Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme

Liqun Chen[1] and Zhaohui Cheng[2]

[1] Hewlett-Packard Laboratories, Bristol, UK
liqun.chen@hp.com
[2] School of Computing Science, Middlesex University
The Burroughs Hendon, London NW4 4BT, UK
m.z.cheng@mdx.ac.uk

Abstract. Identity-based encryption (IBE) is a special asymmetric encryption method where a public encryption key can be an arbitrary identifier and the corresponding private decryption key is created by binding the identifier with a system's master secret. In 2003 Sakai and Kasahara proposed a new IBE scheme, which has the potential to improve performance. However, to our best knowledge, the security of their scheme has not been properly investigated. This work is intended to build confidence in the security of the Sakai-Kasahara IBE scheme. In this paper, we first present an efficient IBE scheme that employs a simple version of the Sakai-Kasahara scheme and the Fujisaki-Okamoto transformation, which we refer to as SK-IBE. We then prove that SK-IBE has chosen ciphertext security in the random oracle model based on a reasonably well-explored hardness assumption.

1 Introduction

Shamir in 1984 [33] first formulated the concept of Identity-Based Cryptography (IBC) in which a public and private key pair is set up in a special way, i.e., the public key is the identifier (an arbitrary string) of an entity, and the corresponding private key is created by using an identity-based key extraction algorithm, which binds the identifier with a master secret of a trusted authority. In the same paper, Shamir provided the first key extraction algorithm that was based on the RSA problem, and presented an identity-based signature scheme. By using varieties of the Shamir key extraction algorithm, more identity-based signature schemes and key agreement schemes were proposed (e.g., [22, 23]). However, constructing a practical Identity-Based Encryption (IBE) scheme remained an open problem for many years.

After nearly twenty years, Boneh and Franklin [4], Cocks [15] and Sakai *et al.* [30] presented three IBE solutions in 2001. The Cocks solution is based on the quadratic residuosity. Both the Boneh and Franklin solution and the Sakai *et al.* solution are based on bilinear pairings on elliptic curves [34], and the security of their schemes is based on the Bilinear Diffie-Hellman (BDH) problem [4]. Their schemes are efficient in practice. Boneh and Franklin defined a well-formulated security model for IBE in [4]. The Boneh-Franklin scheme (BF-IBE for short) has received much attention owing to the fact that it was the first IBE scheme to have a proof of security in an appropriate model.

Both BF-IBE and the Sakai *et al.* IBE solution have a very similar private key extraction algorithm, in which an identity string is mapped to a point on an elliptic curve and then the corresponding private key is computed by multiplying the mapped point with the master private key. This key extraction algorithm was first shown in Sakai *et al.*'s work [29] in 2000 as the preparation step of an identity-based key establishment protocol. Apart from BF-IBE and the Sakai *et al.* IBE scheme [30], many other identity-based cryptographic primitives have made use of this key extraction idea, such as the signature schemes [10, 21], the authenticated key agreement schemes [11, 35], and the signcryption schemes [7, 12]. The security of these schemes were scrutinized (although some errors in a few reductions were pointed out recently but fixed as well, e.g., [14, 18]).

Based on the same tool, the bilinear pairing, Sakai and Kasahara in 2003 [28] presented a new IBE scheme using another identity-based key extraction algorithm. The idea of this algorithm can be tracked back to the work in 2002 [27]. This algorithm requires much simpler hashing and therefore improves performance. More specifically, it maps an identity to an element $h \in \mathbb{Z}_q^*$ instead of a point on an elliptic curve. The corresponding private key is generated as follow: first, compute the inverse of the sum of the master key (a random integer from $\mathbb{Z}_q^*$) and the mapped h; secondly, multiply a point of the elliptic curve (which is the generator of an order q subgroup of the group of points on the curve) with the inverse (obtained in the first step). After the initial paper was published, a number of other identity-based schemes based on this key extraction idea have been published, for examples [25, 26].

However, these schemes are either unproven or their security proof is problematic (e.g., [13]). In modern cryptography, a carefully scrutinized security reduction in a formal security model to a hardness assumption is desirable for any cryptographic scheme. Towards this end, this work is intended to build confidence in the security of the Sakai and Kasahara IBE scheme.

The remaining part of the paper is organized as follows. In next section, we recall the existing primitive, some related assumptions and the IBE security model. In Section 3, we first employ a simple version of the Sakai and Kasahara IBE scheme from [28] and the Fujisaki-Okamoto transformation [16] to present an efficient IBE scheme (we refer to it as SK-IBE). We then prove that SK-IBE has chosen ciphertext security in the random oracle model. Our proof is based on a reasonably well-explored hardness assumption. In Section 4, we show some possible improvements of SK-IBE, both on security and performance. In Section 5, we compare between SK-IBE and BF-IBE. We conclude the paper in Section 6.

2 Preliminaries

In this section, we recall the existing primitives, including bilinear pairings, some related assumptions and the security model of IBE.

2.1 Bilinear Groups and Some Assumptions

Here we review the necessary facts about bilinear maps and the associated groups using a similar notation of [5].

- $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order q.
- P_1 is a generator of $\mathbb{G}_1$ and P_2 is a generator of $\mathbb{G}_2$.
- ψ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\psi(P_2) = P_1$.
- $\hat{e}$ is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

The map $\hat{e}$ must have the following properties.

Bilinear: For all $P \in \mathbb{G}_1$, all $Q \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}$ we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
Non-degenerate: $\hat{e}(P_1, P_2) \neq 1$.
Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.

Note that following [36], we can either assume that ψ is efficiently computable or make our security proof relative to some oracle which computes ψ.

There are a batch of assumptions related to the bilinear groups. Some of them have already been used in the literature and some are new variants. We list them below and show how they are related to each other. We also correct a minor inaccuracy in stating an assumption in the literature. Recently it has come to our attention that some other related assumptions were discussed in [39].

We use a unified naming method; in particular, provided that X stands for an assumption, sX stands for a stronger assumption of X, which implies that the problem corresponding to sX would be easier than the problem corresponding to X. In the following description, $\alpha \in_R \beta$ denotes that α is an element chosen at random from a set β.

Assumption 1 (Diffie-Hellman (DH)) *For $x, y \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, given (P, xP, yP), computing xyP is hard.*

Assumption 2 (Bilinear DH (BDH) [4]) *For $x, y, z \in_R \mathbb{Z}_q^*$, $P_2 \in \mathbb{G}_2^*$, $P_1 = \psi(P_2)$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, given $(P_1, P_2, xP_2, yP_2, zP_2)$, computing $\hat{e}(P_1, P_2)^{xyz}$ is hard.*

Assumption 3 (Decisional Bilinear DH (DBDH)) *For $x, y, z, r \in_R \mathbb{Z}_q^*$, $P_2 \in \mathbb{G}_2^*$, $P_1 = \psi(P_2)$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, distinguishing between the distributions $(P_1, P_2, xP_2, yP_2, zP_2, \hat{e}(P_1, P_2)^{xyz})$ and $(P_1, P_2, xP_2, yP_2, zP_2, \hat{e}(P_1, P_2)^r)$ is hard.*

Assumption 4 (DH Inversion (k-DHI) [27]) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, given $(P, xP, x^2P, \ldots, x^kP)$, computing $\frac{1}{x}P$ is hard.*

Theorem 1 (Mitsunari et al. [27]) *DH and 1-DHI are polynomial time equivalent, i.e., if there exists a polynomial time algorithm to solve DH, then there exists a polynomial time algorithm for 1-DHI, and if there exists a polynomial time algorithm to solve 1-DHI, then there exists a polynomial time algorithm for DH.*

Assumption 5 (Collision Attack Assumption 1 (k-CAA1)) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, given $(P,\ xP,\ h_0,\ (h_1,\ \frac{1}{h_1+x}P),\ \ldots,\ (h_k,\ \frac{1}{h_k+x}P))$ where $h_i \in_R \mathbb{Z}_q^*$ and distinct for $0 \leq i \leq k$, computing $\frac{1}{h_0+x}P$ is hard.*

Theorem 2 *If there exists a polynomial time algorithm to solve (k-1)-DHI, then there exists a polynomial time algorithm for k-CAA1. If there exists a polynomial time algorithm to solve (k-1)-CAA1, then there exists a polynomial time algorithm for k-DHI.*

The proof is presented in Appendix A.

Assumption 6 (Collision Attack Assumption 2 (k-CAA2) [27]) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, given $(P, h_0, (h_1, \frac{1}{h_1+x}P), \ldots, (h_k, \frac{1}{h_k+x}P))$ where $h_i \in_R \mathbb{Z}_q^*$ and distinct for $0 \leq i \leq k$, computing $\frac{1}{h_0+x}P$ is hard.*

Mitsunari *et al.* established the relation between k-CAA2 and k-DHI (also called k-wDHA) in [27], while in the definition of k-CAA2 the value h_0 was not given as input. However, when consulting their proof of Theorem 3.5 [27], we note that h_0 has to be given as part of the problem.

Theorem 3 (Mitsunari et al. [27]) *There exists a polynomial time algorithm to solve (k-1)-DHI if and only there exists a polynomial time algorithm for k-CAA2.*

Assumption 7 (Strong CAA (k-sCAA1) [40]) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, given $(P,\ xP,\ (h_1, \frac{1}{h_1+x}P),\ \ldots,\ (h_k,\ \frac{1}{h_k+x}P))$ where $h_i \in_R \mathbb{Z}_q^*$ and distinct for $1 \leq i \leq k$, computing $(h, \frac{1}{h+x}P)$ for some $h \in \mathbb{Z}_q^*$ but $h \notin \{h_1, \ldots, h_k\}$ is hard.*

Zhang *et al.*'s short signature proof [40] and Mitsunari *et al.*'s traitor tracing scheme [27] used this assumption. However, the traitor tracing scheme was broken by Tô et al. in [37] because it was found to be in fact based on a "slightly" different assumption, which does not require to output the value of h. Obviously, if one does not have to demonstrate that he knows the value of h, the problem is not hard. He can simply choose a random element from $\mathbb{G}_1$ that is not shown in the problem as the answer, because $\mathbb{G}_1$ is of prime order q and any $r \in \mathbb{Z}_q^*$ satisfies $r = \frac{1}{h+x} \bmod q$ for some h.

Assumption 8 (Strong DH (k-sDH) [2]) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, given $(P, xP, x^2P, \ldots, x^kP)$, computing $(h, \frac{1}{h+x}P)$ where $h \in \mathbb{Z}_q^*$ is hard.*

Theorem 4 *If there exists a polynomial time algorithm to solve (k-1)-sCAA1, then there exists a polynomial time algorithm for k-sDH. If there exists a polynomial time algorithm to solve (k-1)-sDH, then there exists a polynomial time algorithm for k-sCAA1.*

The proof is presented in Appendix B.

Assumption 9 (Exponent Problem ((k+1)-EP) [40]) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, given $(P, xP, x^2P, \ldots, x^kP)$, computing $x^{k+1}P$ is hard.*

Theorem 5 (Zhang et al. [40]) *There exists a polynomial time algorithm to solve k-DHI if and only if there exists a polynomial time algorithm for (k+1)-EP.*

Assumption 10 (Bilinear DH Inversion (k-BDHI) [1]) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P_2 \in \mathbb{G}_2^*$, $P_1 = \psi(P_2)$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, given $(P_1, P_2, xP_2, x^2P_2, \ldots, x^kP_2)$, computing $\hat{e}(P_1, P_2)^{1/x}$ is hard.*

Assumption 11 (Decisional Bilinear DH Inversion (k-DBDHI)) *For an integer k, and $x, r \in_R \mathbb{Z}_q^*$, $P_2 \in \mathbb{G}_2^*$, $P_1 = \psi(P_2)$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, distinguishing between the distributions $(P_1, P_2, xP_2, x^2P_2, \ldots, x^kP_2, \hat{e}(P_1, P_2)^{1/x})$ and $(P_1, P_2, xP_2, x^2P_2, \ldots, x^kP_2, \hat{e}(P_1, P_2)^r)$ is hard.*

Theorem 6 *BDH and 1-BDHI are polynomial time equivalent, i.e., if there exists a polynomial time algorithm to solve BDH, then there exists a polynomial time algorithm for 1-BDHI, and if there exists a polynomial time algorithm to solve 1-BDHI, then there exists a polynomial time algorithm for BDH.*

The proof is presented in Appendix C.

Assumption 12 (Bilinear CAA 1 (k-BCAA1)) *For an integer k, and $x \in_R \mathbb{Z}_q^*$, $P_2 \in \mathbb{G}_2^*$, $P_1 = \psi(P_2)$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, given $(P_1, P_2, xP_2, h_0, (h_1, \frac{1}{h_1+x}P_2), \ldots, (h_k, \frac{1}{h_k+x}P_2))$ where $h_i \in_R \mathbb{Z}_q^*$ and distinct for $0 \le i \le k$, computing $\hat{e}(P_1, P_2)^{1/(x+h_0)}$ is hard.*

Theorem 7 *If there exists a polynomial time algorithm to solve (k-1)-BDHI, then there exists a polynomial time algorithm for k-BCAA1. If there exists a polynomial time algorithm to solve (k-1)-BCAA1, then there exists a polynomial time algorithm for k-BDHI.*

The proof is presented in Appendix D.

The relation among these assumptions can be described by Fig. 1. In the literature, the k-DBDHI assumption was used in [1] to construct a selective-identity secure IBE scheme (see next section for definition) without random oracles [6] and k-sDH is used to construct a short signature [2] without random oracles, while k-sCAA1 is used by [40] to construct a short signature with random oracles and to build a traitor tracing scheme [27].

2.2 IBE Schemes and Their Security Model

Let k be a security parameter, and $\mathbb{M}$ and $\mathbb{C}$ denote the message and ciphertext spaces respectively. An IBE scheme is specified by four polynomial time algorithms:

$$
\begin{tikzcd}
\text{k-sCAA1} \arrow[r, dotted, leftrightarrow] & \text{k-sDH} & \text{(k+1)-EP} \arrow[d, leftrightarrow] \arrow[dl, leftrightarrow] \\
\text{k-CAA1} \arrow[u] \arrow[d] \arrow[r, dotted, leftrightarrow] & \text{k-DHI} \arrow[u] \arrow[d] \arrow[r, leftrightarrow] & \text{(k+1)-CAA2} \\
\text{k-BCAA1} \arrow[r, dotted, leftrightarrow] & \text{k-BDHI} \arrow[r] & \text{k-DBDHI}
\end{tikzcd}
$$

k-A $\longrightarrow$ k-B: if k-A is polynomial-time solvable, so is k-B;
k-A $\dashrightarrow$ k-B: if (k-1)-A is polynomial-time solvable, so is k-B.

Fig. 1. Relation among the assumptions.

- **Setup** takes as input 1^k, and returns a master public key M_{pk} and a master secret key M_{sk};
- **Extract** takes as input M_{pk}, M_{sk} and $ID_A \in \{0,1\}^*$, an identifier string for entity A, and returns the associated private key d_A;
- **Encrypt** takes as input M_{pk}, ID_A and a message $m \in \mathbb{M}$, and returns a ciphertext $C \in \mathbb{C}$; and
- **Decrypt** takes as input M_{pk}, ID_A, d_A and C, and returns the corresponding value of the plaintext m or a failure symbol $\perp$.

The security of an IBE scheme is defined by the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formalized in [4].

- Setup. $\mathcal{C}$ takes a security parameter k and runs the Setup algorithm. It gives $\mathcal{A}$ M_{pk} and keeps M_{sk} to itself.
- Phase 1. $\mathcal{A}$ issues queries as one of follows:
  - Extraction query on ID_i. $\mathcal{C}$ runs the Extract algorithm to generate d_{ID_i} and passes it to $\mathcal{A}$.
  - Decryption query on (ID_i, C_i). $\mathcal{C}$ decrypts the ciphertext by finding d_{ID_i} first (through running Extract if necessary), and then running the Decrypt algorithm. It responds with the resulting plaintext.
- Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts $m_0, m_1 \in \mathbb{M}$, and an identity ID_{ch} on which it wishes to be challenged. The only constraint is that $\mathcal{A}$ must not have queried the extraction oracle on ID_{ch} in Phase 1. $\mathcal{C}$ picks a random bit $b \in \{0,1\}$ and sets C_{ch}=Encrypt(M_{pk}, ID_{ch}, m_b) $\in \mathbb{C}$. It sends C_{ch} as the challenge to $\mathcal{A}$.
- Phase 2. $\mathcal{A}$ issues more queries as in Phase 1 but with two restrictions: (1) Extraction queries cannot be issued on ID_{ch}; (2) Decryption queries cannot be issued on (ID_{ch}, C_{ch}).
- Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins the game if $b' = b$.

We refer to this type of adversary as an IND-ID-CCA adversary. If $\mathcal{A}$ cannot ask decryption queries, we call it an IND-ID-CPA adversary. The advantage of an IND-ID-CCA adversary $\mathcal{A}$ against an IBE scheme $\mathcal{E}_{\mathtt{ID}}$ is the function of security parameter k defined as: $Adv_{\mathcal{E}_{\mathtt{ID}},\mathcal{A}}(k) = |Pr[b' = b] - 1/2|$.

Definition 1 *An identity-based encryption scheme* $\mathcal{E}_{\mathtt{ID}}$ *is IND-ID-CCA secure if for any IND-ID-CCA adversary,* $Adv_{\mathcal{E}_{\mathtt{ID}},\mathcal{A}}(k)$ *is negligible.*

Canetti *et al.* formulated a weaker IBE notion, selective-identity adaptive chosen ciphertext attacks secure scheme (IND-sID-CCA for short), in which, an adversary has to commit the identity on which it wants to be challenged before it sees the public system parameters (the master public key) [8]. The latest work [19] provides some formal security analysis of this formulation.

3 SK-IBE

In this section, we investigate the security strength of SK-IBE. We choose the simplest variant of the Sakai and Kasahara IBE scheme [28] as the basic version of SK-IBE. This basic version was also described by Scott in [31]. To achieve security against adaptive chosen ciphertext attacks, we make use of the Fujisaki-Okamoto transformation [16] as it was used in BF-IBE [4].

3.1 Scheme

SK-IBE is specified by four polynomial time algorithms:

Setup. Given a security parameter k, the parameter generator follows the steps.

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order q, an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Pick a random generator $P_2 \in \mathbb{G}_2^*$ and set $P_1 = \psi(P_2)$.
2. Pick a random $s \in \mathbb{Z}_q^*$ and compute $P_{pub} = sP_1$.
3. Pick four cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_T \to \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$ for some integer $n > 0$.

The message space is $\mathbb{M} = \{0,1\}^n$. The ciphertext space is $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$. The master public key is $M_{pk} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi, \hat{e}, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4)$, and the master secret key is $M_{sk} = s$.

Extract. Given an identifer string $ID_A \in \{0,1\}^*$ of entity A, M_{pk} and M_{sk}, the algorithm returns $d_A = \frac{1}{s+H_1(ID_A)}P_2$.

Remark 1 The result of the Extract algorithm is a short signature d_A on the message ID_A signed under the private signing key s. As proved in Theorem 3 of [40], this signature scheme is existentially unforgeable under chosen-message attack [20] in the random oracle model [6], provided that the k-sCAA1 assumption is sound in $\mathbb{G}_2$.

Encrypt. Given a plaintext $m \in \mathbb{M}$, ID_A and M_{pk}, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m)$.
2. Compute $Q_A = H_1(ID_A)P_1 + P_{pub}$, $g^r = \hat{e}(P_1, P_2)^r$.
3. Set the ciphertext to $C = (rQ_A, \sigma \oplus H_2(g^r), m \oplus H_4(\sigma))$.

Remark 2 In the Encrypt algorithm, the pairing $g = \hat{e}(P_1, P_2)$ is fixed and can be pre-computed. It can further be treated as a system public parameter. Therefore, no pairing computation is required in Encrypt.

Decrypt. Given a ciphertext $C = (U, V, W) \in \mathbb{C}$, ID_A, d_A and M_{pk}, follow the steps:

1. Compute $g' = \hat{e}(U, d_A)$ and $\sigma' = V \oplus H_2(g')$
2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m')$.
3. If $U \neq r'(H_1(ID_A)P_1 + P_{pub})$, output $\perp$, else return m' as the plaintext.

3.2 Security of SK-IBE

Now we evaluate the security of SK-IBE. We prove that the security of SK-IBE can reduce to the hardness of the k-BDHI problem. The reduction is similar to the proof of BF-IBE [4]. However, we will take into account the error in Lemma 4.6 of [4] corrected by Galindo [18].

Theorem 8 *SK-IBE is secure against IND-ID-CCA adversaries provided that $H_i (1 \leq i \leq 4)$ are random oracles and the k-BDHI assumption is sound. Specifically, suppose there exists an IND-ID-CCA adversary $\mathcal{A}$ against SK-IBE that has advantage $\epsilon(k)$ and running time $t(k)$. Suppose also that during the attack $\mathcal{A}$ makes at most q_D decryption queries and at most q_i queries on H_i for $1 \leq i \leq 4$ respectively (note that H_i can be queried directly by $\mathcal{A}$ or indirectly by an extraction query, a decryption query or the challenge operation). Then there exists an algorithm $\mathcal{B}$ to solve the q_1-BDHI problem with advantage $Adv_{\mathcal{B}}(k)$ and running time $t_{\mathcal{B}}(k)$ where*

$$
\begin{aligned}
Adv_{\mathcal{B}}(k) &\geq \tfrac{1}{q_2(q_3+q_4)}[(\tfrac{\epsilon(k)}{q_1}+1)(1-\tfrac{2}{q})^{q_D}-1] \\
t_{\mathcal{B}}(k) &\leq t(k) + O((q_3+q_4)\cdot(n+\log q) + q_D \cdot \mathcal{T}_1 + q_1^2 \cdot \mathcal{T}_2 + q_D \cdot \chi)
\end{aligned}
$$

where χ is the time of computing pairing, $\mathcal{T}_i$ is the time of a multiplication operation in $\mathbb{G}_i$, and q is the order of $\mathbb{G}_1$ and n is the length of σ. We assume the computation complexity of ψ is trivial.

Proof: The theorem follows immediately by combining Lemma 1, 2 and 3. The reduction with three steps can be sketched as follow. First we prove that if there exists an IND-ID-CCA adversary, who is able to break SK-IBE by launching the adaptive chosen ciphertext attacks as defined in the security model of Section 2.2, then there exists an IND-CCA adversary to break the $\mathbf{BasicPub}^{\mathbf{hy}}$ scheme defined in Lemma 1 with the adaptive chosen ciphertext attacks. Second, if such IND-CCA adversary exists, then we show (in Lemma 2) that there must be an IND-CPA adversary that breaks the corresponding **BasicPub** scheme by merely launching the chosen plaintext attacks. Finally, in Lemma 3 we prove that if the **BasicPub** scheme is not secure against an IND-CPA adversary, then the corresponding k-BDHI assumption is flawed. □

Lemma 1 *Suppose that* H_1 *is a random oracle and that there exists an IND-ID-CCA adversary* $\mathcal{A}$ *against SK-IBE with advantage* $\epsilon(k)$ *which makes at most* q_1 *distinct queries to* H_1 *(note that* H_1 *can be queried directly by* $\mathcal{A}$ *or indirectly by an extraction query, a decryption query or the challenge operation). Then there exists an IND-CCA adversary* $\mathcal{B}$ *which runs in time* $O(time(\mathcal{A}) + q_D \cdot (\chi + \mathcal{T}_1))$ *against the following* $\mathbf{BasicPub^{hy}}$ *scheme with advantage at least* $\epsilon(k)/q_1$ *where* χ *is the time of computing pairing and* $\mathcal{T}_1$ *is the time of a multiplication operation in* $\mathbb{G}_1$*.*

$\mathbf{BasicPub^{hy}}$ *is specified by three algorithms:* **keygen**, **encrypt** *and* **decrypt***.*
keygen: *Given a security parameter* k*, the parameter generator follows the steps.*

1. *Identical with step 1 in Setup algorithm of SK-IBE.*
2. *Pick a random* $s \in \mathbb{Z}_q^*$ *and compute* $P_{pub} = sP_1$*. Randomly choose different elements* $h_i \in \mathbb{Z}_q^*$ *and compute* $\frac{1}{h_i+s}P_2$ *for* $0 \le i < q_1$*.*
3. *Pick three cryptographic hash functions:* $H_2 : \mathbb{G}_T \to \{0,1\}^n$*,* $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ *and* $H_4 : \{0,1\}^n \to \{0,1\}^n$ *for some integer* $n > 0$*.*

The message space is $\mathbb{M} = \{0,1\}^n$*. The ciphertext space is* $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$*. The public key is* $K_{pub} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi, \hat{e}, n, P_1, P_2, P_{pub}, h_0, (h_1, \frac{1}{h_1+s}P_2), \ldots, (h_i, \frac{1}{h_i+s}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2), H_2, H_3, H_4)$ *and the private key is* $d_A = \frac{1}{h_0+s}P_2$*. Note that* $\hat{e}(h_0P_1 + P_{pub}, d_A) = \hat{e}(P_1, P_2)$*.*
encrypt: *Given a plaintext* $m \in \mathbb{M}$ *and the public key* K_{pub}*,*

1. *Pick a random* $\sigma \in \{0,1\}^n$ *and compute* $r = H_3(\sigma, m)$*, and* $g^r = \hat{e}(P_1, P_2)^r$*.*
2. *Set the ciphertext to* $C = (r(h_0P_1 + P_{pub}), \sigma \oplus H_2(g^r), m \oplus H_4(\sigma))$*.*

decrypt: *Given a ciphertext* $C = (U, V, W)$*,* K_{pub}*, and the private key* d_A*, follow the steps.*

1. *Compute* $g' = \hat{e}(U, d_A)$ *and* $\sigma' = V \oplus H_2(g')$*,*
2. *Compute* $m' = W \oplus H_4(\sigma')$ *and* $r' = H_3(\sigma', m')$*,*
3. *If* $U \neq r'(h_0P_1 + P_{pub})$*, reject the ciphertext, else return* m' *as the plaintext.*

Proof: We construct an IND-CCA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage against $\mathbf{BasicPub^{hy}}$. The game between a challenger $\mathcal{C}$ and the adversary $\mathcal{B}$ starts with the challenger first generating a random public key K_{pub} by running algorithm keygen of $\mathbf{BasicPub^{hy}}$ ($\log q_1$ is part of the security parameter of $\mathbf{BasicPub^{hy}}$). The result is $K_{pub} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi, \hat{e}, n, P_1, P_2, P_{pub}, h_0, (h_1, \frac{1}{h_1+s}P_2), \ldots, (h_i, \frac{1}{h_i+s}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2), H_2, H_3, H_4)$, where $P_{pub} = sP_1$ with $s \in \mathbb{Z}_q^*$, and the private key $d_A = \frac{1}{h_0+s}P_2$. The challenger passes K_{pub} to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ mounts an IND-CCA attack on the $\mathbf{BasicPub^{hy}}$ scheme with the public key K_{pub} using the help of $\mathcal{A}$ as follows.

$\mathcal{B}$ chooses an index I with $1 \le I \le q_1$ and simulates the algorithm Setup of SK-IBE for $\mathcal{A}$ by supplying $\mathcal{A}$ with the SK-IBE master public key $M_{pk} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi, \hat{e}, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4)$ where H_1 is a random oracle controlled by $\mathcal{B}$. The master secret key M_{sk} for this cryptosystem is s, although

$\mathcal{B}$ does not know this value. Adversary $\mathcal{A}$ can make queries on H_1 at any time. These queries are handled by the following algorithm H_1**-query**.

H_1**-query** (ID_i)**:** $\mathcal{B}$ maintains a list of tuples (ID_i, h_i, d_i) indexed by ID_i as explained below. We refer to this list as H_1^{list}. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point ID_i, $\mathcal{B}$ responds as follows:

1. If ID_i already appears on the H_1^{list} in a tuple (ID_i, h_i, d_i), then $\mathcal{B}$ responds with $H_1(ID_i) = h_i$.
2. Otherwise, if the query is on the I-th distinct ID and $\perp$ is not used as d_i (this could be inserted by the challenge operation specified later) by any existing tuple, then $\mathcal{B}$ stores $(ID_I, h_0, \perp)$ into the tuple list and responds with $H_1(ID_I) = h_0$.
3. Otherwise, $\mathcal{B}$ selects a random integer $h_i(i > 0)$ from K_{pub} which has not been chosen by $\mathcal{B}$ and stores $(ID_i, h_i, \frac{1}{h_i+s}P_2)$ into the tuple list. $\mathcal{B}$ responds with $H_1(ID_i) = h_i$.

Phase 1: $\mathcal{A}$ launches Phase 1 of its attack, by making a series of requests, each of which is either an extraction or a decryption query. $\mathcal{B}$ replies to these requests as follows.

Extraction query (ID_i)**:** $\mathcal{B}$ first looks through list H_1^{list}. If ID_i is not on the list, then $\mathcal{B}$ queries $H_1(ID_i)$. $\mathcal{B}$ then checks the value d_i: if $d_i \neq \perp$, $\mathcal{B}$ responds with d_i; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

Decryption query (ID_i, C_i)**:** $\mathcal{B}$ first looks through list H_1^{list}. If ID_i is not on the list, then $\mathcal{B}$ queries $H_1(ID_i)$. If $d_i = \perp$, then $\mathcal{B}$ sends the decryption query $C_i = (U, V, W)$ to $\mathcal{C}$ and simply relays the plaintext got from $\mathcal{C}$ to $\mathcal{A}$ directly. Otherwise, $\mathcal{B}$ decrypts the ciphertext by first computing $g' = \hat{e}(U, d_i)$, then querying $\zeta = H_2(g')$ (H_2 is controlled by $\mathcal{C}$), and computing $\sigma' = V \oplus \zeta, m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m')$. Finally $\mathcal{B}$ checks the validity of C_i as step 3 of algorithm decrypt and returns m', if C_i is valid, otherwise the failure symbol $\perp$.

Challenge: At some point, $\mathcal{A}$ decides to end Phase 1 and picks ID_{ch} and two messages (m_0, m_1) of equal length on which it wants to be challenged. Based on the queries on H_1 so far, $\mathcal{B}$ responds differently.

1. If the I-th query on H_1 has been issued,
   - if $ID_I = ID_{ch}$ (and so $d_{ch} = \perp$), $\mathcal{B}$ continues;
   - otherwise, $\mathcal{B}$ aborts the game (**Event 2**).
2. Otherwise,
   - if the tuple corresponding to ID_{ch} is on list H_1^{list} (and so $d_{ch} \neq \perp$), then $\mathcal{B}$ aborts the game (**Event 3**);
   - otherwise, $\mathcal{B}$ inserts the tuple $(ID_{ch}, h_0, \perp)$ into the list and continues (this operation is treated as an H_1 query in the simulation).

Note that after this point, it must have $H_1(ID_{ch}) = h_0$ and $d_{ch} = \perp$. $\mathcal{B}$ passes $\mathcal{C}$ the pair (m_0, m_1) as the messages on which it wishes to be challenged. $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$, encrypts m_b and responds with the ciphertext $C_{ch} = (U', V', W')$. Then $\mathcal{B}$ forwards C_{ch} to $\mathcal{A}$.

Phase 2: $\mathcal{B}$ continues to respond to requests in the same way as it did in Phase 1. Note that following the rules, the adversary will not issue the extraction query

on ID_{ch} (for which $d_{ch} = \perp$) and the decryption query on (ID_{ch}, C_{ch}). And so, $\mathcal{B}$ always can answer other queries without aborting the game.
Guess: $\mathcal{A}$ makes a guess b' for b. $\mathcal{B}$ outputs b' as its own guess.

Claim: If the algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}$'s view is identical to its view in the real attack.

Proof: $\mathcal{B}$'s responses to H_1 queries are uniformly and independently distributed in $\mathbb{Z}_q^*$ as in the real attack because of the behavior of algorithm keygen of the $\mathbf{BasicPub}^{\mathbf{hy}}$ scheme. All responses to $\mathcal{A}$'s requests are valid, if $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext $C_{ch} = (U', V', W')$ is a valid encryption in SK-IBE for m_b where $b \in \{0, 1\}$ is random.

The remaining problem is to calculate the probability that $\mathcal{B}$ does not abort during simulation. Algorithm $\mathcal{B}$ could abort when one of the following events happens: (1) **Event 1**, denoted as $\mathcal{H}_1$: $\mathcal{A}$ queried a private key which is represented by $\perp$ at some point. Recall that only one private key is represented by $\perp$ in the whole simulation which could be inserted in an H_1 query (as the private key of ID_I) in Phase 1 or in the challenge phase (as the private key of ID_{ch}). Because of the rules of the game, the adversary will not query the private key of ID_{ch}. Hence, this event only happens when the adversary extracted the private key of $ID_I \neq ID_{ch}$, meanwhile $d_I = \perp$, i.e., $ID_I \neq ID_{ch}$ and $H_1(ID_I)$ was queried in Phase 1; (2) **Event 2**, denoted as $\mathcal{H}_2$: the adversary wants to be challenged on an identity $ID_{ch} \neq ID_I$ and $H_1(ID_I)$ was queried in Phase 1; (3) **Event 3**, denoted as $\mathcal{H}_3$: the adversary wants to be challenged on an identity $ID_{ch} \neq ID_I$ and $H_1(ID_I)$ was queried in Phase 2.

Notice that all the three events imply **Event 4**, denoted by $\mathcal{H}_4$, that the adversary did not choose ID_I as the challenge identity. Hence we have

$$Pr[\mathcal{B} \text{ does not abort}] = Pr[\neg\mathcal{H}_1 \wedge \neg\mathcal{H}_2 \wedge \neg\mathcal{H}_3] \geq Pr[\neg\mathcal{H}_4] \geq 1/q_1.$$

So, the lemma follows. □

Remark 3 If an adversary only engages in the selective-identity adaptive chosen ciphertext attack game, the reduction could be tighter ($\mathcal{B}$ has the advantage $\epsilon(k)$ as $\mathcal{A}$), because $\mathcal{B}$ now knows exactly which identity should be hashed to h_0, so the game will never abort. Note that, in such game, $\mathcal{B}$ can pass the SK-IBE system parameters (the master public key) to $\mathcal{A}$ first, then $\mathcal{A}$ commits an identity ID_{ch} before issuing any oracle query. Hence the reduction could still be tightened to a stronger formulation than the one in [8] (see the separation in [19]).

Lemma 2 *Let H_3, H_4 be random oracles. Let $\mathcal{A}$ be an IND-CCA adversary against* $\mathbf{BasicPub}^{\mathbf{hy}}$ *defined in Lemma 1 with advantage $\epsilon(k)$. Suppose $\mathcal{A}$ has running time $t(k)$, makes at most q_D decryption queries, and makes q_3 and q_4 queries to H_3 and H_4 respectively. Then there exists an IND-CPA adversary $\mathcal{B}$ against the following* **BasicPub** *scheme, which is specified by three algorithms:* **keygen**, **encrypt** *and* **decrypt**.
keygen: *Given a security parameter k, the parameter generator follows the steps.*

1. *Identical with step 1 in algorithm keygen of* **BasicPub**$^{\mathbf{hy}}$.
2. *Identical with step 2 in algorithm keygen of* **BasicPub**$^{\mathbf{hy}}$.
3. *Pick a cryptographic hash function* $H_2 : \mathbb{G}_T \rightarrow \{0,1\}^n$ *for some integer* $n > 0$.

The message space is $\mathbb{M} = \{0,1\}^n$. *The ciphertext space is* $\mathbb{C} = \mathbb{G}_1^* \times \{0,1\}^n$. *The public key is* $K_{pub} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi, \hat{e}, n, P_1, P_2, P_{pub}, h_0, (h_1, \frac{1}{h_1+s}P_2), \ldots, (h_i, \frac{1}{h_i+s}P_2), \ldots, (h_{q_1-1}, \frac{1}{h_{q_1-1}+s}P_2), H_2)$ *and the private key is* $d_A = \frac{1}{h_0+s}P_2$. *Again it has* $\hat{e}(h_0P_1 + P_{pub}, d_A) = \hat{e}(P_1, P_2)$.
encrypt: *Given a plaintext* $m \in \mathbb{M}$ *and the public key* K_{pub}, *choose a random* $r \in \mathbb{Z}_q^*$ *and compute ciphertext* $C = (r(h_0P_1 + P_{pub}), m \oplus H_2(g^r))$ *where* $g^r = \hat{e}(P_1, P_2)^r$.
decrypt: *Given a ciphertext* $C = (U, V)$, K_{pub}, *and the private key* d_A, *compute* $g' = \hat{e}(U, d_A)$ *and plaintext* $m = V \oplus H_2(g')$.

with advantage $\epsilon_1(k)$ and running time $t_1(k)$ where

$$\begin{aligned} \epsilon_1(k) &\geq \tfrac{1}{2(q_3+q_4)}[(\epsilon(k)+1)(1-\tfrac{2}{q})^{q_D} - 1] \\ t_1(k) &\leq t(k) + O((q_3+q_4)\cdot(n+\log q)). \end{aligned}$$

Proof: This lemma follows from the result of the Fujisaki-Okamoto transformation [16] and BF-IBE has a similar result (Theorem 4.5 [4]). We note that it is assumed that n and $\log q$ are of similar size in [4]. □

Lemma 3 *Let* H_2 *be a random oracle. Suppose there exists an IND-CPA adversary* $\mathcal{A}$ *against the* **BasicPub** *defined in Lemma 2 which has advantage* $\epsilon(k)$ *and queries* H_2 *at most* q_2 *times. Then there exists an algorithm* $\mathcal{B}$ *to solve the* q_1*-BDHI problem with advantage at least* $2\epsilon(k)/q_2$ *and running time* $O(time(\mathcal{A}) + q_1^2 \cdot \mathcal{T}_2)$ *where* $\mathcal{T}_2$ *is the time of a multiplication operation in* $\mathbb{G}_2$.

Proof: Algorithm $\mathcal{B}$ is given as input a random q_1-BDHI instance $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi, \hat{e}, P_1, P_2, xP_2, x^2P_2, \ldots x^{q_1}P_2)$ where x is a random element from $\mathbb{Z}_q^*$. Algorithm $\mathcal{B}$ finds $\hat{e}(P_1, P_2)^{1/x}$ by interacting with $\mathcal{A}$ as follows:

Algorithm $\mathcal{B}$ first simulates algorithm keygen of **BasicPub**, which was defined in Lemma 2, to create the public key as below. A similar approach is used in [1, 2].

1. Randomly choose different $h_0, \ldots, h_{q_1-1} \in \mathbb{Z}_q^*$ and let $f(z)$ be the polynomial $f(z) = \prod_{i=1}^{q_1-1}(z+h_i)$. Reformulate f to get $f(z) = \sum_{i=0}^{q_1-1} c_i z^i$. The constant term c_0 is non-zero because $h_i \neq 0$ and c_i are computable from h_i.
2. Compute $Q_2 = \sum_{i=0}^{q_1-1} c_i x^i P_2 = f(x)P_2$ and $xQ_2 = \sum_{i=0}^{q_1-1} c_i x^{i+1} P_2 = xf(x)P_2$.
3. Check that $Q_2 \in \mathbb{G}_2^*$. If $Q_2 = 1_{\mathbb{G}_2}$, then there must exist an $h_i = -x$ which can be easily identified, and so, $\mathcal{B}$ solves the q_1-BDHI problem directly. Otherwise, $\mathcal{B}$ computes $Q_1 = \psi(Q_2)$ and continues.
4. Compute $f_i(z) = f(z)/(z+h_i) = \sum_{j=0}^{q_1-2} d_j z^j$ and $\frac{1}{x+h_i}Q_2 = f_i(x)P_2 = \sum_{j=0}^{q_1-2} d_j x^j P_2$ for $1 \leq i < q_1$.

5. Set $T' = \sum_{i=1}^{q_1-1} c_i x^{i-1} P_2$ and compute $T_0 = \hat{e}(\psi(T'), Q_2 + c_0 P_2)$.
6. Now, $\mathcal{B}$ passes $\mathcal{A}$ the public key $K_{pub} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi, \hat{e}, n, Q_1, Q_2, xQ_1 - h_0Q_1, h_0, (h_1+h_0, \frac{1}{h_1+x}Q_2), \ldots, (h_i+h_0, \frac{1}{h_i+x}Q_2), \ldots, (h_{q_1-1}+h_0, \frac{1}{h_{q_1-1}+x}Q_2), H_2)$ (i.e., setting $P_{pub} = xQ_1 - h_0Q_1$), and the private key is $d_A = \frac{1}{x}Q_2$ which $\mathcal{B}$ does not know. H_2 is a random oracle controlled by $\mathcal{B}$. Note that $\hat{e}((h_i + h_0)Q_1 + P_{pub}, \frac{1}{h_i+x}Q_2) = \hat{e}(Q_1, Q_2)$ for $i = 1, \ldots, q_1 - 1$ and $\hat{e}(h_0Q_1 + P_{pub}, d_A) = \hat{e}(Q_1, Q_2)$. Hence K_{pub} is a valid public key of **BasicPub**.

Now $\mathcal{B}$ starts to respond to queries as follows.
H_2-query (X_i): At any time algorithm $\mathcal{A}$ can issue queries to the random oracle H_2. To respond to these queries $\mathcal{B}$ maintains a list of tuples called H_2^{list}. Each entry in the list is a tuple of the form (X_i, ζ_i) indexed by X_i. To respond to a query on X_i, $\mathcal{B}$ does the following operations:

1. If on the list there is a tuple indexed by X_i, then $\mathcal{B}$ responds with ζ_i.
2. Otherwise, $\mathcal{B}$ randomly chooses a string $\zeta_i \in \{0,1\}^n$ and inserts a new tuple (X_i, ζ_i) to the list. It responds to $\mathcal{A}$ with ζ_i.

Challenge: Algorithm $\mathcal{A}$ outputs two messages (m_0, m_1) of equal length on which it wants to be challenged. $\mathcal{B}$ chooses a random string $R \in \{0,1\}^n$ and a random element $r \in \mathbb{Z}_q^*$, and defines $C_{ch} = (U, V) = (rQ_1, R)$. $\mathcal{B}$ gives C_{ch} as the challenge to $\mathcal{A}$. Observe that the decryption of C_{ch} is

$$V \oplus H_2(\hat{e}(U, d_A)) = R \oplus H_2(\hat{e}(rQ_1, \frac{1}{x}Q_2)).$$

Guess: After algorithm $\mathcal{A}$ outputs its guess, $\mathcal{B}$ picks a random tuple (X_i, ζ_i) from H_2^{list}. $\mathcal{B}$ first computes $T = X_i^{1/r}$, and then returns $(T/T_0)^{1/c_0^2}$. Note that $\hat{e}(P_1, P_2)^{1/x} = (T/T_0)^{1/c_0^2}$ if $T = \hat{e}(Q_1, Q_2)^{1/x}$.

Let $\mathcal{H}$ be the event that algorithm $\mathcal{A}$ issues a query for $H_2(\hat{e}(rQ_1, \frac{1}{x}Q_2))$ at some point during the simulation above. Using the same methods in [4], we can prove the following two claims:
Claim 1: $\Pr[\mathcal{H}]$ in the simulation above is equal to $\Pr[\mathcal{H}]$ in the real attack.
Claim 2: In the real attack we have $Pr[\mathcal{H}] \geq 2\epsilon(k)$.

Following from the above two claims, we have that $\mathcal{B}$ produces the correct answer with probability at least $2\epsilon(k)/q_2$. □

Remark 4 In the proof, $\mathcal{B}$'s simulation of algorithm keygen of **BasicPub** is similar to the preparation step in Theorem 5.1 [1] (both follow the method in [27]. Note that in [1] ψ is an identity map, so $Q = Q_1 = Q_2$). However, the calculation of T_0 in [1] is incorrect, and should be computed as $T_0 = \prod_{i=0}^{q-1} \prod_{j=0}^{q-2} \hat{e}(g^{(\alpha^i)}, g^{(\alpha^j)})^{c_i c_{j+1}} \cdot \prod_{j=0}^{q-2} \hat{e}(g, g^{(\alpha^j)})^{c_0 c_{j+1}}$.

This completes the proof of Theorem 8.

4 Possible Improvements of SK-IBE

SK-IBE can be improved both on computation performance and security reduction. The only two known bilinear pairing instances so far are the Weil pairing and Tate pairing on elliptic curves (and hyperelliptic curves) [34]. When implementing these pairings, some special structures of these pairings can be exploited to improve the performance. As noticed by Scott and Barreto [32], the Tate pairing can be compressed when the curve has the characteristic 3 or greater than 3. The compressing technique not only can reduce the size of pairing, but also can speed up the computation of pairing and the exponentiation in $\mathbb{G}_T$. Pointed by Galindo [18], an improved Fujisaki-Okamoto's transformation [17] has a tighter security reduction. Using the trick played in [24], the reduction can be further tightened by including the point rQ_A in H_2 (this also removes the potential ambiguity introduced by the compressed pairing). So, combined with these two improvements, a faster scheme (SK-IBE2) with better security reduction can be specified as follow.

Setup. Identical with SK-IBE, except that H_4 is not required and $H_2 : \mathbb{G}_1 \times \mathbb{F} \to \{0,1\}^{2n}$, where $\mathbb{F}$ depends on the used compressed pairing (see [32] for details).

Extract. Identical with SK-IBE.

Encrypt. Given a plaintext $m \in \mathbb{M}(\{0,1\}^n)$, the identity ID_A of entity A and the master public key M_{pk}, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m)$.
2. Compute $Q_A = H_1(ID_A)P_1 + P_{pub}$, $\varphi(g^r) = \varphi(\hat{e}(P_1, P_2)^r)$, where φ is the pairing compressing algorithm as specified in [32]. Note that φ and $\hat{e}$ can be computed by a single algorithm, so to improve the computation performance [32].
3. Set the ciphertext to $C = (rQ_A, (m\|\sigma) \oplus H_2(rQ_A, \varphi(g^r)))$.

Decrypt. Given a ciphertext $(U, V) \in \mathbb{C}$, the identity ID_A, the private key d_A and M_{pk}, follow the steps:

1. Compute $\varphi(g') = \varphi(\hat{e}(U, d_A))$ and $m'\|\sigma' = V \oplus H_2(U, \varphi(g'))$.
2. Compute $r' = H_3(\sigma', m')$. If $U \neq r'(H_1(ID_A)P_1 + P_{pub})$, output $\perp$, else return m' as the plaintext.

Using the similar approach employed in the proof of Theorem 8 and the result of Theorem 5.4 in [17], we can reduce the security of SK-IBE2 to the k-BDHI assumption. We leave the details to the readers.

5 Comparison between SK-IBE and BF-IBE

From the reduction described in Section 3.2, we have proved that SK-IBE is a secure IBE scheme based on the k-BDHI problem. The complexity analysis of k-DHI, k-sDH and k-BDHI in [1, 2, 40] has built confidence on these assumptions.

The security of BF-IBE is based on the BDH problem [4]. As shown in Theorem 6, BDH and 1-BDHI are polynomial time equivalent. It is obvious that the k-BDHI problem (when $k > 1$) is easier that the 1-BDHI problem, and therefore, is easier than the BDH problem as well. This certainly shows the disadvantage of current reduction for SK-IBE as compared with one for BF-IBE [4, 18]. We leave it an open problem to find a tight reduction for SK-IBE based on a harder problem than k-BDHI.

However, the advantage of SK-IBE is that it has better performance than BF-IBE, particularly in encryption. We show a comparison of their performances in Table 1. If taking a closer look between SK-IBE and BF-IBE, SK-IBE is faster

	pairings		multiplications		exponentiations		hashes	
Scheme	Encrypt	Decrypt	Encrypt	Decrypt	Encrypt	Decrypt	Encrypt	Decrypt
SK-IBE	0	1	2^{*1}	1	1	0	4	3
BF-IBE	1	1	1	1	1	0	4^{*2}	3

$*1$ An extra multiplication required than BF-IBE is used to map an identifier to an element in $\mathbb{G}_1$.

$*2$ BF-IBE requires the *maptopoint* operation to map an identifier to an element in $\mathbb{G}_1$ (or $\mathbb{G}_2$) which is slower than the hash function used in SK-IBE which maps an identifier to an element in $\mathbb{Z}_q^*$.

Table 1. Performance comparison between SK-IBE and BF-IBE

than BF-IBE in two aspects. First, in the Encrypt algorithm of SK-IBE, no pairing computation is required because $\hat{e}(P_1, P_2)$ can be pre-computed. Second, in operation of mapping an identity to an element in $\mathbb{G}_1$ or $\mathbb{G}_2$, the *maptopoint* algorithm used by BF-IBE is not required. Instead of that, SK-IBE makes use of an ordinary hash-function.

6 Conclusion

In this paper, an identity-based encryption scheme, SK-IBE, is investigated. SK-IBE provides an attractive performance. We prove that SK-IBE is secure against adaptive chosen ciphertext attacks in the random oracle model based on the k-BDHI assumption.

Acknowledgements

We thank Keith Harrison, John Malone-Lee and Nigel Smart for helpful discussions and comments on this work, David Galindo for his valuable comments on the manuscript and for sharing with us the latest work on the security notions of IBE [19], Alex Dent for his detailed and valuable comments on the manuscript, and anonymous referees for useful feedback.

References

1. D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Proceedings of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, pp. 223–238, Springer-Verlag, 2004.
2. D. Boneh and X. Boyen. Short signatures without random oracles. In *Proceedings of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, pp. 56–73, Springer-Verlag, 2004.
3. D. Boneh and X. Boyen. Secure identity-based encryption without random oracles. In *Proceedings of Advances in Cryptology - Crypto 2004*, Springer-Verlag, 2004.
4. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Proceedings of Advances in Cryptology - Crypto 2001*, LNCS 2139, pp.213–229, Springer-Verlag, 2001.
5. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Advances in Cryptology – Asiacrypt 2001*, Springer-Verlag LNCS 2248, 514–532, 2001.
6. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the First Annual Conference on Computer and Communications Security*, ACM, 1993.
7. X. Boyen. Multipurpose identity-based signcryption: a swiss army knife for identity-based cryptography. In *Proceedings of Advances in Cryptology - CRYPTO 2003*, LNCS 2729, pp. 382–398, Springer-Verlag, 2003.
8. R. Canetti, S. Halevi and J. Katz. A forward-secure public-key encryption scheme. In *Proceedings of Advances in Cryptology - Eurocrypt 2003*, LNCS 2656, pp. 255-271, Springer-Verlag, 2003.
9. R. Canetti, S. Halevi and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Proceedings of Advances in Cryptology - Eurocrypt 2004*. Springer-Verlag, 2004. See also Cryptology ePrint Archive, Report 2003/182.
10. J. C. Cha and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Proceedings of Practice and Theory in Public Key Cryptography - PKC 2003*, LNCS 2567, pp. 18–30, Springer-Verlag, 2003. See also Cryptology ePrint Archive, Report 2002/018.
11. L. Chen and C. Kudla. Identity-based authenticated key agreement from pairings. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, pp. 219-233, IEEE, 2003. See also Cryptology ePrint Archive, Report 2002/184.
12. L. Chen and J. Malone-Lee. Improved identity-based signcryption. In *Proceedings of Public Key Cryptography - PKC 2005*, LNCS 3386, pages 362–379, Springer-Verlag, 2005. See also Cryptology ePrint Archive, Report 2004/114.
13. Z. Cheng and L. Chen. On security proof of McCullagh-Barreto's key agreement protocol and its variants. Cryptology ePrint Archive, Report 2005/201.
14. Z. Cheng, M. Nistazakis, R. Comley and L. Vasiu. On the indistinguishability-based security model of key agreement protocols-simple cases. In *Proceedings of ACNS 2004*. Full version available on Cryptology ePrint Archive, Report 2005/129.
15. C. Cocks. An identity-based encryption scheme based on quadratic residues. In *Proceedings of Cryptography and Coding*, LNCS 2260, pp. 360–363, Springer-Verlag, 2001.
16. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of Advances in Cryptology - CRYPTO '99*, LNCS 1666, pp. 535-554, Springer-Verlag, 1999.

17. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. *IEICE Trans. Fundamentals*, E83-9(1):24-32, 2000.
18. D. Galindo. Boneh-Franklin identity based encryption revisited. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*. Also available on Cryptology ePrint Archive, Report 2005/117.
19. D. Galindo and I. Hasuo. Security Notions for Identity Based Encryption. Manuscript, 2005.
20. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
21. F. Hess. Efficient identity based signature schemes based on pairings. In *Proceedings of Selected Areas in Cryptography – SAC 2002*, LNCS 2595, pp. 310–324, Springer-Verlag, 2002.
22. ISO/IEC 11770-3:1999. Information technology - Security techniques - Key management - Part 3: Mechanisms using asymmetric techniques.
23. ISO/IEC 14888-2:1998. Information technology - Security techniques - Digital signatures with appendix - Part 2: Identity-based mechanisms.
24. ISO/IEC 2nd FCD 18033-2:2004-12-06. Information technology - Security techniques - Encryption algorithms - Part 2: Asymmetric ciphers.
25. N. McCullagh and P. S. L. M. Barreto. Efficient and forward-secure identity-based signcryption. Available on Cryptology ePrint Archive, Report 2004/117.
26. N. McCullagh and P. S. L. M. Barreto. A new two-party identity-based authenticated key agreement. In *Proceedings of CT-RSA 2005*. See also Cryptology ePrint Archive, Report 2004/122.
27. S. Mitsunari, R. Sakai and M. Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–484, 2002.
28. R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054.
29. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. *The 2000 Symposium on Cryptography and Information Security*, Okinawa, Japan, January 2000.
30. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing over elliptic curve (in Japanese). *The 2001 Symposium on Cryptography and Information Security*, Oiso, Japan, January 2001.
31. M.Scott. Computing the Tate pairing. In *Proceedings of CT-RSA 2005*, LNCS 3376, pp. 293–304, Springer-Verlag, 2005.
32. M. Scott and P. S. L. M. Barreto. Compressed pairings. In *Proceedings of Advances in Cryptology - Crypto 2004*, LNCS 3152, 2004, Springer-Verlag, 2004. See also Cryptology ePrint Archive, Report 2004/032.
33. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of Advances in Cryptology - Crypto '84*, LNCS 196, pp.47–53, Springer-Verlag, 1985.
34. J. Silverman. *The arithmetic of elliptic curve*. Springer-Verlag, 1986.
35. N. P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38(13):630–632, 2002. See also Cryptology ePrint Archive, Report 2001/111.
36. N. Smart and F. Vercauteren. On computable isomorphisms in efficient pairing based systems. Cryptology ePrint Archive, Report 2005/116.
37. V.D. Tô, R. Safavi-Naini and F. Zhang. New traitor tracing schemes using bilinear map. In *Proceedings of 2003 DRM Workshop*, 2003.

38. B. R. Waters. Efficient identity-based encryption without random oracles. In *Proceedings of Advances in Cryptology - Eurocrypt 2005*, Springer-Verlag, 2005. See also Cryptology ePrint Archive, Report 2004/180.
39. V. Wei. Tight Reductions among Strong Diffie-Hellman Assumptions. Cryptology ePrint Archive, Report 2005/057.
40. F. Zhang, R. Safavi-Naini and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Proceedings of International Workshop on Practice and Theory in Public Key Cryptography - PKC 2004*, 2004.

Appendix

A Proof of Theorem 2

This proof is similar to the proof of Theorem 3.5 [27].

Proof: If there is a polynomial time algorithm $\mathcal{A}$ to solve the (k-1)-DHI problem, we construct a polynomial time algorithm $\mathcal{B}$ to solve the k-CAA1 problem. Given an instance of k-CAA1 problem $(Q, yQ, h_0, (h_1, \frac{1}{h_1+y}Q), \ldots, (h_k, \frac{1}{h_k+y}Q))$, $\mathcal{B}$ works as follow to compute $\frac{1}{y+h_0}Q$.

1. Set $x = y + h_0$ which $\mathcal{B}$ does not know, and $P = \frac{1}{(y+h_1)\cdots(y+h_k)}Q$.
2. For $j = 0, \ldots, (k-1)$, $\mathcal{B}$ computes $x^j P = \frac{(y+h_0)^j}{(y+h_1)\cdots(y+h_k)}Q = \sum_{i=1}^{k} \frac{c_{ij}}{y+h_i}Q$ where $c_{ij} \in \mathbb{Z}_q$ are computable from h_i.
3. Pass $\mathcal{A}$ the (k-1)-DHI challenge, $(P, xP, \cdots, x^{k-1}P)$, and get $T = \frac{1}{x}P$.
4. Set $f(z) = \prod_{i=1}^{k}(z + h_i - h_0) = \sum_{i=0}^{k} d_i z^i$ where d_i are computable from h_i and $d_0 \neq 0$ because h_i are different.
5. Note that $Q = f(x)P = \sum_{i=0}^{k} d_i x^i P$, so compute $\frac{1}{y+h_0}Q = \frac{1}{x}Q = \frac{f(x)}{x}P = \sum_{i=0}^{k} d_i x^{i-1}P = d_0 \frac{1}{x}P + \sum_{i=1}^{k} d_i x^{i-1}P = d_0 T + \sum_{i=1}^{k} d_i x^{i-1}P$.

If there is a polynomial time algorithm $\mathcal{A}$ to solve the (k-1)-CAA1 problem, we construct a polynomial time algorithm $\mathcal{B}$ to solve the k-DHI problem. Given an instance of k-DHI problem $(P, xP, x^2P, \ldots, x^kP)$, $\mathcal{B}$ works as follow to compute $\frac{1}{x}P$.

1. Randomly choose different $h_0, \ldots, h_{k-1} \in \mathbb{Z}_q^*$ and set $y = x - h_0$ which $\mathcal{B}$ does not know.
2. Let $f(z)$ be the polynomial $f(z) = \prod_{i=1}^{k-1}(z + h_i - h_0) = \sum_{i=0}^{k-1} c_i z^i$. The constant term c_0 is non-zero because h_i are different.
3. Compute $Q = \sum_{i=0}^{k-1} c_i x^i P = f(x)P$ and $yQ = \sum_{i=0}^{k-1} c_i x^{i+1}P - h_0 Q = xf(x)P - h_0 Q$.
4. Compute $f_i(z) = f(z)/(z+h_i-h_0) = \sum_{j=0}^{k-2} d_j z^j$ and $\frac{1}{y+h_i}Q = \frac{1}{x+h_i-h_0}f(x)P = f_i(x)P = \sum_{j=0}^{k-2} d_j x^j P$ for $1 \leq i \leq k-1$.
5. Pass the following instance of the (k-1)-CAA problem to $\mathcal{A}$

$$(Q, yQ, h_0, (h_1, \frac{1}{y+h_1}Q), \ldots, (h_{k-1}, \frac{1}{y+h_{k-1}}Q))$$

   to get the response $T = \frac{1}{y+h_0}Q = \frac{1}{x}Q$.

6. Note that $T = \frac{f(x)}{x}P = \sum_{i=0}^{k-1} c_i x^{i-1} P = c_0 \frac{1}{x} P + \sum_{i=1}^{k-1} c_i x^{i-1} P$. So compute $\frac{1}{x}P = c_0^{-1}(T - \sum_{i=1}^{k-1} c_i x^{i-1} P)$. $\square$

B Proof of Theorem 4

This proof is similar to the proof of Theorem 2 above.

Proof: If there exists an algorithm $\mathcal{A}$ to solve a random instance of the (k-1)-sCAA1 problem in polynomial time, we can construct a polynomial time algorithm $\mathcal{B}$ to solve the k-sDH problem. Given a random instance of the k-sDH problem, $(P, xP, x^2P, \ldots, x^kP)$, $\mathcal{B}$ takes the following steps to compute $(h, \frac{1}{x+h}P)$.

1. Randomly choose different $h_1, \ldots, h_{k-1} \in \mathbb{Z}_q^*$ and let $f(z)$ be the polynomial $f(z) = \prod_{i=1}^{k-1}(z + h_i)$. Reformulate f to get $f(z) = \sum_{i=0}^{k-1} c_i z^i$. The constant term c_0 is non-zero and c_i are computable from h_i.
2. Compute $Q = \sum_{i=0}^{k-1} c_i x^i P = f(x)P$ and $xQ = \sum_{i=0}^{k-1} c_i x^{i+1} P = xf(x)P$.
3. Check that $Q \in \mathbb{G}_1^*$. If $Q = 1_{\mathbb{G}_1}$, then there must be such $h_i = -x$ which can be easily identified, and so, $\mathcal{B}$ solves the problem directly. Otherwise, $\mathcal{B}$ continues.
4. Compute $f_i(z) = f(z)/(z + h_i) = \sum_{j=0}^{k-2} d_j z^j$ and $\frac{1}{x+h_i}Q = f_i(x)P = \sum_{j=0}^{k-2} d_j x^j P$ for $1 \le i \le k-1$.
5. Pass the following instance of the (k-1)-sCAA1 problem to $\mathcal{A}$.
$$(Q, xQ, (h_1, \frac{1}{x+h_1}Q), \ldots, (h_{k-1}, \frac{1}{x+h_{k-1}}Q))$$
to get $(h_0, \frac{1}{h_0+x}Q)$.
6. Note that $\frac{1}{h_0+x}f(x) = \frac{w_0}{h_0+x} + \sum_{i=1}^{k-1} w_i x^{i-1}$ where w_i are computable from h_i, and $w_0 \neq 0$ because h_i are different. Compute $\frac{1}{x+h_0}P = w_0^{-1}(\frac{1}{x+h_0}Q - \sum_{i=1}^{k-1} w_i x^{i-1} P)$. Output $(h_0, \frac{1}{x+h_0}P)$.

If there is a polynomial time algorithm $\mathcal{A}$ to solve the (k-1)-sDH problem, we construct a polynomial time algorithm $\mathcal{B}$ to solve the k-sCAA1 problem. Given an instance of k-sCAA1 problem $(Q, yQ, (h_1, \frac{1}{h_1+y}Q), \ldots, (h_k, \frac{1}{h_k+y}Q))$, $\mathcal{B}$ works as follow to compute $(h, \frac{1}{y+h}Q)$.

1. For $j = 0, \ldots, (k-1)$, $\mathcal{B}$ computes $y^j P = \frac{y^j}{(y+h_1)\cdots(y+h_k)}Q = \sum_{i=1}^{k} \frac{c_{ij}}{y+h_i}Q$ where $c_{ij} \in \mathbb{Z}_q$ are computable from h_i.
2. Pass $\mathcal{A}$ the (k-1)-sDH challenge, $(P, yP, \cdots, y^{k-1}P)$, and get $(h_0, \frac{1}{y+h_0}P)$.
3. Note that $\frac{1}{y+h_0}P = \frac{1}{(y+h_0)(y+h_1)\cdots(y+h_k)}Q = \sum_{i=0}^{k} \frac{c_i}{y+h_i}Q$, for $c_i \in \mathbb{Z}_q$ are computable from h_i and $c_0 \neq 0$ because h_i are different. Compute $\frac{1}{y+h_0}Q = c_0^{-1}(\frac{1}{y+h_0}P - \sum_{i=1}^{k} \frac{c_i}{y+h_i}Q)$. Output $(h_0, \frac{1}{y+h_0}Q)$. $\square$

C Proof of Theorem 6

Proof: If there is a polynomial time algorithm $\mathcal{A}$ to solve the BDH problem, we construct a polynomial time algorithm $\mathcal{B}$ to solve the 1-BDHI problem. Given an instance of 1-BDHI problem (Q_1, Q_2, yQ_2), $\mathcal{B}$ works as follow to compute $\hat{e}(Q_1, Q_2)^{1/y}$.

1. Set $x = 1/y$, which $\mathcal{B}$ does not know.
2. Set $P_1 = Q_1$, $P_2 = yQ_2$ and $xP_2 = Q_2$.
3. Pass $\mathcal{A}$ the BDH challenge, $(P_1, P_2, xP_2, xP_2, P_2)$, and get $T = \hat{e}(P_1, P_2)^{x^2}$ $= \hat{e}(Q_1, yQ_2)^{(1/y)^2} = \hat{e}(Q_1, Q_2)^{1/y}$.

If there is a polynomial time algorithm $\mathcal{A}$ to solve the 1-BDHI problem, we construct a polynomial time algorithm $\mathcal{B}$ to solve the BDH problem. Given an instance of BDH problem $(P_1, P_2, aP_2, bP_2, cP_2)$, $\mathcal{B}$ works as follow to compute $\hat{e}(P_1, P_2)^{abc}$.

1. (a) Set $d = 1/(a + b + c)$, which $\mathcal{B}$ does not know.
   (b) Set $Q_1 = (a + b + c)P_1 = \psi((a + b + c)P_2)$, $Q_2 = (a + b + c)P_2$ and $dQ_2 = P_2$.
   (c) Pass $\mathcal{A}$ the 1-BDHI challenge, (Q_1, Q_2, dQ_2), and get $T_1 = \hat{e}(Q_1, Q_2)^{1/d}$ $= \hat{e}(P_1, P_2)^{(a+b+c)^3}$.
2. Follow Item 1 (a) - (c) to get $T_2 = \hat{e}(P_1, P_2)^{a^3}$, $T_3 = \hat{e}(P_1, P_2)^{b^3}$, $T_4 = \hat{e}(P_1, P_2)^{c^3}$, $T_5 = \hat{e}(P_1, P_2)^{(a+b)^3}$, $T_6 = \hat{e}(P_1, P_2)^{(a+c)^3}$, $T_7 = \hat{e}(P_1, P_2)^{(b+c)^3}$.
3. Compute $\hat{e}(P_1, P_2)^{abc} = (\frac{T_1 \cdot T_2 \cdot T_3 \cdot T_4}{T_5 \cdot T_6 \cdot T_7})^{1/6}$. □

D Proof of Theorem 7

Proof: If there is a polynomial time algorithm $\mathcal{A}$ to solve the (k-1)-BDHI problem, we construct a polynomial time algorithm $\mathcal{B}$ to solve the k-BCAA1 problem. Given an instance of k-BCAA1 problem $(Q_1, Q_2, yQ_2, h_0, (h_1, \frac{1}{h_1+y}Q_2), \ldots, (h_k, \frac{1}{h_k+y}Q_2))$, $\mathcal{B}$ works as follow to compute $\hat{e}(Q_1, Q_2)^{1/(y+h_0)}$.

1. Set $x = y + h_0$ which $\mathcal{B}$ does not know, and $P_2 = \frac{1}{(y+h_1)\cdots(y+h_k)}Q_2$.
2. For $j = 0, \ldots, (k-1)$, $\mathcal{B}$ computes $x^j P_2 = \frac{(y+h_0)^j}{(y+h_1)\cdots(y+h_k)}Q_2 = \sum_{i=1}^{k} \frac{c_{ij}}{y+h_i}Q_2$ where $c_{ij} \in \mathbb{Z}_q$ are computable from h_i.
3. Set $P_1 = \psi(P_2)$.
4. Pass $\mathcal{A}$ the (k-1)-BDHI challenge, $(P_1, P_2, xP_2, \cdots, x^{k-1}P_2)$, and get $T = \hat{e}(P_1, P_2)^{1/x}$.
5. Set $f(z) = \prod_{i=1}^{k}(z + h_i - h_0) = \sum_{i=0}^{k} d_i z^i$ where d_i is computable from h_i and $d_0 \neq 0$ because h_i are different.
6. Note that $Q_2 = f(x)P_2 = \sum_{i=0}^{k} d_i x^i P_2$ and $\frac{1}{x}Q_2 = \frac{f(x)}{x}P_2 = \sum_{i=0}^{k} d_i x^{i-1} P_2$.
7. Compute $\hat{e}(Q_1, Q_2)^{1/(y+h_0)} = \hat{e}(\frac{1}{x}\psi(Q_2), Q_2) = \hat{e}(\sum_{i=0}^{k} d_i x^{i-1}\psi(P_2), Q_2) =$ $T^{d_0^2} \cdot \hat{e}(d_0 P_1, \sum_{i=1}^{k} d_i x^{i-1} P_2) \cdot \hat{e}(\sum_{i=1}^{k} d_i \psi(x^{i-1} P_2), Q_2)$.

If there is a polynomial time algorithm $\mathcal{A}$ to solve the (k-1)-BCAA1 problem, we construct a polynomial time algorithm $\mathcal{B}$ to solve the k-BDHI problem. Given an instance of k-BDHI problem $(P_1, P_2, xP_2, x^2P_2, \ldots, x^kP_2)$, $\mathcal{B}$ works as follow to compute $\hat{e}(P_1, P_2)^{1/x}$.

1. Randomly choose different $h_0, \ldots, h_{k-1} \in \mathbb{Z}_q^*$ and set $y = x - h_0$ which $\mathcal{B}$ does not know.
2. Let $f(z)$ be the polynomial $f(z) = \prod_{i=1}^{k-1}(z + h_i - h_0) = \sum_{i=0}^{k-1} c_i z^i$. The constant term c_0 is non-zero because h_i are different and c_i are computable from h_i.
3. Compute $Q_2 = \sum_{i=0}^{k-1} c_i x^i P_2 = f(x)P_2$ and $yQ_2 = \sum_{i=0}^{k-1} c_i x^{i+1} P_2 - h_0 Q_2 = xf(x)P_2 - h_0 Q_2$.
4. Compute $f_i(z) = f(z)/(z+h_i-h_0) = \sum_{j=0}^{k-2} d_j z^j$ and $\frac{1}{y+h_i}Q_2 = \frac{1}{x+h_i-h_0} f(x)P_2$ $= f_i(x)P_2 = \sum_{j=0}^{k-2} d_j x^j P_2$ for $1 \leq i \leq k-1$.
5. Set $Q_1 = \psi(Q_2)$.
6. Pass the following instance of the (k-1)-BCAA1 problem to $\mathcal{A}$

$$(Q_1, Q_2, yQ_2, h_0, (h_1, \frac{1}{y+h_1}Q_2), \ldots, (h_{k-1}, \frac{1}{y+h_{k-1}}Q_2))$$

   to get $T = \hat{e}(Q_1, Q_2)^{1/(y+h_0)} = \hat{e}(Q_1, Q_2)^{1/x} = \hat{e}(P_1, P_2)^{f^2(x)/x}$.
7. Note that $\frac{1}{x}Q_2 = \frac{f(x)}{x}P_2 = \sum_{i=0}^{k-1} c_i x^{i-1} P_2 = c_0 \frac{1}{x} P_2 + \sum_{i=1}^{k-1} c_i x^{i-1} P_2$. Set $T' = \sum_{i=1}^{k-1} c_i x^{i-1} P_2 = \frac{f(x)-c_0}{x} P_2$. Then, $\hat{e}(\frac{1}{x}Q_1, Q_2) = \hat{e}(P_1, P_2)^{c_0^2/x} \cdot \hat{e}(\psi(T'), Q_2 + c_0 P_2)$. Compute $\hat{e}(P_1, P_2)^{1/x} = (T/\hat{e}(\psi(T'), Q_2 + c_0 P_2))^{1/c_0^2}$.

□