s, g^s, and $u^{\mathsf{tag}_c \cdot s}$ can be calculated in fixed bases A, g, and u, respectively. Instead, the hashing $H_1(\mathsf{ID})$ and its exponentiation $H_1(\mathsf{ID})^s$ will be done separately without precomputation in usual situations. Thus, the encryption cost becomes 3 fixed-base exponentiations plus 1 hashing and 1 general exponentiation.

Upon decryption, it seems that the decryption algorithm requires computing 5 pairings, but these can be saved into 1.2 parings. We first can change the above formula (1) into:

$$e(d_0, C_1) \cdot \left(\frac{e(C_2, d_1)}{e(d_2, C_1)} \right)^{\frac{-1}{\mathsf{tag}_c - \mathsf{tag}_k}} = \frac{e\Big(C_2,\, d_1^{\frac{-1}{\mathsf{tag}_c - \mathsf{tag}_k}}\Big)}{e\Big(d_0^{-1} d_2^{\frac{-1}{\mathsf{tag}_c - \mathsf{tag}_k}},\, C_1\Big)}.$$

Next, by using the *implicit* consistency check [41], we do not need to perform the pairing consistency test explicitly. Instead, the decryption algorithm randomizes two elements $H_1(\mathsf{ID})u^{\mathsf{tag}_c}$ and g by raising a randomly chosen exponent $\tilde{r} \in \mathbb{Z}_p$ and performs the following computation:

$$\frac{e\Big(C_2,\, d_1^{\frac{-1}{\mathsf{tag}_c - \mathsf{tag}_k}} g^{\tilde{r}}\Big)}{e\Big(d_0^{-1} d_2^{\frac{-1}{\mathsf{tag}_c - \mathsf{tag}_k}} (H_1(\mathsf{ID})u^{\mathsf{tag}_c})^{\tilde{r}},\, C_1\Big)}. \tag{2}$$

If the pairing test passes, the output of the above equation becomes the same as that of the original decryption algorithm. Otherwise, the fresh random value $\tilde{r}$ chosen by the decryption algorithm survives and thus prevents an adversary from gaining any information on an ill-formed ciphertext. As a consequence, the decryption cost is determined by the computation in the equation (2) that shows five exponentiations and two pairing computations. All the exponentiations can be done in fixed bases such as g, d_1, d_2, u, and $H_1(\mathsf{ID})$. Notice that a user with identity ID can compute $H_1(\mathsf{ID}) \in \mathbb{G}$ and prepare for fixed bases related with it, regardless of any received ciphertext. Also, a ratio of two pairings can be estimated into 1.2 pairings [19]. Thus, the decryption cost is concluded with 5 fixed-base exponentiations plus 1.2 parings.

Achieving Perfect Correctness. Upon decryption, our decryption algorithm cannot proceed in the event that $\mathsf{tag}_c = \mathsf{tag}_k$ occurs. Obviously, the probability that the event happens is negligible when the value tag is in $\mathbb{Z}_p$ and p is represented by approximately 160 bits. However, in order to avoid even the negligible correctness error, we can locate a suitable approach in a recent tag-based dual system encryption [50] where a similar decryption process to ours appears. A possible solution is to simply run an efficient selectively (chosen-ciphertext) secure IBE system [10] in parallel. When a message is encrypted under ID with tag_c, an encryptor also encrypts the message under the tag_c in the second selective system. When the two tags are different, we can use our original IBE system. In the unlikely event that the two tags are equal, we can use the second ciphertext. An alternative approach in [50] such as giving two private keys for an identity ID seems to not be applicable to our system, because we can assign a hash value $H_1(\mathsf{ID})$ to only one tag value in our security analysis.

Construction under Asymmetric Parings. The system can be instantiated in bilinear groups where asymmetric pairing $e : \mathbb{G} \times \widehat{\mathbb{G}} \to \mathbb{G}_T$ is defined over different groups $\mathbb{G}$ and $\widehat{\mathbb{G}}$. In that case, we need to hash an identity into a group element in either $\mathbb{G}$ or $\widehat{\mathbb{G}}$. If we choose $\mathbb{G}$, then a private key consists of group elements in $\mathbb{G} \times \widehat{\mathbb{G}} \times \mathbb{G}$ and a ciphertext has elements in $\mathbb{G}_T \times \widehat{\mathbb{G}} \times \mathbb{G}$. The reverse is also possible, and the selection of each option affects the efficiency of the IBE system when instantiated with MNT [19] or BN [26] curves.

3.2 Security

Theorem 1. *Let H_1 and H_2 be modeled as random oracles. Suppose the (t', ε')-DBDH assumption holds in $\mathbb{G}$. Then our IBE system is $(t, \varepsilon, q_K, q_D)$-IND-ID-CCA secure, where*

$$\varepsilon \leq \Big(1 - \frac{q_{H_2}}{p} - \frac{2q_D}{p}\Big) \cdot \varepsilon',$$
$$t \geq t' - \mathcal{O}(q_K \cdot t_e) - \mathcal{O}(q_D \cdot t_p).$$

Here, t_e is the cost of an exponentiation in $\mathbb{G}$ and t_p is the cost of a pairing computation in $\mathbb{G}$.

Proof. Suppose that there exists an adversary $\mathcal{A}$ which can break the CCA security of our IBE system. We can then build an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve a DBDH problem in $\mathbb{G}$. On input (g, g^a, g^b, g^c, T), $\mathcal{B}$ attempts to output 1 if $T = e(g,g)^{abc}$ and 0 otherwise. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup $\mathcal{B}$ selects a random element $\delta \in \mathbb{Z}_p$ and sets $u = g^{-a}g^{\delta}$ and $A = e(g^a, g^b)$. Note that $\alpha = ab \in \mathbb{Z}_p$, which is unknown to $\mathcal{B}$. Then, $\mathcal{A}$ is given the public key $\mathsf{PK} = (g, u, A, H_1, H_2)$, where H_1 and H_2 are modeled as random oracles.

Query Phase 1 $\mathcal{A}$ issues H_1, H_2, private key, and decryption queries. $\mathcal{B}$ responds as follows:

$\underline{H_1 \text{ queries:}}$ To respond to H_1 queries, $\mathcal{B}$ maintains a list of tuples $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$ as explained below. We refer to this list as the H_1^{list}. When $\mathcal{B}$ is given an identity ID_i as an input to H_1, $\mathcal{B}$ first scans through the H_1^{list} to see if the input ID_i appears in a tuple $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$. If it does, $\mathcal{B}$ responds with $H_1(\mathsf{ID}_i)$. Otherwise, $\mathcal{B}$ picks two random exponents $\gamma_i, \pi_i \in \mathbb{Z}_p$ and sets $H_1(\mathsf{ID}_i) = (g^a)^{\gamma_i} g^{\pi_i} \in \mathbb{G}$. $\mathcal{B}$ adds the new tuple $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$ to the H_1^{list} and responds with $H_1(\mathsf{ID}_i)$. Recall that the values $\{\gamma_i\}$ are information-theoretically hidden to $\mathcal{A}$'s view.

$\underline{H_2 \text{ queries:}}$ To respond to H_2 queries, $\mathcal{B}$ maintains a list of tuples $< W_i, Q_i, \mu_i >$ as explained below. We refer to this list as the H_2^{list}. When $\mathcal{B}$ is given values (W_i, Q_i), which is in either $\mathbb{G}^2$ or $\mathbb{G}_T \times \mathbb{G}$, as an input to H_2, $\mathcal{B}$ first scans through the H_2^{list} to see if the input (W_i, Q_i) appears in a tuple $< W_i, Q_i, \mu_i >$. If it does, $\mathcal{B}$ responds with $H_2(W_i, Q_i) = \mu_i$. Otherwise, $\mathcal{B}$ picks a random exponent $\mu_i \in \mathbb{Z}_p$ and sets $H_2(W_i, Q_i) = \mu_i$. $\mathcal{B}$ adds the new tuple $< W_i, Q_i, \mu_i >$ to the H_2^{list} and responds with $H_2(W_i, Q_i)$.

$\underline{\text{Key queries:}}$ When $\mathcal{B}$ is given an identity $\mathsf{ID}_i \in \mathcal{ID}$ as an input to a private key query, $\mathcal{B}$ selects a random exponent $r \in \mathbb{Z}_p$ and (implicitly) sets $\widetilde{r} = b + r \in \mathbb{Z}_p$. $\mathcal{B}$ generates key elements $(d_{0,i}, d_{1,i})$ as $d_{0,i} = (g^a)^{-r}(g^b)^{\delta} g^{\delta r}$ and $d_{1,i} = g^b g^r$. The validity of those elements can be verified as follows:

$$d_{0,i} = (g^a)^{-r}(g^b)^{\delta} g^{\delta r} = g^{ab}(g^{-a}g^{\delta})^{b+r} = g^{\alpha} u^{\widetilde{r}}, \qquad d_{1,i} = g^b g^r = g^{\widetilde{r}}.$$

Next, $\mathcal{B}$ refers to the H_1^{list} to find out the tuple $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$. (If no tuple exists, $\mathcal{B}$ can run the H_1-query process before replying to the key query.) At this moment, $\mathcal{B}$'s goal is to set $H_2(d_{0,i}, d_{1,i}) = \gamma_i$. Thus, if there is a tuple $< d_{0,i}, d_{1,i}, \gamma_i >$ in the H_2^{list}, $\mathcal{B}$ can continue the key query process.

In fact, $\mathcal{B}$ can make such a (favorable) tuple always exist in the H_2^{list} as follows: whenever $\mathcal{B}$ adds a new tuple $< \mathsf{ID}_i$, γ_i, π_i, $H_1(\mathsf{ID}_i) >$ to the H_1^{list}, $\mathcal{B}$ generates $\mathsf{sk}_{\mathsf{ID}_i}$ by choosing a random r, constructing key elements $(d_{0,i}, d_{1,i})$ as above, setting $H_2(d_{0,i}, d_{1,i}) = \gamma_i$, and adding the tuple $< d_{0,i}, d_{1,i}, \gamma_i >$ to the H_2^{list}. On the other hand, if $H_2(d_{0,i}, d_{1,i})$ has already be set to μ_i, then $\mathcal{B}$ simply adds a new tuple $< \mathsf{ID}_i$, μ_i, π_i, $H_1(\mathsf{ID}_i) >$ to the H_1^{list}.

Without loss of generality, let the tuple $H_2(d_{0,i}, d_{1,i}) = \gamma_i$ (where γ_i is from the tuple in the H_1^{list} above) be in the H_2^{list}. $\mathcal{B}$ generates the element $d_{2,i}$ as $d_{2,i} = (g^b)^{\pi_i + \gamma_i \delta} g^{(\pi_i + \gamma_i \delta) r}$. The validity of $d_{2,i}$ can be verified as follows:

$$d_{2,i} = (g^b)^{\pi_i+\gamma_i\delta} g^{(\pi_i+\gamma_i\delta)r} = \big((g^a)^{\gamma_i} g^{\pi_i} \cdot (g^{-a+\delta})^{\gamma_i}\big)^{b+r} = \big(H_1(\mathsf{ID}_i) u^{H_2(d_{0,i},d_{1,i})}\big)^{\widetilde{r}}.$$

Then, $\mathcal{B}$ responds with a private key $\mathsf{sk}_{\mathsf{ID}_i} = (d_{0,i}, d_{1,i}, d_{2,i}, \mathrm{tag}_k = \gamma_i)$ for the requested identity ID_i.

Decryption queries: When $\mathcal{B}$ is given a ciphertext $\mathsf{CT}_i = (C_{0,i}, C_{1,i}, C_{2,i})$ (as well as an identity ID_i) as an input to a decryption query, $\mathcal{B}$ first refers to the H_1^{list} to find out the tuple $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$. (If no tuple exists, $\mathcal{B}$ can run the H_1-query process in advance as explained above.) Next, $\mathcal{B}$ generates a private key $\mathsf{sk}_{\mathsf{ID}_i} = (d_{0,i}, d_{1,i}, d_{2,i}, \mathrm{tag}_k)$ for the identity ID_i or uses the private key $\mathsf{sk}_{\mathsf{ID}_i}$ that was generated before. Recall that $H_2(d_{0,i}, d_{1,i}) = \gamma_i = \mathrm{tag}_k$.

To check whether $H_2(C_{0,i}, C_{1,i}) = \mathrm{tag}_c \overset{?}{=} \mathrm{tag}_k = H_2(d_{0,i}, d_{1,i})$, $\mathcal{B}$ refers to the H_2^{list} to search for a tuple $< C_{0,i}, C_{1,i}, \widetilde{\mu}_i >$. If such a tuple regarding $(C_{0,i}, C_{1,i})$ does not exist, $\mathcal{B}$ sets $H_2(C_{0,i}, C_{1,i}) = \widetilde{\mu}_i$ by choosing a random $\widetilde{\mu}_i \in \mathbb{Z}_p$ and adds the new tuple $< C_{0,i}, C_{1,i}, \widetilde{\mu}_i >$ to the H_2^{list}.

[*Case 1.*] If $\widetilde{\mu}_i = \gamma_i$, $\mathcal{B}$ aborts. (We refer to this event as abort1.) Notice that γ_i is from the tuple $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$ in the H_1^{list}. In the real decryption, the equality means that $\mathrm{tag}_c = \mathrm{tag}_k$ and thus the normal decryption is expected to output $\perp$, but $\mathcal{B}$ simply aborts in our simulation. We will give the reason below. Fortunately, the probability that the event abort1 happens is negligible while H_2 acts like a random oracle.

[*Case 2.*] If $\widetilde{\mu}_i \neq \gamma_i$, $\mathcal{B}$ performs the normal decryption using $\mathsf{sk}_{\mathsf{ID}_i}$ and replies with the resulting message.

Challenge $\mathcal{A}$ outputs two messages $M_0, M_1 \in \mathbb{G}_T$ and an identity ID^* on which it wishes to be challenged. If necessary, $\mathcal{B}$ runs the algorithm for responding to H_1 query on ID^*. Let $< \mathsf{ID}^*, \gamma^*, \pi^*, H_1(\mathsf{ID}^*) >$ be the tuple in the H_1^{list} regarding the challenged identity ID^*. Notice that $\mathcal{A}$ cannot query a private key for ID^*. This means that the exponent γ^* in the tuple is not revealed to $\mathcal{A}$ (with overwhelming probability) until the Challenge phase.

$\mathcal{B}$ picks a random bit $\sigma \in \{0,1\}$ and sets $C_0^* = M_\sigma T$ and $C_1^* = g^c$. It sets $H_2(C_0^*, C_1^*) = \gamma^*$. If the tuple $< C_0^*, C_1^*, \gamma_j >$ are already in the H_2^{list} and $\gamma_j \neq \gamma^*$, then $\mathcal{B}$ aborts. (We refer to this event as abort2.) Otherwise, $\mathcal{B}$ generates the ciphertext $\mathsf{CT}^* = (C_0^*, C_1^*, C_2^*) = \big(M_\sigma T,\, g^c,\, (g^c)^{\pi^*+\delta\gamma^*}\big)$. $\mathcal{B}$ (implicitly) sets $s = c$. The validity of C_2^* can then be verified as follows:

$$(g^c)^{\pi^*+\delta\gamma^*} = \big((g^a)^{\gamma^*} g^{\pi^*} \cdot (g^{-a+\delta})^{\gamma^*}\big)^c = \big(H_1(\mathsf{ID}^*) u^{H_2(C_0^*, C_1^*)}\big)^s.$$

Query Phase 2 $\mathcal{A}$ issues more $\{H_i\}_{i=1,2}$, private key, and decryption queries. $\mathcal{B}$ responds as in Query Phase 1. At this phase, however, there are challenging decryption queries $\mathcal{B}$ has to deals with. That happens when $\mathcal{A}$ issues *valid* ciphertexts such as $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_{2,i})$, where C_1^* is the same as in CT^*. Here, we call a ciphertext $\mathsf{CT} = (C_0, C_1, C_2)$ under an identity ID *valid* if the pairing test upon decryption holds, i.e., $e\big(H_1(\mathsf{ID}) u^{H_2(C_0,C_1)},\, C_1\big) = e(C_2, g)$. In such a case, $\mathcal{B}$ should decrypt it correctly using the value $e(g,g)^{abc}$, which is infeasible. More precisely, there are four cases:

[*Case 1.*] $\mathsf{CT}_i = (C_0^*, C_1^*, C_{2,i})$ on ID^*, where $C_{2,i} \neq C_2^*$. As the ciphertext is valid, it passes the pairing test upon decryption. Thus, $\mathcal{B}$ has that $e\big(H_1(\mathsf{ID}^*) u^{H_2(C_0^*, C_1^*)},\, C_1^*\big) = e(C_{2,i}, g)$. Since $C_1^* = g^c$, the equation shows $C_{2,i} = (H_1(\mathsf{ID}^*) u^{H_2(C_0^*, C_1^*)})^c$, which must be the same as C_2^*. This means that such a valid ciphertext in the form of $(C_0^*, C_1^*, C_{2,i})$ such that $C_{2,i} \neq C_2^*$ is not possible.

[*Case 2.*] $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_2^*)$ on ID^*, where $C_{0,i} \neq C_0^*$. This case can happen only if $\mathcal{B}$ sets $H_2(C_{0,i}, C_1^*) = \gamma^* \in \mathbb{Z}_p$. In this case, $\mathcal{B}$ aborts. (We refer to this event as abort3.) This is because otherwise, i.e., $\mathcal{B}$ returns $\perp$ as the decryption result and this gives the information of γ^* (as tag_k) in the $\mathsf{sk}_{\mathsf{ID}^*}$. γ^* (as tag_c^*) is already used for the challenge ciphertext, which gives the knowledge that γ^* is used two times in both $\mathsf{sk}_{\mathsf{ID}^*}$ and

CT^*. Naturally, such an unusual leakage can cause $\mathcal{A}$ to distinguish between the simulation and the real attack.

[*Case 3.*] $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_{2,i})$ on ID^*, where $C_{0,i} \neq C_0^*$ and $C_{2,i} \neq C_2^*$. As the ciphertext is valid, $\mathcal{B}$ has that $e\big(H_1(\mathsf{ID}^*)u^{H_2(C_{0,i},C_1^*)},\ C_1^*\big) = e(C_{2,i}, g)$. Since $C_1^* = g^c$, the equation shows $C_{2,i} = (H_1(\mathsf{ID}^*)u^{H_2(C_{0,i},C_1^*)})^c$. Also, since $C_{2,i} \neq C_2^*$, we know that $H_2(C_{0,i}, C_1^*) \neq \gamma^*$. Then, $\mathcal{B}$ has that

$$\begin{aligned} C_{2,i} &= \big(H_1(\mathsf{ID}^*)u^{H_2(C_{0,i},C_1^*)}\big)^s = \big((g^a)^{\gamma^*} g^{\pi^*} \cdot (g^{-a+\delta})^{H_2(C_{0,i},C_1^*)}\big)^c \\ &= (g^{ac})^{\gamma^* - H_2(C_{0,i},C_1^*)} (g^c)^{\pi^* + \delta H_2(C_{0,i},C_1^*)}, \end{aligned}$$

in which case $\mathcal{B}$ can obtain g^{ac} by computing $\big[C_{2,i}/(C_1^*)^{\pi^*+\delta H_2(C_{0,i},C_1^*)}\big]^{1/(\gamma^* - H_2(C_{0,i},C_1^*))}$. It follows that $\mathcal{B}$ can solve the given DBDH problem immediately.

[*Case 4.*] $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_{2,i})$ on $\mathsf{ID}(\neq \mathsf{ID}^*)$. Let $< \mathsf{ID}, \gamma, \pi, H_1(\mathsf{ID}) >$ be the tuple in the H_1^{list} regarding ID. From the pairing test equation, $\mathcal{B}$ has that $e\big(H_1(\mathsf{ID})u^{H_2(C_{0,i},C_1^*)},\ C_1^*\big) = e(C_{2,i}, g)$. Since $C_1^* = g^c$, the equation shows that $C_{2,i} = (H_1(\mathsf{ID})u^{H_2(C_{0,i},C_1^*)})^c$. If $H_2(C_{0,i}, C_1^*) = \gamma$, $\mathcal{B}$ outputs $\perp$ as the result of normal decryption.[2] Otherwise, $\mathcal{B}$ has that

$$\begin{aligned} C_{2,i} &= \big(H_1(\mathsf{ID})u^{H_2(C_{0,i},C_1^*)}\big)^s = \big((g^a)^{\gamma} g^{\pi} \cdot (g^{-a+\delta})^{H_2(C_{0,i},C_1^*)}\big)^c \\ &= (g^{ac})^{\gamma - H_2(C_{0,i},C_1^*)} (g^c)^{\pi + \delta H_2(C_{0,i},C_1^*)}, \end{aligned}$$

in which case $\mathcal{B}$ can obtain g^{ac} by computing $\big[C_{2,i}/(C_1^*)^{\pi+\delta H_2(C_{0,i},C_1^*)}\big]^{1/(\gamma - H_2(C_{0,i},C_1^*))}$. It follows that $\mathcal{B}$ can solve the given DBDH problem immediately.

Guess $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$. $\mathcal{B}$ then outputs its guess $\sigma' \in \{0,1\}$ as the solution to the given DBDH instance.

<u>*Comment.*</u> The reason why $\mathcal{B}$ aborts in the event $\mathsf{abort1}$ is that the equality can leak the information on the tag_k such that $H_2(d_{0,i}, d_{1,i}) = \gamma_i = \text{tag}_k$, where γ_i is from the tuple $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$ and $(d_{0,i}, d_{1,i})$ are from the private key $\mathsf{sk}_{\mathsf{ID}_i} = (d_{0,i}, d_{1,i}, d_{2,i}, \text{tag}_k = \gamma_i)$. That is, $\mathcal{A}$ is able to know that the value $\text{tag}_k (= \gamma_i)$ is used in $\mathsf{sk}_{\mathsf{ID}_i}$, even though all key elements in $\mathsf{sk}_{\mathsf{ID}_i}$ are not revealed to $\mathcal{A}$. The problem can then happen in the Challenge phase where $\mathcal{A}$ can select such ID_i as the challenge identity. Then, $\mathcal{B}$ has no choice but to use the same γ_i as the tag_c^* in constructing the challenge ciphertext such that $H_2(C_0^*, C_1^*) = \gamma_i$. This gives the information that γ_i is used two times in both $\mathsf{sk}_{\mathsf{ID}_i}$ and CT^*. Naturally, such a leakage can cause $\mathcal{A}$ to distinguish between the simulation and the real attack.

<u>*Analysis.*</u> The dominated additional computation that $\mathcal{B}$ requires is both the exponentiations for handling q_K private key queries and the pairings for handling q_D decryption queries. Thus, the inequality about computational time can easily be obtained.

Next, we assume Cases 3 and 4 described in the Query Phase 2 do not happen (which would rather increase $\mathcal{B}$'s success probability to solve the DBDH problem). To analyze $\mathcal{B}$'s advantage, we first prove the following claim that argues that the probability that $\mathcal{B}$ aborts in the simulation is at most $\frac{q_{H_2}}{p} + \frac{2q_D}{p}$, which is negligible.

Claim 1: $\Pr[\mathsf{abort}] = \Pr[\mathsf{abort1} \vee \mathsf{abort2} \vee \mathsf{abort3}]$ in the simulation is at most $\frac{q_{H_2}}{p} + \frac{2q_D}{p}$.

Proof. The event $\mathsf{abort1}$ happens if $\mathcal{B}$ sets $H_2(C_{0,i}, C_{1,i}) = \gamma_i$ for any queried ciphertext $\mathsf{CT}_i = (C_{0,i}, C_{1,i}, C_{2,i})$, where γ_i is from the tuple $< \mathsf{ID}_i, \gamma_i, \pi_i, H_1(\mathsf{ID}_i) >$ in the H_1^{list}. γ_i is a pre-determined value and the output of

[2]After the Challenge phase, we do not need to consider the event $\mathsf{abort1}$ in the case of $\mathsf{ID}(\neq \mathsf{ID}^*)$, since the distribution of tag values regarding $\mathsf{ID}(\neq \mathsf{ID}^*)$ is statistically identical to that in the real attack.

H_2 query is just set by choosing a random value in $\mathbb{Z}_p$. Thus, the probability that the event $\mathsf{abort1}$ happens is at most $1/p$ on each decryption query. Since $\mathcal{B}$ has to handle q_D decryption queries, the probability that the event $\mathsf{abort1}$ occurs throughout the simulation becomes at most q_D/p.

The event $\mathsf{abort2}$ can occur if the value $(M_\sigma T, g^c)$ already exists in the H_2^{list} as the input value queried by $\mathcal{A}$. There are p possibilities in picking a value as an input to the H_2 query, because it is determined by a randomly chosen exponent $c \in \mathbb{Z}_p$. This gives the probability at most q_{H_2}/p that the event $\mathsf{abort2}$ happens. (If we consider the possible cases from the selection of a value in $\mathbb{G}_T$, then the probability will be much smaller.)

Regarding the event $\mathsf{abort3}$, the event happens if $\mathcal{B}$ sets $H_2(C_{0,i}, C_1^*) = \gamma^*$ for any queried ciphertext $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_2^*)$ on ID^*, where $C_{0,i} \neq C_0^*$. Here, the value γ^* is the pre-determined value and the output of H_2 query is just set by choosing a random value in $\mathbb{Z}_p$. Thus, the probability that the event $\mathsf{abort3}$ happens is at most $1/p$ on each decryption query. Since $\mathcal{B}$ has to handle q_D decryption queries, the probability that the event $\mathsf{abort3}$ occurs throughout the simulation becomes at most q_D/p.

We know that all the events that $\mathcal{B}$ aborts are relatively independent. As a result, the probability $\Pr[\mathsf{abort1} \vee \mathsf{abort2} \vee \mathsf{abort3}]$ in the simulation is at most $\frac{q_{H_2}}{p} + \frac{2q_D}{p}$. $\square$

From Claim 1, we can see that the probability that $\mathcal{B}$ aborts in the simulation is negligible (under the appropriate selection of security parameters). We argue that as long as $\mathcal{B}$ does not abort, $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation whose distribution is identical to the distribution in a real attack. This is because (1) the simulation of H_1 and H_2 oracles are obviously perfect as the output values are determined by randomly chosen values in $\mathbb{G}$ and $\mathbb{Z}_p$, respectively, and (2) the simulation of private key oracles is also perfect as each key on an identity ID_i is generated with a randomly chosen exponent $r \in \mathbb{Z}_p$ such that $\widetilde{r} = b + r$, and (3) the simulation of decryption oracles is also perfect as it is done via normal decryption using private keys, and (4) the values $\{\gamma_i\}$ in the H_1^{list} are uniformly distributed and information-theoretically hidden from $\mathcal{A}$'s view until private keys and CT^* are given to $\mathcal{A}$.

As long as $\mathcal{B}$ does not abort in the simulation, $\mathcal{B}$ can use the $\mathcal{A}$'s advantage to break the chosen ciphertext security straightforwardly. This can be checked as follows: if $T = e(g,g)^{abc}$, then the challenge ciphertext CT^* is a valid encryption of M_σ under ID^*. Otherwise, i.e., if T is random in $\mathbb{G}_T$, then $M_\sigma T$ is independent of the bit σ. Thus, if $\mathcal{A}$ distinguishes between the two ciphertexts, then $\mathcal{B}$ can distinguish between the two possible values of T with the same advantage. Therefore, unless $\mathcal{B}$ does not abort, we have the following result:

$$\mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{IBE},\mathcal{A}}(k) \leq \left(1 - \frac{q_{H_2}}{p} - \frac{2q_D}{p}\right) \cdot \mathbf{Adv}^{\mathrm{DBDH}}_{\mathbb{G},\mathcal{B}}(k),$$

as required. This concludes the proof of Theorem 1. $\blacksquare$

3.3 Hash-BDH and BDH Construction

Our IBE system can be slightly modified to encrypt arbitrary n-bit message strings such as $M \in \{0,1\}^n$. To do this, we consider a family of hash functions of the form $H_3 : \mathbb{G}_T \rightarrow \{0,1\}^n$ (where $n \in Z^+$ is determined by the security parameter). A resultant ciphertext is then computed as $C_0 = H_3(A^s) \oplus M$, $C_1 = g^s$, and $C_3 = \left(H_1(\mathsf{ID})u^{\mathrm{tag}_c}\right)^s$ where $\mathrm{tag}_c = H_2(C_0, C_1)$. Decryption can be performed by hashing the pairing value in the equation (2) and XOR-ing the result with C_0.

The security of the modified system can be proven in two flavors: if H_3 is a random selection of the (appropriate) hash family, then the modified system is IND-ID-CCA secure under the Hash-BDH [10] assumption and the security reduction becomes tight. The proof is almost identical to that of Theorem 1. On the other hand, if H_3 is modeled as a random oracle, then the modified system is IND-ID-CCA secure under the

BDH [15] assumption and the security loss becomes $O(q_{H_3})$. In this case, $\mathcal{B}$ maintains additional H_3^{list} with respect to H_3, and the challenge ciphertext is constructed as $\mathsf{CT}^* = (C_0^* = R, C_1^* = g^c, C_2^* = \big(H_1(\mathsf{ID}^*)u^{\mathrm{tag}_c^*}\big)^c)$ where R is a randomly chosen string in $\{0,1\}^n$ and $\mathrm{tag}_c^* = H_2(C_0^*, C_1^*)$. C_0^* is not relevant to any of two challenged messages, and $\mathcal{B}$ just wants to employ the adversary's advantage to issue the correct answer of a BDH problem as the input of H_3 query. At the end of the simulation, $\mathcal{B}$ selects a random input value among tuples in the H_3^{list} as its answer to the BDH problem, which causes the security loss of $O(q_{H_3})$. We notice that a similar proof strategy was already used to prove IND-ID-CPA security of 'BasicIdent' in [15]. The rest of the proof can be completed based on the proof of the BasicIdent as well as Theorem 1.

4 Extension for Arbitrary length Messages

In this section we extend our IBE system to deal with arbitrary length messages. Our extended system is based on the well-known framework using the key encapsulation mechanism (KEM) and data encapsulation mechanism (DEM). Identity-Based KEM (IBKEM) encrypts a symmetric key under which an arbitrarily long message is encrypted under a symmetric-key cipher DEM. Usually, to achieve CCA security of an entire IBE system, both IBKEM and DEM should be CCA-secure respectively [8] or DEM should be an authenticated symmetric-key encryption [42]. However, a slight difference resides in the part of DEM of our extended IBE system where it is sufficient for DEM to be a one-time symmetric-key encryption secure against passive attacks [28]. In practice, such a weak DEM can easily be instantiated with a block cipher using a so-called 'counter mode'. The reason for the difference is that our IBE system is able to provide a consistency check (using pairing) to see if ciphertext elements including the DEM part are the same as what an encryptor constructed. We remark that a similar result concerning a weak DEM was achieved in [8] where BF-IBKEM is converted into a CCA-secure IBE system using the Fujisaki-Okamoto transform [30].

4.1 Construction

Setup(k)**:** As in the previous IBE system. Additionally, the setup algorithm chooses a one-time symmetric-key encryption scheme $\mathcal{SKE} = (\mathcal{E}, \mathcal{D})$. The public parameters PP and the master secret key msk are generated as

$$\mathsf{PP} = \big(g, u, A, H_1, H_2, \mathcal{SKE}\big), \quad \mathsf{msk} = \alpha.$$

KeyGen(msk, PP, ID)**:** As in the previous IBE system.

Encrypt(PP, ID, M)**:** To encrypt an arbitrary length message $M \in \{0,1\}^*$ under an identity $\mathsf{ID} \in \mathcal{ID}$, the encryption algorithm does as follows:

1. Pick a random exponent $s \in \mathbb{Z}_p$.
2. Compute a key $K = A^s \in \mathbb{G}_T$.
3. Compute $C_0 = \mathcal{E}_K(M)$, $C_1 = g^s$, and $\mathrm{tag}_c = H_2(C_0, C_1) \in \mathbb{Z}_p$.
4. Compute $C_2 = \big(H_1(\mathsf{ID})u^{\mathrm{tag}_c}\big)^s$.
5. Output a ciphertext $\mathsf{CT} = (C_0, C_1, C_2) \in \{0,1\}^{|M|} \times \mathbb{G}^2$.

Decrypt(PP, CT, $\mathsf{sk}_{\mathsf{ID}}$)**:** To decrypt a ciphertext $\mathsf{CT} = (C_0, C_1, C_2)$ using a private key $\mathsf{sk}_{\mathsf{ID}} = (d_0, d_1, d_2, \mathrm{tag}_k)$ for ID, the decryption algorithm does as follows:

1. Compute $\mathsf{tag}_c = H_2(C_0, C_1)$ and check if $e\big(H_1(\mathsf{ID})u^{\mathsf{tag}_c},\ C_1\big) \overset{?}{=} e(C_2, g)$.
2. If the equality above fails, output $\perp$.
3. Otherwise, check if $\mathsf{tag}_c \overset{?}{=} \mathsf{tag}_k$.
4. If the equality above holds, output $\perp$.
5. Otherwise, compute a key
$$K = e(d_0, C_1) \cdot \left(\frac{e(C_2, d_1)}{e(d_2, C_1)}\right)^{\frac{-1}{\mathsf{tag}_c - \mathsf{tag}_k}}.$$
6. Output a message $M = \mathcal{D}_K(C_0)$.

Remark. The efficiency of the IBE system above is almost the same as that in the previous section. Notice that the KEM part in a ciphertext is not expanded and the DEM part $C_0 = \mathcal{E}_K(M)$ is also hashed and embedded into the ciphertext element C_3. As a result, we can check the consistency of ciphertext elements including the DEM part and therefore avoid relying on an authenticated encryption scheme with the help of a secure message authentication code (MAC). In practice, a one-time symmetric-key encryption scheme $\mathcal{SKE}$ with key-space $\mathcal{K} \in \{0,1\}^k$ can be implemented by AES with a counter mode, and a real symmetric key for $\mathcal{E}$ can be obtained via a key-derivation function [28] that maps an element from $\mathbb{G}_T$ into $2k$-bit strings.

4.2 Security

Theorem 2. *Let H_1 and H_2 be modeled as random oracles. Suppose the (t', ε')-DBDH assumption holds in $\mathbb{G}$ and the one-time symmetric-key encryption scheme $\mathcal{SKE}$ is (t'', ε'')-secure against passive attacks. Then our IBE system is $(t, \varepsilon, q_K, q_D)$-IND-ID-CCA secure, where*

$$\varepsilon \leq \left(1 - \frac{q_{H_2}}{p} - \frac{2q_D}{p}\right) \cdot (\varepsilon' + \varepsilon''),$$
$$t \geq t' + t'' - \mathcal{O}(q_K \cdot t_e) - \mathcal{O}(q_D \cdot (t_p + t_{symD})).$$

Here, t_e is the cost of an exponentiation in $\mathbb{G}$, t_p is the cost of a pairing computation in $\mathbb{G}$, and t_{symD} is the cost of symmetric-key decryption in the $\mathcal{SKE}$.

Proof. The proof of Theorem 2 is almost similar to that of Theorem 1. For clarity, we reconstruct the entire proof by creating a sequence of hybrid games. If necessary, we will adapt several notations that appeared in the proof of Theorem 1 without explicit explanation.

Let $\mathcal{A}$ be an adversary on the chosen ciphertext security of our IBE system above. We will consider two games, **Game 0** and **Game 1**, each game interacting with $\mathcal{A}$. Let X_i (for $i = 0, 1$) be the event that in **Game i**, $\mathcal{A}$ wins the game.

Game 0. The game is a real attack game of chosen ciphertext security, so that (for a security parameter k) we have

$$|\Pr[X_0] - 1/2| = \mathbf{Adv}^{\text{CCA}}_{\mathcal{IBE},\mathcal{A}}(k). \tag{3}$$

Game 1. The game is the same as Game 0, except that the value K^* used for a symmetric key in CT^* is replaced by a random value $K \in \mathbb{G}_T$. Unless the events $\mathsf{abort1}$ and $\mathsf{abort2}$ and $\mathsf{abort3}$ (defined in the proof of Theorem 1) occur in **Game 1**, we have

$$|\Pr[X_1] - \Pr[X_0]| = \mathbf{Adv}^{\text{DBDH}}_{\mathbb{G},\mathcal{B}_1}(k), \tag{4}$$

where $\mathcal{B}_1$ is an algorithm whose goal is to solve a DBDH problem.

The proof of (4) is almost the same as that of Theorem 1. On an input (g, g^a, g^b, g^c, T), we describe how $\mathcal{B}_1$ constructs CT^*: when $\mathcal{A}$ outputs two messages $M_0, M_1 \in \{0,1\}^*$ of the same length and an identity ID^*, $\mathcal{B}_1$ picks a random bit $\sigma \in \{0,1\}$. Let $< \mathsf{ID}^*, \gamma^*, \pi^*, H_1(\mathsf{ID}^*) >$ be the tuple in the H_1^{list} regarding ID^*. $\mathcal{B}_1$ sets $C_0^* = \mathcal{E}_T(M_\sigma)$, $C_1^* = g^c$, and $H_2(C_0^*, C_1^*) = \gamma^*$. If the tuple $< C_0^*, C_1^*, \gamma_j >$ are already in the H_2^{list} and $\gamma_j \neq \gamma^*$, then $\mathcal{B}_1$ aborts. (In the proof of Theorem 1 (and thus Game 1 above), the event has already been taken into consideration under the event $\mathsf{abort2}$.) Otherwise, the challenge ciphertext is generated as $\mathsf{CT}^* = (C_0^*, C_1^*, C_2^*) = \big(\mathcal{E}_T(M_\sigma),\, g^c,\, (g^c)^{\pi^* + \delta\gamma^*}\big)$. Other slight differences reside in handling decryption and H_2 queries. Answering decryption queries needs to run $\mathcal{D}$ of $\mathcal{SKE}$, and inputs (W_i, Q_i) to H_2 queries are in either $\mathbb{G}^2$ or $\{0,1\}^* \times \mathbb{G}$ (instead of $\mathbb{G}^2$ or $\mathbb{G}_T \times \mathbb{G}$). However, the way of answering H_2 queries is the same as in the proof of Theorem 1. Thus, the probability that the events $\mathsf{abort1}$ and $\mathsf{abort2}$ and $\mathsf{abort3}$ happen in proving the equation (4) is the same as that in Theorem 1.

Finally, in **Game 1** we have the following relation that directly comes from the definition of ciphertext indistinguishability for $\mathcal{SKE}$ unless the event $\mathsf{abort3}$ occurs in **Game 1**:

$$|\Pr[X_1] - 1/2| = \mathbf{Adv}^{\text{OT-IND}}_{\mathcal{SKE},\mathcal{B}_2}(k), \tag{5}$$

where $\mathcal{B}_2$ is an algorithm whose goal is to break $\mathcal{SKE}$.

One thing we want to clarify is that $\mathcal{B}_2$ does not need to deal with any decryption query on the one-time symmetric-key encryption scheme $\mathcal{SKE}$. To prove the equation (5), $\mathcal{B}_2$ relays the two messages $M_0, M_1 \in \{0,1\}^*$ to its challenger and is given a challenge ciphertext $\mathcal{E}_{K^*}(M_\sigma)$ for a random (unknown) key $K^* \in \mathbb{G}_T$. $\mathcal{B}_2$ then reconstructs its challenge ciphertext CT^* as explained above and gives it to $\mathcal{A}$. Whenever $\mathcal{A}$ requests any decryption query on CT_i, $\mathcal{B}_2$ can use private keys (generated by $\mathcal{B}_2$) to perform normal decryption. The only troublesome queries happen when $\mathcal{A}$ issues decryption queries on valid $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_{2,i})$, in which case $\mathcal{B}_2$ would have to decrypt CT_i with the *unknown* symmetric-key K^*. Fortunately, those troublesome queries can be handled as in Query Phase 2 of Theorem 1. More precisely, there are four cases:

[*Case 1.*] $\mathsf{CT}_i = (C_0^*, C_1^*, C_{2,i})$ on ID^*, where $C_{2,i} \neq C_2^*$. As before, it is impossible to generate a valid ciphertext $(C_0^*, C_1^*, C_{2,i})$ such that $C_{2,i} \neq C_2^*$.

[*Case 2.*] $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_2^*)$ on ID^*, where $C_{0,i} \neq C_0^*$. This happens only if $\mathcal{B}$ sets $H_2(C_{0,i}, C_1^*) = \text{tag}_c^* \in \mathbb{Z}_p$. In this case, $\mathcal{B}$ aborts. (This event has already been referred to as $\mathsf{abort3}$.)[3]

[*Case 3.*] $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_{2,i})$ on ID^*, where $C_{0,i} \neq C_0^*$ and $C_{2,i} \neq C_2^*$. We have shown that the $\mathcal{A}$'s ability to issue such a valid ciphertext can be used to compute a computational Diffie-Hellman (CDH) value in the previous Query Phase 2.

[*Case 4.*] $\mathsf{CT}_i = (C_{0,i}, C_1^*, C_{2,i})$ on $\mathsf{ID}(\neq \mathsf{ID}^*)$. $\mathcal{B}$ generates $\mathsf{sk}_{\mathsf{ID}}$ and performs normal decryption. If $H_2(C_{0,i}, C_1^*) = \text{tag}_k$ where tag_k is from $\mathsf{sk}_{\mathsf{ID}}$, then $\mathcal{B}$ outputs $\perp$ as the result of normal decryption. Otherwise, we also have shown that the $\mathcal{A}$'s ability to issue such a valid ciphertext can be used to compute a computational Diffie-Hellman (CDH) value in the previous Query Phase 2.

Analysis. It is easy to see that time complexity is obviously achieved as argued in Theorem 2. As long as the events $\mathsf{abort1}$ and $\mathsf{abort2}$ and $\mathsf{abort3}$ do not happen throughout the simulation, the equations (4) and (5)

[3]Notice that $\mathcal{B}$ can handle this ciphertext if the one-time symmetric-key encryption scheme $\mathcal{SKE}$ is CCA-secure. However, we simply consider $\mathcal{SKE}$ as being secure against passive attacks by adding q_D/p into the probability that $\mathcal{B}_2$ aborts.

Table 1: Security assumptions and reductions for IBE systems

	Assumption	Security	Security reduction	
			Asymptotic bound	Concrete loss♠
BF	DBDH	IND-ID-CCA	$q_K q_H$	$> 2^{50}$
BB$_1$♡	DBDH	IND-ID-CCA	q_H	2^{50}
SK	q-DBDHI	IND-ID-CCA	q_H^2	$> 2^{50}$
Ours	DBDH	IND-ID-CCA	1	1

q_H: the number of (random-oracle) hash queries; q_K: the number of private key queries;
♠: estimated with $q_H = 2^{50}$ and $q_K = 2^{25}$; $q \approx q_H + q_K$;
♡: we borrow the CCA-secure variant of BB$_1$ system from [19];

hold. As a result, by putting the results of all relations (3), (4), and (5) above together, we gain a bound on the $\mathcal{A}$'s advantage as follows:

$$\mathbf{Adv}^{\text{CCA}}_{\mathcal{IBE},\mathcal{A}}(k) \leq \Big(1 - \frac{q_{H_2}}{p} - \frac{2q_D}{p}\Big) \cdot \big(\mathbf{Adv}^{\text{DBDH}}_{\mathbb{G},\mathcal{B}_1}(k) + \mathbf{Adv}^{\text{OT-IND}}_{\mathcal{SKE},\mathcal{B}_2}(k)\big),$$

which concludes the proof of Theorem 2. ■

5 Comparison to previous IBE systems

In this section we compare our IBE system with the previous practical IBE systems such as BF [15], SK [46, 24, 25], and the variant[4] of BB$_1$ [19] in terms of security and efficiency. For more detailed comparison between the three previous ones, we refer to Boyen's work [19]. Following Bellare and Rogaway [6, 42], we estimate the number of (random oracle) hash queries as $q_H = 2^{50}$ and the number of private key queries as $q_K = 2^{25}$. For a fair comparison we consider the decisional type of security assumptions in each system, so that BF and BB$_1$ systems are based on the Decisional BDH assumption and the SK system relies on the q-Decisional Bilinear Diffie-Hellman Inversion (DBDHI) assumption[5] [10] for $q \approx q_H + q_K$. We also refer to [31] for correcting a flawed security analysis of BF system. Table 1 presents the comparison result with respect to security assumptions and reductions, which was also addressed by [19, 42]. The 'asymptotic bound' in the security reduction means that the advantage of breaking the CCA security of an IBE system is larger than that of solving a security assumption in comparable time, by a factor of each bound. In other words, the larger the bound is, the bigger the security loss (i.e., gap) is between an IBE system and a security assumption. In Table 1, a concrete bound at each IBE system is estimated when considering the reasonable number of adversarial queries q_H and q_K as 2^{50} and 2^{25}, respectively. The respective bound tells us that, roughly speaking,

$$\mathbf{Adv}^{\text{CCA}}_{\mathcal{IBE},\mathcal{A}} \leq (\text{bound}) \cdot \mathbf{Adv}^{\text{Assumption}}_{\mathcal{B}} \tag{6}$$

for algorithms $\mathcal{A}$ and $\mathcal{B}$. When the bound is quite large, we have to choose a larger security parameter k for a security assumption so that we can make $(\text{bound}) \cdot \mathbf{Adv}^{\text{Assumption}}_{\mathcal{B}}$ small and consequently $\mathbf{Adv}^{\text{CCA}}_{\mathcal{IBE},\mathcal{A}}$

[4]We are not sure that the CCA security proof about the variant is correct because we cannot find any security proof for the variant. [19] stated that the proof of CCA security was adapted by [10], but in any part of [10] there exists no security proof related to the variant.

[5]It is defined from the problem: given $(g, g^x, g^{x^2}, \ldots, g^{x^q}, T)$, decide whether $T = e(g,g)^{1/x}$ or T is random in $\mathbb{G}_T$.

Table 2: Representation sizes and estimated calculation times for various algebraic operations

	Representation sizes (bits)				Relative timings								
	$\mathbb{Z}_p$	$\mathbb{G}$	$\widehat{\mathbb{G}}$	$\mathbb{G}_T$	$\mathbb{G}$		$\widehat{\mathbb{G}}$			$\mathbb{G}_T$			
					$E_f^{\clubsuit}$	$E^{\diamondsuit}$	E_f	E	$H^{\heartsuit}$	E_f	E	$P^{\flat}$	$P_r^{\spadesuit}$
SS / 80	160	512	512	1024	2	10	2	10	10	2	10	100	120
MNT / 80	160	171	1026	1026	0.2	1	8	40	40	2	10	100	120
MNT / 128	256	512	3072	3072	3	15	100	500	500	30	150	1500	1800

$\clubsuit$: fix-base exponentiation; $\diamondsuit$: general exponentiation; $\heartsuit$: hashing; $\flat$: single pairing; $\spadesuit$: ration of pairings;

Table 3: Efficiency comparison between CCA-secure IBE systems for SS curves at 80-bit security level (not considering security loss)

	Curves / security level	Overheads (bits)		Relative computational costs		
		Public params.	Ciphertext$^{\spadesuit}$	Key extraction	Encryption	Decryption
BF	SS / 80	1024	672	20	114	100
BB_1	SS / 80	2560	1184	4	8	124
SK	SS / 80	2048	672	2	6	106
Ours	SS / 80	2048	1024	28	26	130

$\spadesuit$: considering only KEM part;

small enough at a desired security level. This is the reason why we have to choose a larger system security parameter than an idealized security level when a large security loss arises at reduction. In contrast to the BF, SK, and BB_1 systems, ours has a *tight* security reduction to the standard DBDH assumption and thus we can say that our IBE system is as secure as the DBDH assumption. We notice that security of the previous three IBE systems as well as ours is all proven in the random oracle model.

When assuming the interactive version of the Gap-BDH assumption [6], BF and BB_1 systems can be modified into their respective KEMs [44, 19] that allow for shorter size of ciphertexts and the asymptotic bound for BF-KEM is improved to q_K. However, the Gap-BDH assumption has been perceived as being a non-standard one, and we have no way of comparing it with the DBDH assumption in a fair manner. Also, their corresponding DEMs need authenticated symmetric encryption, which causes a MAC tag to be included into a ciphertext. Indeed, requiring such an additional tag weakens the merit of the shorter ciphertexts. For these reasons, we do not consider their KEMs in our efficiency comparison.

Based on the result of Table 1, we give an efficiency comparison between the previous IBE systems and ours. We consider 80-bit security as a desired security level of IBE systems. As previously mentioned, BF, SK, and BB_1 systems all have concrete security loss of at least 2^{50} so that they must have a larger system security parameter than 80 bits. Obviously, it will be unfair to straightforwardly compare ours with the previous systems at the same security level. However, as a warm-up case, we first give a comparison result when instantiated in supersingular (SS) curves at an 80-bit security level. We notice that our comparison is based on Boyen's work [19] that investigates relatively estimated calculation times for various operations

[6]It is defined by the following problem: given (g, g^a, g^b, g^c, O), output $e(g,g)^{abc}$. Here, the O is a decision oracle that takes as inputs four arbitrary elements (g^x, g^y, g^z, T) in $\mathbb{G}^3 \times \mathbb{G}_T$, outputs 1 if $T = e(g,g)^{xyz}$ and otherwise 0.

Table 4: Efficiency comparison between CCA-secure IBE systems at corrected 80-bit security level

	Curves / security level	Overheads (bits)		Relative computational costs		
		Public params.	Ciphertext♠	Key extraction	Encryption	Decryption
BF	MNT / 128	1024	768	1000	2006	1503
BB_1	MNT / 128	4608	1280	200	39	1833
SK	MNT / 128	4096	768	100	36	1536
Ours	SS / 80	2048	1024	28	26	130

♠: considering only KEM part;

and representation sizes for group elements. Table 2 presents the result (see Table 7 and 8 in [19] for more details) when considering SS curves at the 80-bit security level and MNT curves at security levels 80 and 128. The numerical results of our comparison are given in Table 3. In doing so, we estimate that the encryption algorithm in our system requires three fixed-base exponentiations, one hashing $H_1(\mathsf{ID})$ for an identity ID, and one general exponentiation involving $H_1(\mathsf{ID})^s$, which makes relative encryption cost 26. Upon decryption, such hashing and general exponentiation do not need to be counted as a decryptor is able to compute $H_1(\mathsf{ID})$ for *its* identity ID beforehand. As discussed in Section 3.1, the decryption algorithm then needs to compute five fixed-base exponentiations plus 1.2 parings. Table 3 demonstrates that our system is comparable to the other IBE systems in terms of all efficiency respects, even though the relative computational costs are slightly more expensive than others. We again emphasize that the comparison above does not take into account the security losses caused by the security reductions on Table 1.

For a fairer comparison, we try to compensate for such security losses by boosting the security parameter of the underlying assumptions. There is no general rule of such compensation, but we might be able to use conjectures that were made by Bellare and Rogaway [7] for advantage functions of various block ciphers. For instance, the advantage of DES with 128-bit keys was conjectured as (roughly speaking) $\mathbf{Adv}^{\mathrm{CPA}}_{\mathrm{AES}} \leq c/2^{128}$ for some constant c. A similar approach can be made for advantages of IBE systems, so that we want to make $\mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{IBE},\mathcal{A}} \leq c/2^{80}$ at the 80-bit security level. In that case, equation (6) tells us that we have to make $\mathbf{Adv}^{\mathrm{Assumption}}_{\mathcal{B}} \leq c/2^{130}$ when considering that the security loss is bounded under 2^{50}. For simplicity, we assume that the reduction bounds in both BF and SK systems are 2^{50} (although they are much larger than it). Under these assumptions, the actual security parameter must be of 130 bits in size (approximately) to gain the real system security of an IBE system at the desired 80-bit security level. To accomplish this, we consider that the BF, BB_1, and SK systems are instantiated in MNT curves at the 128-bit security level, whereas ours is based on SS curves at the 80-bit security level as before. Table 4 shows the efficiency comparison result between CCA-secure IBE systems. We can see that the computational cost in ours becomes superior to the other systems. In particular, decryption time becomes at least 11 times faster than the others and encryption time becomes roughly 77 times faster than the BF system.

6 Discussion

6.1 A New Public Key Signature Scheme

According to Naor's observation [15], any IBE system can be converted into a secure (public key) signature scheme under the same assumptions. Naturally, we can have a signature scheme based on our IBE system, the security of which is proven under the DBDH assumption in the random oracle model. However, by using the private key structure of the IBE system, we can derive a new signature scheme whose security relies on the Computational Diffie-Hellman (CDH) assumption. Similar derivations have already been used for obtaining previous signature schemes such as BLS [17], Waters [49], and Boyen and Boneh [11]. One favorable feature of our derived signature scheme is that ours can have a *tight* security reduction to the CDH assumption. Prior to our work, BLS signature can be modified into a scheme that has a tighter security reduction to the CDH assumption with the help of the Katz-Wang technique [40]. Our signature scheme gives an alternative method for achieving such a tight security reduction. Another favorable feature is that ours is secure in the sense of *strong* unforgeability. Informally, the 'strong' means that an adversary cannot even generate a new signature for a previously-signed message. This notion is stronger than the standard notion of GMR unforgeability [37] and can be used for providing CCA security [22, 13] of various encryption schemes.

Construction. We describe our signature scheme for completeness.

Setup(k): Given a security parameter $k \in \mathbb{Z}^+$, the setup algorithm runs $\mathcal{G}(k)$ to obtain a tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$. The algorithm selects a random generator $g \in \mathbb{G}$, a random group element $u \in \mathbb{G}$, and a random exponent $\alpha \in \mathbb{Z}_p$. The algorithm sets $A = e(g,g)^\alpha$ and chooses two cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$. The public key PK (with the description of $(p, \mathbb{G}, \mathbb{G}_T, e)$) and the secret key sk are generated as

$$\mathsf{PK} = \big(g, u, A, H_1, H_2\big), \quad \mathsf{sk} = \alpha.$$

Sign$(m, \mathsf{sk}, \mathsf{PK})$: To sign a message $m \in \{0,1\}^*$, the signing algorithm does the following:

1. Choose a random $r \in \mathbb{Z}_p$.
2. Compute $\sigma_1 = g^\alpha u^r$, $\sigma_2 = g^r$, and $\text{tag} = H_2(\sigma_1, \sigma_2) \in \mathbb{Z}_p$.
3. Compute $\sigma_3 = \big(H_1(m) u^{\text{tag}}\big)^r$.
4. Output a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in \mathbb{G}^3$.

Verify(PK, m, σ): To verify a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ on a message m, the verification algorithm does the following:

1. Compute $\text{tag} = H_2(\sigma_1, \sigma_2)$.
2. Output accept if the following two equations hold:

$$e(\sigma_1, g) \stackrel{?}{=} A \cdot e(u, \sigma_2), \qquad e(\sigma_3, g) \stackrel{?}{=} e(H_1(m) u^{\text{tag}}, \sigma_2).$$

   If either check fails, output reject.

The number of pairings in the verification algorithm is four, but it can be reduced to two by the similar calculation to the one as in the decryption algorithm of the IBE system. To do this, the verification algorithm picks a random $s \in \mathbb{Z}_p$ and checks if the following equation holds:

$$e(\sigma_1 \cdot \sigma_3^s,\ g) \stackrel{?}{=} A \cdot e\big(u \cdot \big(H_1(m) u^{\text{tag}}\big)^s,\ \sigma_2\big).$$

We can prove the security of our signature scheme. The proof of Theorem 3 will be given in Appendix A, along with definitions of public key signature and strong unforgeability.

Theorem 3. *Let H_1 and H_2 be modeled as random oracles. Suppose the (t', ε')-CDH assumption holds in $\mathbb{G}$. Then our signature scheme is (t, ε, q_S)-secure in the sense of strong unforgeability under adaptive chosen message attacks, where*

$$\varepsilon' = \left(1 - \frac{2}{p}\right) \cdot \varepsilon, \qquad t \approx t' - \mathcal{O}(q_S \cdot t_e).$$

Here, t_e is the cost of an exponentiation in $\mathbb{G}$.

6.2 On Extension for Hierarchical IBE system

In a hierarchical IBE (HIBE) system [39, 35], a user's identity ID can be hierarchically scalable by delegating a private key $\mathsf{sk}_{\mathsf{ID}}$ to lower-level identities. For instance, a user with identity ID_1 can generate a private key $\mathsf{sk}_{\mathsf{ID}'}$ for a lower-level identity $\mathsf{ID}' = (\mathsf{ID}_1, \mathsf{ID}_2)$ using its own private key $\mathsf{sk}_{\mathsf{ID}_1}$. The reverse of key generation (i.e., from lower level to upper level) is not possible. This is called the 'delegation mechanism'. Using it, an HIBE system can be used for several applications including forward-secure encryption [21] and conversion for public key broadcast encryption [29]. In a security analysis for HIBE, an adversary is given the capability to request either private keys generated by a key generation center or ones delegated from upper-level identities of its choice.

One may wonder if our IBE system can be extended for supporting hierarchical identities. As far as we know, the answer is no. Since the private key structure of our system has similarity to that of Waters' tag-based dual system encryption [50], it may seem possible that a similar extension method can be applied to ours. However, the problem occurs because of the 'locked' tag values associated with upper-level identities. In the security analysis of the resulting HIBE system, an adversary requests a private key for an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_\ell)$. In order to generate a private key $\mathsf{sk}_{\mathsf{ID}}$, we have to use one of the hidden values[7] that are embedded into $\{H_i(\mathsf{ID}_i)\}$ for $i = 1, \ldots, \ell$. Assume we use a hidden value in $H_k(\mathsf{ID}_k)$ for $k \leq \ell$. In that case, the tag values corresponding to j for $j < k$ are chosen at random and mapped to H_2-query outputs in an appropriate sense. However, those random tag values are locked and cannot be changed into other different values. The adversary can still query private keys for upper-level identities, e.g., $\mathsf{ID}' = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$, in which $\mathsf{sk}_{\mathsf{ID}'}$ should be generated using those locked tags. Unfortunately, such a private key cannot be generated. One solution would be to use hidden values for each level of hierarchy, but instead it can reveal all hidden tag values that must be secretly reserved for challenge ciphertext. We leave it as an open problem to build a hierarchical version from our IBE system.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010.

[7] Those are γ_i values that appear in the H_1^{list} in the proof of Theorem 1 and 2.

[3] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 2010.

[4] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2011.

[5] Nuttapong Attrapadung, Jun Furukawa, Takeshi Gomi, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Efficient identity-based encryption with tight security reduction. In David Pointcheval, Yi Mu, and Kefei Chen, editors, *CANS*, volume 4301 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2006.

[6] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In Ueli M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[7] Mihir Bellare and Phillip Rogaway. In *Introduction to Modern Cryptography*. University of California at San Diego, 2005.

[8] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless kems. *IACR Cryptology ePrint Archive*, 2005:58, 2005.

[9] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[10] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[11] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[12] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[13] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[14] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[15] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[16] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. *IACR Cryptology ePrint Archive*, 2007:177, 2007.

[17] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[18] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[19] Xavier Boyen. A tapestry of identity-based encryption: practical frameworks compared. *IJACT*, 1(1):3–21, 2008.

[20] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006.

[21] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.

[22] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[23] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT*, pages 523–552, 2010.

[24] Liqun Chen and Zhaohui Cheng. Security proof of sakai-kasahara's identity-based encryption scheme. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 442–459. Springer, 2005.

[25] Liqun Chen, Zhaohui Cheng, John Malone-Lee, and Nigel P. Smart. An efficient id-kem based on the sakai-kasahara key construction. *IACR Cryptology ePrint Archive*, 2005:224, 2005.

[26] Liqun Chen, Zhaohui Cheng, and Nigel P. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Sec.*, 6(4):213–241, 2007.

[27] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[28] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *IACR Cryptology ePrint Archive*, 2001:108, 2001.

[29] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In Joan Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.

[30] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[31] 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.

[32] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2006.

[33] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 437–456. Springer, 2009.

[34] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[35] 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.

[36] Eu-Jin Goh and Stanislaw Jarecki. A signature scheme as secure as the diffie-hellman problem. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 401–415. Springer, 2003.

[37] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[38] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[39] 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.

[40] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 155–164. ACM, 2003.

[41] 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.

[42] Eike Kiltz and Yevgeniy Vahlis. Cca2 secure ibe: Standard model efficiency through authenticated symmetric encryption. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 221–238. Springer, 2008.

[43] Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy*, pages 273–285. IEEE Computer Society, 2010.

[44] Benoît Libert and Jean-Jacques Quisquater. Identity based encryption without redundancy. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 285–300. Springer, 2005.

[45] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[46] Ryuichi Sakai and Masao Kasahara. Id based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54, 2003.

[47] Jae Hong Seo, Tetsutaro Kobayashi, Miyako Ohkubo, and Koutarou Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 215–234. Springer, 2009.

[48] 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.

[49] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[50] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

A Public Key Signature

A.1 Definition of Public Key Signature

A (public key) signature scheme is a tuple of three algorithms $\mathcal{PKS} = (\textbf{Setup}, \textbf{Sign}, \textbf{Verify})$ over a message space $\mathcal{M}$.
Setup(k)**:** takes as input a security parameter k and outputs a public key PK and a secret key sk.
Sign(sk, PK, m)**:** takes a secret key sk, the public key PK, and a message $m \in \mathcal{M}$ as input and returns a signature σ.
Verify(PK, m, σ)**:** takes a public key PK, a message m , and a signature σ as input and returns accept or reject.

We make the standard correctness requirement: for all $(\mathsf{PK}, \mathsf{sk})$ output by **Setup** and all $m \in \mathcal{M}$, we have $\textbf{Verify}(\mathsf{PK}, m, \textbf{Sign}(\mathsf{sk}, \mathsf{PK}, m)) = \mathsf{accept}$.

A.2 Security: strong unforgeability

We give the definition of *strong* unforgeability under adaptive chosen message attacks [11]. This security notion states that an adversary cannot even generate a new signature for a previously-signed message. The security is defined via an interaction between an adversary $\mathcal{A}$ (i.e., a forger against a signature scheme) and a challenger $\mathcal{C}$:
Setup: $\mathcal{C}$ runs the setup algorithm to obtain a pair $(\mathsf{PK}, \mathsf{sk})$. It gives PK to $\mathcal{A}$ and keeps sk secret.
Query Phase : $\mathcal{A}$ issues signature queries on messages $\{m_i\}$ that can be adaptively chosen, depending on previous signatures and messages. Using sk, $\mathcal{C}$ runs the signing algorithm for each message and returns a resulting signature as a response.
Output : $\mathcal{A}$ outputs a valid signature σ^* and a message m^* such that: (1) $\textbf{Verify}(\mathsf{PK}, m^*, \sigma^*) = \mathsf{accept}$, and (2) $(m^*, \sigma^*) \notin \Sigma$, where Σ is the set of pairs (m_i, σ_i) such that σ_i was the response to a signature query.

The advantage of an adversary $\mathcal{A}$ that breaks the strong unforgeability of a signature scheme $\mathcal{PKS}$ is defined as

$$\mathbf{Adv}^{\text{suf}}_{\mathcal{PKS},\mathcal{A}}(k) = \Pr\Big[\mathcal{A} \to (m^*, \sigma^*) : \mathbf{Verify}(\mathsf{PK}, m^*, \sigma^*) = \mathsf{accept} \bigwedge (m^*, \sigma^*) \notin \Sigma\Big].$$

Definition 4. *We say that signature scheme $\mathcal{PKS}$ is (t, ε, q_S)-secure in the sense of strong unforgeability if no adversary that runs in time at most t and issues at most q_S signature queries breaks the strong unforgeability with advantage at most ε.*

A.3 CDH assumption

The Computational Diffie-Hellman (CDH) Problem: The well-known CDH problem is defined as follows: given $(g, g^a, g^b) \in \mathbb{G}^3$ as input, output g^{ab}. We say that an algorithm $\mathcal{A}$ has advantage $\mathbf{Adv}^{\text{CDH}}_{\mathbb{G},\mathcal{A}} = \varepsilon$ in solving the CDH problem in $\mathbb{G}$ if $\mathbf{Adv}^{\text{CDH}}_{\mathbb{G},\mathcal{A}} = \Pr\big[\mathcal{A}(g, g^a, g^b) = g^{ab}\big]$, where the probability is taken over the random choice of a, $b \in \mathbb{Z}_p$ and the random bits used by $\mathcal{A}$.

Definition 5. *We say that the (t, ε)-CDH assumption holds in $\mathbb{G}$ if for any polynomial time adversary $\mathcal{A}$ that runs in time at most t in solving the CDH problem in $\mathbb{G}$, we have that $\mathbf{Adv}^{CDH}_{\mathbb{G},\mathcal{A}} < \varepsilon$.*

A.4 Proof of Theorem 3

Proof. Suppose that there exists an adversary $\mathcal{A}$ which can break the strong unforgeability of our signature scheme. We can then build an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve a CDH problem in $\mathbb{G}$. On input $(g$, g^a, $g^b)$, $\mathcal{B}$ tries to output g^{ab}. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup $\mathcal{B}$ selects a random element $\delta \in \mathbb{Z}_p$ and sets $u = g^{-a} g^{\delta}$ and $A = e(g^a, g^b)$. Note that $\alpha = ab \in \mathbb{Z}_p$, which is unknown to $\mathcal{B}$. Then $\mathcal{A}$ is given the public key $\mathsf{PK} = (g, u, A, H_1, H_2)$, where H_1 and H_2 are modeled as random oracles.

Query Phase $\mathcal{A}$ issues $\{H_i\}_{i=1,2}$ and signature queries. $\mathcal{B}$ responds as follows:

$\underline{H_1 \text{ queries:}}$ To respond to H_1 queries, $\mathcal{B}$ maintains a list of tuples $< m_i, \gamma_i, \pi_i, H_1(m_i) >$ as explained below. We refer to this list as the H_1^{list}. When $\mathcal{B}$ is given a message $m_i \in \{0,1\}^*$ as an input to H_1, $\mathcal{B}$ first scans through the H_1^{list} to see if the input m_i appears in a tuple $< m_i, \gamma_i, \pi_i, H_1(m_i) >$. If it does, $\mathcal{B}$ responds with $H_1(m_i)$. Otherwise, $\mathcal{B}$ picks two random exponents $\gamma_i, \pi_i \in \mathbb{Z}_p$ and sets $H_1(m_i) = (g^a)^{\gamma_i} g^{\pi_i} \in \mathbb{G}$. $\mathcal{B}$ adds the new tuple $< m_i, \gamma_i, \pi_i, H_1(m_i) >$ to the H_1^{list} and responds with $H_1(m_i)$. Recall that the values $\{\gamma_i\}$ are information-theoretically hidden to $\mathcal{A}$'s view.

$\underline{H_2 \text{ queries:}}$ To respond to H_2 queries, $\mathcal{B}$ maintains a list of tuples $< W_i, Q_i, \mu_i >$ as explained below. We refer to this list as the H_2^{list}. When $\mathcal{B}$ is given values $(W_i, Q_i) \in \mathbb{G}^2$ as an input to H_2, $\mathcal{B}$ first scans through the H_2^{list} to see if the input (W_i, Q_i) appears in a tuple $< W_i, Q_i, \mu_i >$. If it does, $\mathcal{B}$ responds with $H_2(W_i, Q_i) = \mu_i$. Otherwise, $\mathcal{B}$ picks a random exponent $\mu_i \in \mathbb{Z}_p$ and sets $H_2(W_i, Q_i) = \mu_i$. $\mathcal{B}$ adds the new tuple $< W_i, Q_i, \mu_i >$ to the H_2^{list} and responds with $H_2(W_i, Q_i)$.

$\underline{\text{Signature queries:}}$ When $\mathcal{B}$ is given a message $m_i \in \{0,1\}^*$ as an input to a signature query, $\mathcal{B}$ selects a random exponent $r \in \mathbb{Z}_p$ and (implicitly) sets $\widetilde{r} = b + r \in \mathbb{Z}_p$. $\mathcal{B}$ computes $\sigma_{1,i} = (g^a)^{-r} (g^b)^{\delta} g^{\delta r}$ and $\sigma_{2,i} = g^b g^r$. The validity of those elements can be verified as follows:

$$\sigma_{1,i} = (g^a)^{-r} (g^b)^{\delta} g^{\delta r} = g^{ab} (g^{-a} g^{\delta})^{b+r} = g^{\alpha} u^{\widetilde{r}}, \qquad \sigma_{2,i} = g^b g^r = g^{\widetilde{r}}.$$

Next, $\mathcal{B}$ refers to the H_1^{list} to find out the tuple $< m_i, \gamma_i, \pi_i, H_1(m_i) >$. (If no tuple exists, $\mathcal{B}$ can run the H_1-query process before replying to the signature query.) At this moment, $\mathcal{B}$'s goal is to set $H_2(\sigma_{1,i}, \sigma_{2,i}) = \gamma_i$. Thus, if there is a tuple $< \sigma_{1,i}, \sigma_{2,i}, \gamma_i >$ in the H_2^{list}, $\mathcal{B}$ can continue the signature query process. As in the proof of Theorem 1, $\mathcal{B}$ can make such a (favorable) tuple always exist in the H_2^{list} as follows: whenever $\mathcal{B}$ adds a new tuple $< m_i$, γ_i, π_i, $H_1(m_i) >$ to the H_1^{list}, $\mathcal{B}$ generates a signature by choosing a random r, constructing elements $(\sigma_{1,i}, \sigma_{2,i})$ as above, setting $H_2(\sigma_{1,i}, \sigma_{2,i}) = \gamma_i$, and adding the tuple $< \sigma_{1,i}, \sigma_{2,i}, \gamma_i >$ to the H_2^{list}. On the other hand, if $H_2(\sigma_{1,i}, \sigma_{2,i})$ has already be set to μ_i, then $\mathcal{B}$ simply adds a new tuple $< m_i$, μ_i, π_i, $H_1(m_i) >$ to the H_1^{list}.

Without loss of generality, let the tuple $H_2(\sigma_{1,i}, \sigma_{2,i}) = \gamma_i$ (where γ_i is from the tuple in the H_1^{list} above) be in the H_2^{list}. $\mathcal{B}$ generates the element $\sigma_{3,i}$ as $\sigma_{3,i} = (g^b)^{\pi_i + \gamma_i \delta} g^{(\pi_i + \gamma_i \delta) r}$. The validity of $\sigma_{3,i}$ can be verified as follows:

$$\sigma_{3,i} = (g^b)^{\pi_i + \gamma_i \delta} g^{(\pi_i + \gamma_i \delta) r} = \big((g^a)^{\gamma_i} g^{\pi_i} \cdot (g^{-a+\delta})^{\gamma_i}\big)^{b+r} = \big(H_1(m_i) u^{H_2(\sigma_{1,i}, \sigma_{2,i})}\big)^{\widetilde{r}}.$$

Then, $\mathcal{B}$ responds with a signature $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i})$ for the requested message m_i.

Output At this moment, $\mathcal{A}$ outputs a valid forgery (m^*, σ^*) where $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$. There are two possible cases:

[*Case 1.*] m^* is not queried during the signature queries. $\mathcal{B}$ refers to the H_1^{list} and finds the tuple $< m^*, \gamma^*, \pi^*, H_1(m^*) >$ and it also refers to the H_2^{list} to find the tuple $< \sigma_1^*, \sigma_2^*, \mu^* >$. If $\gamma^* = \mu^*$, then $\mathcal{B}$ aborts. (We refer to this event as abort1.) The probability that the event abort1 happens during the simulation is at most $1/p$, because the exponent γ^* (regarding m^*) is predetermined and each output corresponding to a H_2 query can map to the γ^* with probability at most $1/p$. If $\gamma^* \neq \mu^*$, we know that $H_2(\sigma_1^*, \sigma_2^*) = \mu^* \neq \gamma^*$. From the two equality checks in the verification algorithm, $\mathcal{B}$ has that for some unknown exponent $\widehat{r} \in \mathbb{Z}_p$ such that $\sigma_2^* = g^{\widehat{r}}$:

$$\sigma_1^* = g^\alpha \cdot u^{\widehat{r}}, \quad \sigma_2^* = g^{\widehat{r}}, \quad \sigma_3^* = \big(H_1(m^*) u^{\mu^*}\big)^{\widehat{r}}.$$

We know that $H_1(m^*) = (g^a)^{\gamma^*} g^{\pi^*}$ and $u = g^{-a} g^\delta$, which leads to $\sigma_3^* = (g^a)^{(\gamma^* - \mu^*)\widehat{r}} g^{(\pi^* + \delta)\widehat{r}}$. Thus, $\mathcal{B}$ can obtain the value $g^{a\widehat{r}}$ by computing $\big(\sigma_3^* / (\sigma_1^*)^{\pi^* + \delta}\big)^{(\gamma^* - \mu^*)^{-1}}$. Next, we know that $\sigma_1^* = g^{ab} \cdot \big(g^{-a} g^\delta\big)^{\widehat{r}}$. It follows that $\mathcal{B}$ can obtain g^{ab} as $\sigma_1^* \cdot g^{a\widehat{r}} / (\sigma_1^*)^\delta$.

[*Case 2.*] m^* is one of queried messages. Say $m^* = m_t$ for some $t \in \{1, \ldots, q_S\}$ and the corresponding signature (that was generated in Query Phase) is $\sigma_t = (\sigma_{1,t}, \sigma_{2,t}, \sigma_{3,t})$. Again, $\mathcal{B}$ refers to the H_1^{list} and finds the tuple $< m_t, \gamma_t, \pi_t, H_1(m_t) >$ and also it refers to the H_2^{list} to find the two tuples $< \sigma_1^*, \sigma_2^*, \mu^* >$ and $< \sigma_{1,t}, \sigma_{2,t}, \gamma_t >$. Notice that γ_t regarding m_t was used to generate the signature σ_t. In this case, the two signatures should be different, i.e., $(\sigma_1^*, \sigma_2^*, \sigma_2^*) \neq (\sigma_{1,t}, \sigma_{2,t}, \sigma_{3,t})$. We consider all seven possible cases:

[*Case 2.1.*] $\sigma_1^* \neq \sigma_{1,t}$, $\sigma_2^* \neq \sigma_{2,t}$, $\sigma_3^* \neq \sigma_{3,t}$. If $\mu^* = \gamma_t$, $\mathcal{B}$ aborts. (We refer to this event as abort2.) The probability that the event abort3 happens during the simulation is at most $1/p$ as in Case 1. Otherwise, $\mathcal{B}$ can compute g^{ab} as in Case 1.

[*Case 2.2.*] $\sigma_1^* \neq \sigma_{1,t}$, $\sigma_2^* = \sigma_{2,t}$, $\sigma_3^* = \sigma_{3,t}$. In this case, we know from the first equality check in the verification algorithm that σ_1^* should be of the form $g^\alpha u^{\widetilde{r}}$ for some exponent $\widetilde{r} \in \mathbb{Z}_p$ such that $\sigma_2^* = g^{\widetilde{r}}$. Also, we had that $\sigma_{1,t} = g^\alpha u^{\widetilde{r}}$ so that it should be that $\sigma_1^* = \sigma_{1,t}$, which is a contradiction.

[*Case 2.3.*] $\sigma_1^* = \sigma_{1,t}$, $\sigma_2^* \neq \sigma_{2,t}$, $\sigma_3^* = \sigma_{3,t}$. In this case, we know from the first equality check in the verification algorithm that σ_1^* should be of the form $g^\alpha u^{r''}$ for some exponent $r'' \in \mathbb{Z}_p$ such that $\sigma_2^* = g^{r''}$. Also, we had that $\sigma_{1,t} = g^\alpha u^{\widetilde{r}}$ for some exponent $\widetilde{r} \in \mathbb{Z}_p$ such that $\sigma_{2,t} = g^{\widetilde{r}}$. Since $\sigma_2^* \neq \sigma_{2,t}$, this means that $r'' \neq \widetilde{r}$. Nevertheless, it should be that $g^\alpha u^{r''} = g^\alpha u^{\widetilde{r}}$, which is a contradiction.

[*Case 2.4.*] $\sigma_1^* = \sigma_{1,t}$, $\sigma_2^* = \sigma_{2,t}$, $\sigma_3^* \neq \sigma_{3,t}$. In this case, we know that $\mu^* = \gamma_t$. Then, from the second equality check in the verification algorithm, σ_3^* should be of the form $(H(m_t)u^{\mu^*})^{\widetilde{r}}$ for some exponent $\widetilde{r} \in \mathbb{Z}_p$ such that $\sigma_2^* = g^{\widetilde{r}}$. Also, we had that $\sigma_{3,t} = (H(m_t)u^{\gamma_t})^{\widetilde{r}}$. Since $\mu^* = \gamma_t$, it should be that $\sigma_3^* = \sigma_{3,t}$, which is a contradiction.

[*Case 2.5.*] $\sigma_1^* = \sigma_{1,t}$, $\sigma_2^* \neq \sigma_{2,t}$, $\sigma_3^* \neq \sigma_{3,t}$. This cannot happen because of the same reason for Case 2.3.

[*Case 2.6.*] $\sigma_1^* \neq \sigma_{1,t}$, $\sigma_2^* = \sigma_{2,t}$, $\sigma_3^* \neq \sigma_{3,t}$. This cannot happen because of the same reason for Case 2.2.

[*Case 2.7.*] $\sigma_1^* \neq \sigma_{1,t}$, $\sigma_2^* \neq \sigma_{2,t}$, $\sigma_3^* = \sigma_{3,t}$. If $\mu^* = \gamma_t$, we know from the second equality check in the verification algorithm that σ_3^* should be of the form $\left(H_1(m_t)u^{\mu^*}\right)^{r''}$ for some exponent $r'' \in \mathbb{Z}_p$ such that $\sigma_2^* = g^{r''}$. Also, we had that $\sigma_{3,t} = \left(H_1(m_t)u^{\gamma_t}\right)^{\widetilde{r}}$ for some exponent $\widetilde{r} \in \mathbb{Z}_p$ such that $\sigma_{2,t} = g^{\widetilde{r}}$. Since $\sigma_2^* \neq \sigma_{2,t}$, this means that $r'' \neq \widetilde{r}$. Nevertheless, it should be that $\left(H_1(m_t)u^{\mu^*}\right)^{r''} = \left(H_1(m_t)u^{\gamma_t}\right)^{\widetilde{r}}$, which is a contradiction. Otherwise, i.e., if $\mu^* \neq \gamma_t$, then $\mathcal{B}$ can compute g^{ab} as in Case 1.

Analysis. The computational time that $\mathcal{B}$ requires is dominated by exponentiations for handling q_S signature queries. Thus, the inequality concerning the computational time can easily be obtained. Next, it is easy to see that the probability that $\mathcal{B}$ aborts in the simulation is at most $\frac{2}{p}$.

Next, we can see that as long as $\mathcal{B}$ does not abort in the simulation, $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation whose distribution is identical to the distribution in a real interaction with a signer. This is because (1) the simulation of both H_1 and H_2 oracles are obviously perfect as the output values are chosen by randomly chosen values in $\mathbb{G}$ and $\mathbb{Z}_p$, respectively, and (2) the simulation of signature oracles are also perfect as each signature on a message is generated with a randomly chosen exponent $r \in \mathbb{Z}_p$ such that $\widetilde{r} = b + r$, and (3) the values $\{\gamma_i\}$ in the H_1^{list} are uniformly distributed and information-theoretically hidden from $\mathcal{A}$'s view until signatures answered by $\mathcal{B}$ are given to $\mathcal{A}$.

It follows that as long as $\mathcal{B}$ does not abort in the simulation, $\mathcal{B}$ can use $\mathcal{A}$'s advantage to break the strong unforgeability of our signature scheme straightforwardly. Then, $\mathcal{B}$'s success probability is given as follows:

$$\mathbf{Adv}_{\mathbb{G},\mathcal{B}}^{\mathrm{CDH}}(k) = \left(1 - \frac{2}{p}\right) \cdot \mathbf{Adv}_{\mathcal{PKS},\mathcal{A}}^{\mathrm{suf}}(k),$$

as required. This concludes the proof of Theorem 3. ■