Int. J. Inf. Secur. (2016) 15:35–50
DOI 10.1007/s10207-015-0287-8

SPECIAL ISSUE PAPER

Practical chosen-ciphertext secure Hierarchical Identity-Based Broadcast Encryption

Weiran Liu[1,2] · **Jianwei Liu**[1,5] · **Qianhong Wu**[1,3] · **Bo Qin**[4] · **Yan Li**[5]

Published online: 24 April 2015

Abstract We focus on practical Hierarchical Identity-Based Broadcast Encryption (HIBBE) with semantic security against adaptively chosen-ciphertext attacks (CCA2) in the standard model. We achieve this goal in two steps. First, we propose a new HIBBE scheme that is secure against chosen-plaintext attacks (CPA). Compared with the existing HIBBE scheme that is built from composite-order bilinear groups, our construction is based on prime-order bilinear groups. The much better efficiency of group operations in prime-order bilinear groups makes our proposed HIBBE scheme more practical. Then, we convert it into a CCA2-secure scheme at the cost of a one-time signature. Instead of extending one user hierarchy in the Canetti–Halevi–Katz approach from CPA-secure $(l+1)$-Hierarchical Identity-Based Encryption [$(l+1)$-HIBE] to CCA2-secure l-HIBE, our construction merely adds one on-the-fly dummy user in the basic scheme. We formally prove the security of these two schemes in the standard model. Comprehensive theoretical analyses and experimental results demonstrate that the proposed HIBBE schemes achieve desirable performance.

✉ Qianhong Wu
qhwu@xidian.edu.cn

Bo Qin
bo.qin@ruc.edu.cn

[1] School of Electronic and Information Engineering, Beihang University, XueYuan Road No.37, Haidian District, Beijing, China

[2] The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China

[3] The State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

[4] The Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China) Ministry of Education, School of Information, Renmin University of China, ZhongGuanCun Street No.59, Haidian District, Beijing, China

[5] Aerospace Hengxing Science and Technology co. LTD, Beijing, China

Keywords Identity-based encryption · Broadcast encryption · Hierarchical Identity-Based Broadcast Encryption · Access control · Provable security

1 Introduction

Identity-Based Encryption (IBE) is a public key system that allows users to encrypt message using the receiver's identity as the public key. In IBE, users' identities can be arbitrary strings, e.g., social security numbers, IP or email addresses. A private key generator (PKG) is employed in the system that uses a master secret key to distribute secret keys for users associated with their own identities. Only the user whose identity matches the specified identity in the ciphertext can decrypt.

Hierarchical Identity-Based Encryption (HIBE) generalizes IBE by organizing users in a tree-like structure. A secret key delegation mechanism is supported for higher-level users to issue secret keys to their descendant ones. In encryption, one can specify an identity vector, instead of a single identity, to the ciphertext. The users whose identities appearing in the specified identity vector can have capability to decrypt.

Hierarchical Identity-Based Broadcast Encryption (HIBBE) further extends HIBE by allowing ones to broadcast messages to multiple receivers in a hierarchical social organizations. Similar to HIBE, HIBBE organizes users in a hierarchical structure and users can delegate their decryption capability to their subordinates. In encryption, an identity vector set containing intended receivers is associated with

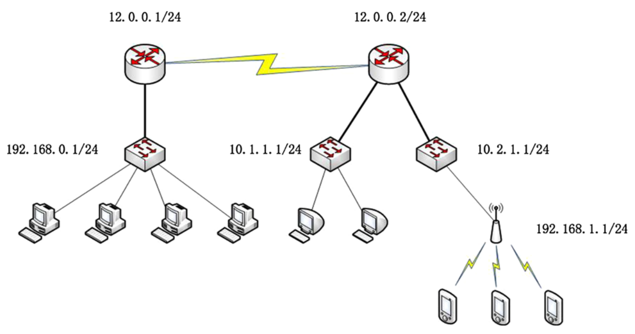

Fig. 1 IP-based network

the ciphertext. Only the users with identities in the identity vector set can correctly decrypt the ciphertext.

HIBBE is shown to have unique applications in practice since it reflects hierarchical social organizations in the real world. To demonstrate the power of HIBBE, we illustrate some application scenarios to which HIBBE is appealing for secure communications.

- **IP-based multicast networks** One application may be to secure IP-based network, in which all nodes are organized in hierarchy that are labeled by their IP addresses and subnet masks [36]. Figure 1 shows a typical IP-based network containing routers, switchers, access points, terminals, and wireless devices, all of which are naturally organized in a tree-like structure by distributing proper IP addresses and subnet masks. IP multicast is the method of broadcasting IP packets to a group of receivers in a single transmission. HIBBE is an efficient cryptographic primitive that can be applied to securely transfer sensitive data in IP-based multicast networks by directly assigning nodes' IP addresses as their identities.
- **Clustering wireless sensor networks** Wireless sensor network (WSN) is a distributed network that monitors physical or environmental conditions with the help of autonomous sensors [32]. Clustering is an efficient routing design methodologies to manage sensors in WSN [21]. As shown in Fig. 2, nodes in a cluster-based WSN naturally form a two-layer hierarchy, where cluster heads are in the first depth and others are in the second. Applying HIBBE systems in such a network is a practical solution to securely sending commands to any subset of nodes for accurately manipulating wireless sensors.

Motivated by the above scenarios, it is desirable to construct a secure HIBBE system. In 2014, Liu et al. introduced and defined the security model of HIBBE systems and proposed a HIBBE construction with constant-size ciphertext [27]. Theoretical analyses show the feasibility and efficiency of their HIBBE in terms of communication so that it can be a candidate scheme to be applied in various networks.

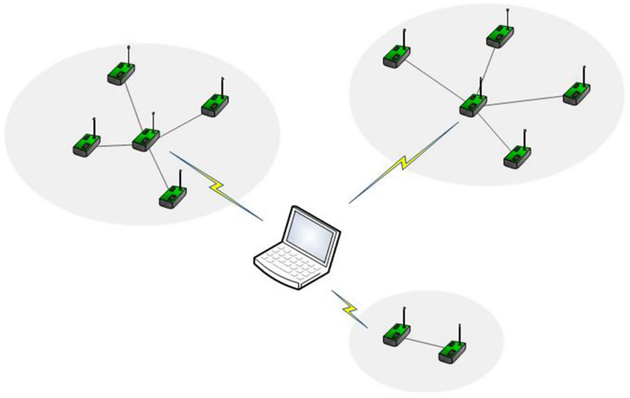

Fig. 2 Clustering wireless sensor network

There are remaining problems for HIBBE to be used in practical applications. The first problem is the efficiency of HIBBE. The construction proposed by Liu et al. is built from composite-order bilinear groups. Aside from its security, leveraging composite-order bilinear groups to construct cryptosystems inevitably incurs heavy computation overhead. Indeed, theoretical analysis and experiment results have shown that group operations, including addition, multiplication, exponentiation, and pairing are prohibitively slow on composite-order bilinear groups [17,29]. Therefore, it is preferable to construct a more practical HIBBE scheme built on prime-order bilinear groups. When applying HIBBE in practice, one can consider the tradeoff between efficiency and security, thus leveraging the suitable construction to meet the actual needs of the system. One solution may be to outsource the expensive exponentiations to a semi-trusted third party suggested by Wang et al. [33]. However, this requires extra interactions between the clients and the computing server with additional communication overheads.

Another problem is the chosen-ciphertext security (CCA2) requirement for HIBBE. The existing HIBBE scheme is only secure against chosen plaintext attacks (CPA), which ensures security against a passive adversary. In the real-life scenarios, however, a powerful adversary may control the network traffic, which allows it to manipulate all ciphertexts of its choice. In this way, the powerful adversary may reveal useful information from the sensitive data even though the underlying cryptosystem is CPA-secure. Semantic security against CCA2 attacks is a strong and useful notion of security for cryptosystems against such active attacks. Therefore, it is preferable to construct a CCA2-secure HIBBE and deploy it in practice to ensure security.

1.1 Our contributions

We focus on constructing practical HIBBE schemes with CCA2 security. We achieve this goal in two steps.

First, we propose an HIBBE scheme with semi-static chosen-identity-vector-set and chosen-plaintext security (IND-ssCIVS-CPA) in the standard model. Unlike the existing HIBBE scheme [27] that is built from composite-order bilinear groups, our construction is built from prime-order bilinear groups and achieve much better efficiency. Also, compared with existing broadcast encryption schemes based on prime-order bilinear groups that can only achieve the widely used full-static security, we show that our construction has a stronger security result named semi-static security [20], in which the adversary is allowed to partly decide which sets of the receivers it wishes to challenge.

Second, we convert the basic scheme into an IND-ssCIVS-CCA2 secure HIBBE scheme by introducing the Canetti–Halevi–Katz technique into our construction. To achieve better efficiency, we do the conversion by only adding one on-the-fly dummy user in the system, instead of extending one hierarchy of users as the Canetti-Halevi-Katz technique. The resulting scheme is at merely marginal cost, i.e., only one extra dummy user is required to achieve CCA2 security.

We conduct thorough theoretical analysis and experiments on our practical HIBBE scheme. The analysis shows that our scheme is much more efficient than the existing HIBBE scheme [27], that is, about 50 times more efficient than the scheme in [27]. The experimental results also indicate that the proposed HIBBE scheme provides better user experience and show that it is applicable to the practical applications.

1.2 Related work

1.2.1 Identity-Based Encryption

Shamir [31] first introduced the concept of IBE in 1984. However, it took a long time for the researchers to construct a practical and secure IBE scheme. In 2001, Boneh and Franklin [5,6] formally defined the security notions of IBE and proposed the first practical IBE system by using bilinear groups. The security of the proposed scheme is based on the random oracle model. Since then, many other constructions of IBE systems were proposed to achieve better security and/or efficiency. Canetti et al. [11] defined a weaker security model for IBE, named selective security model, and provided an IBE scheme that can be proven secure in the selective security model without random oracles. Boneh and Boyen [1] improved their results and described an efficient scheme that is secure in the selective security model. The first fully/adaptively secure IBE scheme in the standard model is presented by Boneh and Boyen [2]. However, their construction is inefficient for practical deployment. The efficient and fully secure IBE systems were proposed by Waters [34].

These IBE schemes are secure against CPA. Canetti et al. [12] showed that any CPA-secure IBE can be directly transformed to be CCA2-secure Public Key Encryption systems. The basic idea is to leverage a one-time signature scheme and treat the verification key as the “identity” in encryption. Canetti et al. also pointed out that this technique can be used to transform CPA-secure 2-level HIBE to be CCA2-secure IBE. Boneh et al. [9] improved the Canetti–Halevi–Katz technique by using a MAC to replace the one-time signature. Boyen et al. [10] further introduced a new transformation technique that can be applied to some HIBE systems without involving extra cryptographic primitives. In 2006, Gentry [18] presented an efficient IBE system and converted it into a CCA2-secure one with a similar technique from the Cramer-Shoup encryption scheme [13]. Very recently, Liu et al. [28] refined this technique with chameleon hash to convert a CPA-secure attribute-based encryption (ABE) scheme into a CCA2-secure ABE scheme. Deng et al. [15,16] shows how to trace compromised secret keys in IBE and ABE systems.

1.2.2 Hierarchical IBE (HIBE)

Horwitz and Lynn [22] introduced the concept of HIBE. Gentry and Silverberg [19] proposed a HIBE scheme in the random oracle model. Boneh and Boyen [1] proposed a HIBE construction that achieves security in the selective model without random oracles. A more efficient selective secure HIBE scheme with constant-size ciphertext was introduced by Boneh et al. [4]. The first fully/adaptively secure HIBE was constructed by Gentry and Halevi that supports polynomial depth of hierarchy, with its security under a new complexity assumption. In 2009, Waters [35] introduced a novel approach, called Dual System, for achieving fully secure IBE and HIBE systems in the composite-order bilinear groups. Dual system was subsequently used to prove full security of HIBE [23] and other related systems [24,25,30]. However, how to construct fully secure HIBE systems in prime-order bilinear groups under simple assumptions remains as a challenging problem [26]. These schemes are CPA-secure. It is possible to convert HIBE schemes to be CCA2-secure by leveraging the previous reviewed conversions [9,10,12].

1.2.3 Spatial encryption

In 2008, Boneh and Hamburg [8] provided a framework for constructing Identity-Based and Broadcast encryption systems, called spatial encryption. They indicated that many systems can be derived from it, including HIBE, inclusive IBE, co-inclusive IBE, etc. HIBBE is also a kind of encryption systems belonging to spatial encryption. However, the HIBBE system derived from their spatial encryption has only selective and CPA.

1.3 Paper organization

The reminder of this paper is organized as follows. In Sect. 2, we review prime-order bilinear groups, and the Weak Bilinear Diffie–Hellman Inversion assumption that will be used in the security proof of our construction. Section 3 formalizes HIBBE and its security definitions. We propose our practical IND-ssCIVS-CPA secure HIBBE in Sect. 4. The IND-ssCIVS-CCA2 secure HIBBE scheme is then presented in Sect. 5. We conduct theoretical and experimental performance analyses in Sect. 6. Finally, we conclude the paper in Sect. 7.

2 Preliminaries

In this section, we briefly review prime-order bilinear groups and the Weak Bilinear Diffie–Hellman Inversion assumption underlying our constructions.

2.1 Prime-order bilinear groups

Prime-order bilinear groups are defined by using a group generator $\mathcal{G}$, an algorithm which takes a security parameter λ as input and outputs a description of a bilinear group, $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, where p is a large prime, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order p, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying the following properties:

1. *Bilinearity* For all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have that $e(g^a, h^b) = e(g, h)^{ab}$;
2. *Non-degeneracy* There exists at least an element $g \in \mathbb{G}$ such that $e(g, g)$ has order p in $\mathbb{G}_T$;
3. *Computability* There exists an efficient algorithm (in polynomial time with respect to λ) to compute the bilinear pairing $e(u, v)$ for all $u, v \in \mathbb{G}$.

2.2 Weak Bilinear Diffie–Hellman Inversion assumption

The security of our practical HIBBE schemes rely on the decision version of the Weak Bilinear Diffie–Hellman Inversion (wBDHI*) assumption [4].

Let $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, $g, h \in \mathbb{G}$ be generators of $\mathbb{G}$ and $\alpha \in \mathbb{Z}_p$. The N-wBDHI* problem in $\mathbb{G}$ is, given the tuple

$$\left(g, h, g^\alpha, g^{(\alpha^2)}, \ldots, g^{(\alpha^N)}\right)$$

as inputs, to compute $e(g, h)^{(\alpha^{N+1})}$. For simple notations, we denote $y_i = g^{(\alpha^i)} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving N-wBDHI* in $\mathbb{G}$ if

Table 1 Notations

Notation	Description
λ	Security parameter
PP	Public parameter
MSK	Master secret key
Hdr	Header
K	Session key
ID	Identity
ID	Identity vector
$\mathrm{I}_{\mathbf{ID}}$	Identity vector position
$SK_{\mathbf{ID}}$	Secret key for **ID**
$\Vert\mathbf{ID}\Vert$	Depth of **ID**
$S_{\mathbf{ID}}$	Identity set for **ID**
V	Identity vector set
$\mathbb{I}_{\mathbf{V}}$	Identity vector set position
$\Vert\mathbf{V}\Vert$	Depth of **V**
$S_{\mathbf{V}}$	Identity set for **V**

$$\left|\Pr\left[\mathcal{A}(g, h, y_1, \ldots, y_N) = e(g, h)^{(\alpha^{N+1})}\right]\right| \geq \epsilon$$

where the probability is over the random choice of generators $g, h \in \mathbb{G}$, the random choice of the exponent $\alpha \in \mathbb{Z}_p$, and the random bits used by $\mathcal{A}$.

The Decisional N-wBDHI* problem in $\mathbb{G}$ can be defined in the usual manner. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving decision N-wBDHI* in $\mathbb{G}$ if

$$\left|\begin{array}{l}\Pr\left[\mathcal{B}\left(g, h, y_1, \ldots, y_N, e(g, h)^{(\alpha^{N+1})}\right) = 0\right] - \\ \Pr\left[\mathcal{B}(g, h, y_1, \ldots, y_N, T) = 0\right]\end{array}\right| \geq \epsilon$$

where the probability is over the random choice of generators $g, h \in \mathbb{G}$, the random choice of the exponent $\alpha \in \mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_T$, and the random bits used by $\mathcal{B}$.

Definition 1 The (t, ϵ, N)-Decision wBDHI* assumption in $\mathbb{G}$ states that there exists no t-time algorithm that has advantage at least ϵ in solving the Decision N-wBDHI* problem in $\mathbb{G}$.

3 System model and security definitions

3.1 Notations

We follow the notations used in [27] to simplify the descriptions of HIBBE systems. Table 1 summaries these notions that will be later used in this paper.

We use $[a, b]$ to denote a set $\{a, a+1, \ldots, b\}$ containing consecutive integers. We write $|S|$ to denote the cardinality of the set S. For an identity vector $\mathbf{ID} = (\mathrm{ID}_1, \ldots, \mathrm{ID}_d)$, we

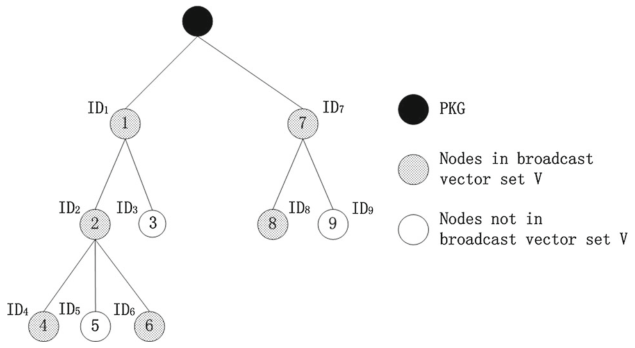
PKG
Nodes in broadcast vector set V
Nodes not in broadcast vector set V
ID1
ID2
ID3
ID4
ID5
ID6
ID7
ID8
ID9
1
2
3
4
5
6
7
8
9

Fig. 3 A typical example of the notations

define $\|\mathbf{ID}\|$ as the depth of **ID**, and $S_{\mathbf{ID}}$ as the identity set associating with **ID**. For an identity vector set **V**, we define the maximal depth of **V** as $\|\mathbf{V}\| = \max\{\|\mathbf{ID}\| : \mathbf{ID} \in \mathbf{V}\}$. The identity set of **V** can be defined similarly. We define the prefix of an identity vector $\mathbf{ID} = (\mathrm{ID}_1, \ldots, \mathrm{ID}_d)$ as an identity vector set represented by

$$Pref(\mathbf{ID}) = \{(\mathrm{ID}_1, \ldots, \mathrm{ID}_{d'}) : d' \in [1, d]\}$$

Clearly, we have that $\|Pref(\mathbf{ID})\| = d = \|\mathbf{ID}\|$. We can similarly define the prefix of a vector set **V** as

$$Pref(\mathbf{V}) = \bigcup_{\mathbf{ID} \in \mathbf{V}} Pref(\mathbf{ID})$$

In practice, a user may have more than one identity or parent nodes. In this case, we will treat them as different users.

We show an example of our notations using a typical HIBBE system illustrated in Fig. 3. For the user with identity vector $\mathbf{ID} = (\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_4)$, the depth of **ID** is $\|\mathbf{ID}\| = 3$, the identity set of **ID** is $S_{\mathbf{ID}} = \{\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_4\}$, and the identity position of **ID** is $\mathbb{I}_{\mathbf{ID}} = \{1, 2, 4\}$. The prefix of **ID** is denoted by

$$\begin{aligned} \mathrm{Pref}(\mathbf{ID}) &= \{(\mathrm{ID}_1, \ldots, \mathrm{ID}_{d'}) : d' \in [1, \|\mathbf{ID}\|]\} \\ &= \{(\mathrm{ID}_1), (\mathrm{ID}_1, \mathrm{ID}_2), (\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_4)\} \end{aligned}$$

For the broadcast identity vector set

$$\mathbf{V} = \{(\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_4), (\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_6), (\mathrm{ID}_7, \mathrm{ID}_8)\}$$

we similarly have that $\|\mathbf{V}\| = \max(3, 3, 2) = 3$, $S_{\mathbf{V}} = \{\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_4, \mathrm{ID}_6, \mathrm{ID}_7, \mathrm{ID}_8\}$, and the identity vector set position $\mathbb{I}_{\mathbf{V}} = \{1, 2, 4, 6, 7, 8\}$. The prefix of the broadcast identity vector set is

$$\mathrm{Pref}(\mathbf{V}) = \left\{ \begin{array}{l} (\mathrm{ID}_1), (\mathrm{ID}_1, \mathrm{ID}_2), (\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_4), \\ (\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{ID}_6), (\mathrm{ID}_7), (\mathrm{ID}_7, \mathrm{ID}_8) \end{array} \right\}$$

3.2 Hierarchical Identity-Based Broadcast Encryption

We are now ready to define HIBBE. An HIBBE scheme involves a PKG as an authority. The PKG has a master secret key *MSK* to be used for granting users capability of decrypting messages by generating secret keys $\mathrm{SK}_{\mathbf{ID}}$ associated with their identity vectors **ID**. Users in the higher level can use their secret key to delegate the decryption capability to their subordinates. The broadcaster encrypts messages under the public parameter *PP* published by the PKG and transmits these to the groups of the users represented by an identity vector set **V** via the broadcast channel. Only users whose identity vector **ID** in the prefix of **V** can decrypt. Compared with the definitions shown in [27], here we follow the key encapsulation mechanism methodology to meet the actual requirement of broadcast encryption, and view the broadcast encryption as a combination of a session key encapsulation mechanism with a symmetric encryption.

Formally, an HIBBE system with the security parameter λ, the maximal depth D of the hierarchy, and the maximal size n of the supported users is a tuple of algorithms $\Pi =$ (**Setup**, **KeyGen**, **Delegate**, **Encrypt**, **Decrypt**) defined as follows.

- **Setup**(D, n, λ) The **Setup** algorithm takes as inputs the maximal depth D of the hierarchy, the maximal size n of the supported users, and the security parameter λ. It outputs a master secret key *MSK* and a public parameter *PP*. The master secret key *MSK* is kept secret by the PKG, while the public parameter *PP* is made public.
- **Encrypt**$(PP, \mathbf{V})$ The **Encrypt** algorithm takes as inputs the public parameter *PP* and a broadcast identity vector set **V**, and outputs a pair (Hdr, K), where Hdr is called the header, $K \in \mathcal{K}$ is a session key in the key space $\mathcal{K}$ and can be used to symmetrically encrypt data of arbitrary length. When a message $M \in \{0, 1\}^*$ will be broadcast to users in *Pref*(**V**), the broadcaster generates $(Hdr, K) \leftarrow$ **Encrypt**$(\mathrm{ID}, \mathbf{V})$, computes the encryption C_M of M under the session key K, and broadcasts the ciphertext as $(Hdr, \mathbf{V}, C_M)$.
- **KeyGen**$(PP, MSK, \mathbf{ID})$ The key generation algorithm takes as inputs the public parameter *PP*, the master key *MSK*, and an identity vector **ID**. It generates a user secret key $SK_{\mathbf{ID}}$ associated with the identity vector **ID**.
- **Delegate**$(PP, SK_{\mathbf{ID}'}, \mathrm{ID})$ The secret key delegation algorithm takes as inputs the public parameter PP, a broadcast secret key for an identity vector $\mathbf{ID}'$ with $\|\mathbf{ID}'\| < D$, and an identity ID. It delegates the secret key $SK_{\mathbf{ID}}$ for the identity vector $\mathbf{ID} = (\mathbf{ID}', ID)$, where we naturally have that $\|\mathbf{ID}\| = \|\mathbf{ID}'\| + 1 \leq D$.
- **Decrypt**$(PP, Hdr, \mathbf{V}, SK_{\mathbf{ID}}, \mathbf{ID})$ The **Decrypt** algorithm takes as inputs the public parameter *PP*, a header *Hdr* and

the corresponding broadcast vector set $\mathbf{V}$, and a secret key $SK_{\mathbf{ID}}$ associated with an identity vector $\mathbf{ID}$. If $\mathbf{ID} \in Pref(\mathbf{V})$, the algorithm outputs the session key $K \in \mathcal{K}$ which can be then used to decrypt C_M and recover M.

An HIBBE system must satisfy the standard consistency constraint, namely for all $D \leq N \in \mathbb{N}$, all $(PP, MSK) \leftarrow \mathbf{Setup}(D, N, \lambda)$, and all $SK_{\mathbf{ID}}$ generated by running $\mathbf{KeyGen}(PP, MSK, \mathbf{ID})$ or running $\mathbf{Delegate}(PP, SK_{\mathbf{ID}'}, ID)$ with $\mathbf{ID} = (\mathbf{ID}', ID)$, if $\mathbf{ID} \in Pref(\mathbf{V})$, then the session key can be correctly recovered as $\mathbf{Decrypt}(PP, Hdr, \mathbf{V}, SK_{\mathbf{ID}}, \mathbf{ID}) = K$, where $(Hdr, K) \leftarrow \mathbf{Encrypt}(PP, \mathbf{V})$.

3.3 Security definitions of HIBBE

We next define the security notion for HIBBE. To capture the most powerful attacks in a broadcast encryption schemes, we consider chosen-ciphertext security against adaptive adversaries in HIBBE. In such a security notion, the adversary is allowed to adaptively ask for several secret keys associated with any identity vectors **ID** of its choice and to issue decryption queries for its chosen broadcast header. Then, it can decide the identity vector set $\mathbf{V}^*$ that it wishes to attack. After deciding $\mathbf{V}^*$, the adversary is still allowed to obtain secret keys and decrypt the broadcast headers, with the constraints that it cannot obtain secret keys for $\mathbf{ID} \in Pref(\mathbf{V}^*)$ and cannot issue decryption query for the challenge broadcast header. We require that even such an adversary cannot reveal any useful information for the session key hidden in the header.

Formally, we define the indistinguishability against chosen-identity-vector-set and chosen-ciphertext attack (IND-CIVS-CCA2) in HIBBE by using the following game between an adversary $\mathcal{A}$ and a challenger. Both of them are given the parameters D, n, and λ as inputs.

Setup The challenger runs $\mathbf{Setup}(D, n, \lambda)$ to obtain a public parameter PP and gives it to the adversary $\mathcal{A}$.

Phase 1 The adversary $\mathcal{A}$ adaptively issues queries, each of which is one of the following forms:

- Secret key query for an identity vector **ID**. The challenger generates a secret key $SK_{\mathbf{ID}}$ for **ID** and returns it to the adversary.
- Decryption query for a header *Hdr* associated with an broadcast identity vector set **V**. The challenger first generates a secret key $SK_{\mathbf{ID}}$ associated with an identity vector $\mathbf{ID} \in Pref(\mathbf{V})$. Then, it responds with $K \leftarrow \mathbf{Decrypt}(PP, Hdr, \mathbf{V}, SK_{\mathbf{ID}}, \mathbf{ID})$.

Challenge When adversary $\mathcal{A}$ decides that **Phase 1** is over, it outputs to the challenger a challenge broadcast identity vector set $\mathbf{V}^*$ which contains all the users $\mathcal{A}$ wishes to attack. The broadcast identity vector set $\mathbf{V}^*$ should satisfy $\mathbf{ID} \notin Pref(\mathbf{V}^*)$ for any **ID** if adversary $\mathcal{A}$ has already queried the secret key $SK_{\mathbf{ID}}$. The challenger runs $(Hdr^*, K_0^*) \leftarrow \mathbf{Encrypt}(PP, \mathbf{V}^*)$. It then chooses a random session key $K_1 \overset{R}{\leftarrow} \mathcal{K}$ in the key space, and flips a random coin $b \overset{R}{\leftarrow} \{0, 1\}$. The challenger finally gives (Hdr^*, K_b^*) to adversary $\mathcal{A}$.

Phase 2 Adversary $\mathcal{A}$ continues to issue queries, each of which is one of the following forms:

- Secret key query for an identity vector **ID** with the constraint that $\mathbf{ID} \notin Pref(\mathbf{V}^*)$.
- Decryption query for a header Hdr corresponding to an broadcast identity vector set **V**, but with the constraint that $Hdr \neq Hdr^*$.

The challenger responds as in **Phase 1**.

Guess Eventually, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins in the game if $b = b'$.

The advantage of such an adversary $\mathcal{A}$ in attacking the (D, N)-HIBBE system with security parameter λ is defined as

$$\mathrm{Adv}_{\mathcal{A},D,N}^{\text{IND-CIVS-CCA2}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

Definition 2 A (D, N)-HIBBE system is IND-CIVS-CCA2 secure if for any polynomial time algorithm $\mathcal{A}$ who makes totally q_E secret key queries and q_D decryption queries, the advantage of breaking the security game defined above is at most ϵ negligible in λ, i.e.,

$$\mathrm{Adv}_{\mathcal{A},D,n}^{\text{IND-CIVS-CCA2}}(\lambda) < \epsilon$$

As usual, we define the indistinguishability against chosen-identity-vector-set and chosen-plaintext attack (IND-CIVS-CPA) in HIBBE as in the proceeding game, but preventing the adversary from issuing decryption queries in **Phase 1** and **Phase 2**. The advantage of such an adversary is defined as

$$\mathrm{Adv}_{\mathcal{A},D,n}^{\text{IND-CIVS-CPA}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

Definition 3 A (D, n)-HIBBE system is IND-CIVS-CPA secure if for any polynomial time algorithm $\mathcal{A}$ who makes totally q_E secret key queries and no decryption query, the advantage of breaking the security game defined above is at most ϵ negligible in λ, i.e.,

$$\mathrm{Adv}_{\mathcal{A},D,n}^{\text{IND-CIVS-CPA}}(\lambda) < \epsilon$$

It is challenging to achieve full/adaptive security in broadcast encryption schemes. Some weaker security notions have been introduced in these cryptographic primitives to bridge security proofs. These notions can be similarly considered in HIBBE settings. The first is static security [7,14], where the adversary should ahead of time commit to the challenge set it wishes to attack in an **Init** phase before the **Setup** phase. This security notion can be defined in HIBBE by adding the **Init** phase before the **Setup** phase, and the adversary should decide the challenge identity vector set $\mathbf{V}^*$ in the **Init** phase.

We also propose a new security definition named semi-static security in HIBBE. This security definition is used by recent (Identity-Based) broadcast encryption systems [20]. In this game the adversary must commit to an identity set $\overline{S}$ with $\left|\overline{S}\right| = poly(\lambda)$ at the **Init** phase before the **Setup** phase is started. In **Phase 1**, the adversary can only query for the secret keys associating with identity vectors **ID** with $S_{\mathbf{ID}} \nsubseteq \overline{S}$. When the adversary decides that **Phase 1** is over, it chooses a challenge identity vector set $\mathbf{V}^*$ satisfying $S_{\mathbf{V}^*} \subseteq \overline{S}$. After that, the adversary is still allowed to query for the secret keys associating with identity vectors **ID** satisfying $S_{\mathbf{ID}} \nsubseteq \overline{S}$. We refer to such an adversary $\mathcal{A}$ as an IND-ssCIVS-CPA adversary and accordingly define the advantage of such an adversary $\mathcal{A}$ in the IND-ssCIVS-CPA game with parameters D, N, and security parameter λ as

$$\mathrm{Adv}_{\mathcal{A},D,N}^{\text{IND-ssCIVS-CPA}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

4 Semi-statically secure HIBBE system with constant-size ciphertexts

In this section, we propose an IND-ssCIVS-CPA secure HIBBE with constant-size ciphertexts. Aside from stronger security than that in the HIBBE construction of [27], leveraging composite-order bilinear groups in it inevitably leads to more computation overhead [17]. Our IND-ssCIVS-CPA secure HIBBE offers an alternative of HIBBE implementation, i.e., based on prime-order bilinear groups to achieve a more efficient deployment. Our starting point is the Boneh-Boyen-Goh selective secure HIBE scheme [4]. We follow the technique in [27] to associate every user in our system, instead of every depth of hierarchy, with distinct random element for blinding its own identity vector in its secret key. This prevents users from revealing any information to other users' secret key from their owned ones. Surprisingly, we show that our construction achieves semi-static security result, which is stronger than the static security since the adversary just needs to partly decide which subset of S can be adaptively chosen to attack [20].

4.1 Our IND-ssCIVS-CPA secure construction

Given a security parameter $\lambda \in \mathbb{Z}^*$, run $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ to generate a prime p, two groups $\mathbb{G}$, $\mathbb{G}_T$ of order p, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Assume that any integer in $\mathbb{Z}_p$ can be used as an identity and the session key space is $\mathcal{K} = \mathbb{G}_T$. Assume also that users' positions $\mathrm{I}_{\mathbf{ID}} = \{i | ID_i \in S_{\mathbf{ID}}\}$ in HIBBE are publicly known. Our (D, n)-HIBBE scheme works as follows.

Setup(D, n, λ). Select a random generator $g \overset{R}{\leftarrow} \mathbb{G}$, a random exponent $\alpha \overset{R}{\leftarrow} \mathbb{Z}_p$, and set $g_1 = g^\alpha$. Next, pick random elements $g_2, g_3 \overset{R}{\leftarrow} \mathbb{G}$, and pick random elements $u_1, u_2, \ldots, u_n \overset{R}{\leftarrow} \mathbb{G}$. The public parameters PP and the master secret key MSK are

$$PP = (g, g_1, g_2, g_3, u_1, u_2, \ldots, u_n), \quad MSK = g_2^\alpha$$

KeyGen$(PP, MSK, \mathbf{ID})$ To generate a secret key for an identity vector **ID** of depth $k \leq D$ by using the masker key MSK, pick a random $r \overset{R}{\leftarrow} \mathbb{Z}_p$ and output

$$SK_{\mathbf{ID}} = \left(g_2^\alpha \cdot \left(g_3 \cdot \prod_{i\in\mathrm{I}} u_i^{ID_i}\right)^r, g^r, \left\{u_j^r\right\}_{j\in[1,n]\setminus\mathrm{I}}\right)$$

Delegate$(PP, SK_{\mathbf{ID}'}, \mathrm{ID})$ To generate a secret key for an identity vector $\mathbf{ID} = (\mathbf{ID}', \mathrm{ID})$ of depth $k+1$ by using the secret key

$$\begin{aligned} SK_{\mathbf{ID}'} &= \left(g_2^\alpha \cdot \left(g_3 \cdot \prod_{i\in\mathrm{I}'} u_i^{ID_i}\right)^{r'}, g^{r'}, \left\{u_j^{r'}\right\}_{j\in[1,n]\setminus\mathrm{I}'}\right) \\ &= \left(a_0, a_1, \left\{b_j\right\}_{j\in[1,n]\setminus\mathrm{I}'}\right) \end{aligned}$$

where r' is the random term used in $SK_{\mathbf{ID}'}$, and $\mathrm{I}_{\mathbf{ID}'} = \{i|\mathrm{ID}_i \in S_{\mathbf{ID}'}\}$. Denote $\mathrm{I}_{\mathbf{ID}} = \{i|\mathrm{ID}_i \in S_{\mathbf{ID}}\}$, pick a random $t \overset{R}{\leftarrow} \mathbb{Z}_p$ and output

$$SK_{\mathbf{ID}} = \left(\begin{array}{l} a_0 \left(b_i^{\mathrm{ID}}\right)_{i\in\mathrm{I}\setminus\mathrm{I}'} \left(g_3 \cdot \prod_{i\in\mathrm{I}} u_i^{\mathrm{ID}_i}\right)^t, \\ a_1 g^t, \left\{b_j u_j^t\right\}_{j\in[1,n]\setminus\mathrm{I}} \end{array}\right)$$

Note that by implicitly setting $r = r' + t \in \mathbb{Z}_p$, the delegated secret key can be written in the form

$$SK_{\mathbf{ID}} = \left(g_2^\alpha \cdot \left(g_3 \cdot \prod_{i\in\mathrm{I}} u_i^{\mathrm{ID}_i}\right)^r, g^r, \left\{u_j^r\right\}_{j\in[1,n]\setminus\mathrm{I}}\right)$$

which is a well-formed secret key as if it were generated by running **KeyGen**. Therefore, it is a properly distributed secret key associating with the identity vector $\mathbf{ID} = (\mathbf{ID}', ID)$.

Encrypt$(PP, \mathbf{V})$ To encapsulate a session key under the broadcast identity vector set **V** with users' position $\mathbb{I} = \{i : \mathrm{ID}_i \in S_{\mathbf{V}}\}$, pick a random $\beta \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute the header as

$$Hdr = (C_0, C_1) = \left(g^{\beta}, \left(g_3 \cdot \prod_{i\in\mathbb{I}} u_i^{\mathrm{ID}_i}\right)^{\beta}\right)$$

Output the session key as $K = e(g_1, g_2)^{\beta}$.

Decrypt(PP, Hdr, $\mathbf{V}$, $SK_{\mathbf{ID}}$, $\mathbf{ID}$). Given the broadcast header $Hdr = (C_0, C_1)$, any user with identity vector $\mathbf{ID} \in Pref(\mathbf{V})$ can use its secret key

$$SK_{\mathbf{ID}} = \left(a_0, a_1, \{b_j\}_{j\in[1,n]\backslash \mathrm{I}}\right)$$

to compute the session key

$$K = \frac{e\left(a_0 \cdot \prod_{i\in\mathbb{I}, i\notin \mathrm{I}} b_i^{\mathrm{ID}_i}, C_0\right)}{e(C_1, a_1)}$$

where $\mathrm{I} = \{i : \mathrm{ID}_i \in S_{\mathbf{ID}}\}$ and $\mathbb{I} = \{i : \mathrm{ID}_i \in S_{\mathbf{V}}\}$.
Consistency If $Hdr = (C_0, C_1)$ is a valid header, one can decapsulate the session key by noting the following equalities.

$$a_0 \prod_{i\in\mathbb{I}, i\notin\mathrm{I}} b_i^{\mathrm{ID}_i} = g_2^{\alpha} \cdot \left(g_3 \cdot \prod_{i\in\mathrm{I}} u_i^{\mathrm{ID}_i}\right)^r \cdot \left(\prod_{i\in\mathbb{I}, i\notin\mathrm{I}} b_i^{\mathrm{ID}_i}\right)^r$$
$$= g_2^{\alpha} \cdot \left(g_3 \cdot \prod_{i\in\mathbb{I}} u_i^{\mathrm{ID}_i}\right)^r$$

Therefore, the session key can be obtained as

$$K = \frac{e\left(a_0 \cdot \prod_{i\in\mathbb{I}, i\notin \mathrm{I}} b_i^{\mathrm{ID}_i}, C_0\right)}{e(C_1, a_1)}$$
$$= \frac{e\left(g_2^{\alpha} \cdot \left(g_3 \cdot \prod_{i\in\mathbb{I}} u_i^{\mathrm{ID}_i}\right)^r, g^{\beta}\right)}{e\left(\left(g_3 \cdot \prod_{i\in\mathbb{I}} u_i^{\mathrm{ID}_i}\right)^{\beta}, g^r\right)}$$
$$= \frac{e(g_2^{\alpha}, g^{\beta}) \cdot e\left(\left(g_3 \cdot \prod_{i\in\mathbb{I}} u_i^{\mathrm{ID}_i}\right)^r, g^{\beta}\right)}{e\left(\left(g_3 \cdot \prod_{i\in\mathbb{I}} u_i^{\mathrm{ID}_i}\right)^{\beta}, g^r\right)}$$
$$= e\left(g_2^{\alpha}, g^{\beta}\right) = e(g_1, g_2)^{\beta}$$

Performance The encryption procedure requires one pairing operation (which can be pre-computed), and the decryption procedure requires two pairing operations. The scheme is efficient in computation. For any subset of receivers, the header always contains two group elements. The public parameters, the master secret key, and the secret key (of each user) are linear with the number of the allowable users, comparable to the up-to-date (IB)BE schemes [4,20].

4.2 Security analysis

In this section, we prove our HIBBE scheme is semi-statically secure (IND-ssCIVS-CPA) under the Decision wBDHI* assumption.

Theorem 1 *Let $\mathbb{G}$ be a group $(of prime order p)$ equipped with a bilinear map. Suppose that the Decision (t, ϵ, N)-wBDHI* assumption holds in $\mathbb{G}$. Then the proposed HIBBE system is (t', q_E, ϵ) semi-statically secure (IND-ssCIVS-CPA) where*

$$t' < t - \Theta(\tau \cdot N \cdot q_E)$$

and q_E is the number of key queries and τ is a time unit.

Proof Suppose that there exists an adversary $\mathcal{A}$ breaking our (D, N)-HIBBE system in the IND-ssCIVS-CPA game with advantage ϵ. We construct an algorithm $\mathcal{B}$ that can solve the Decision N-wBDHI* problem in $\mathbb{G}$. The input of the algorithm $\mathcal{B}$ is the challenge tuple $(g, h, y_1, \ldots, y_N, T)$ of the Decision N-wBDHI* problem, where T is either $T \overset{R}{\leftarrow} \mathbb{G}_T$ or $T \leftarrow e(g, h)^{(\alpha^{N+1})}$. The algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Init Adversary $\mathcal{A}$ outputs an identity set $\overline{S}$ containing identities that $\mathcal{A}$ may decide to be challenged in the challenge phase. Algorithm $\mathcal{B}$ chooses $N \geq |\overline{S}|$. Establish indices $i \in [1, N]$ for identities in $\overline{S}$. We denote the index set by $\bar{\mathrm{I}} = \{i : \overline{\mathrm{ID}_i} \in \overline{S}\}$.

Setup. To generate public parameters PP, algorithm $\mathcal{B}$ first requests an instance of the Decision N-wBDHI* problem. Then, it chooses a random $\gamma \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets

$$g_1 = y_1 = g^{\alpha}, \quad g_2 = y_N \ldots g^{\gamma} = g^{\gamma + (\alpha^N)}$$

Then, algorithm $\mathcal{B}$ randomly chooses $\gamma_1, \ldots, \gamma_N \overset{R}{\leftarrow} \mathbb{Z}_p$. Set $u_i = g^{\gamma_i}$ if $i \in \bar{I}$. Otherwise set $u_i = g^{\gamma_i} \cdot y_{N-i+1}^{-1}$. Also, algorithm $\mathcal{B}$ randomly chooses $\delta \overset{R}{\leftarrow} \mathbb{Z}_p$, and sets $g_3 = g^{\delta}$. Finally, $\mathcal{B}$ gives the public parameters

$$PP = (g, g_1, g_2, g_3, u_1, \cdot, u_N)$$

to the adversary $\mathcal{A}$. The master key is

$$MSK = g_2^{\alpha} \leftarrow g^{\alpha(\alpha^N + \gamma)} = y_{N+1} \cdot y_1^{\gamma}$$

Note that $\mathcal{B}$ does not know the value of y_{N+1}. Hence, $\mathcal{B}$ does not know MSK.

Phase 1 Adversary $\mathcal{A}$ issues secret key queries for identity vector $\mathbf{ID} = (ID_1, ID_2, \ldots, ID_k)$ with $S_{\mathbf{ID}} \subseteq \overline{S}$. Then the identity vector $\mathbf{ID}$ contains at least one $ID_m \in S_{\mathbf{ID}}$ such that $\mathrm{ID}_m \notin \overline{S}$. We set m as the smallest index satisfying this condition. To respond to the query, algorithm $\mathcal{B}$ derives a secret key for the identity vector $\mathbf{ID}_m = (\mathrm{ID}_1, \ldots, \mathrm{ID}_m)$, from which $\mathcal{B}$ then constructs a secret key for the requested identity vector $\mathbf{ID}$.

Denote $\mathrm{I} = \{i : \mathrm{ID}_i \in S_{\mathbf{ID}_m}\}$. To generate the secret key for identity vector $(\mathrm{ID}_1, \ldots, \mathrm{ID}_m)$, algorithm $\mathcal{B}$ randomly chooses $\widetilde{r} \xleftarrow{R} \mathbb{Z}_p$ and then computes the secret key

$$\begin{aligned} SK_{\mathbf{ID}_m} &= \left(a_0, a_1, \{b_j\}_{j\in[1,N]\setminus \mathrm{I}}\right) \\ &= \left(g_2^{\alpha} \cdot \left(g_3 \cdot \prod_{i\in \mathrm{I}} u_i^{\mathrm{ID}_i}\right)^r, g^r, \left\{u_j^r\right\}_{j\in[1,N]\setminus \mathrm{I}}\right) \end{aligned}$$

where

$$r = \frac{\alpha^m}{\mathrm{ID}_m} + \widetilde{r} \in \mathbb{Z}_p.$$

For the first component a_0, we have

$$\begin{aligned} &\left(g_3 \cdot \prod_{i\in \mathrm{I}} u_i^{\mathrm{ID}_i}\right)^r \\ &= \left(g^{\delta} \cdot \left(\prod_{i\in \mathrm{I}, i\in \overline{\mathrm{I}}} u_i^{\overline{\mathrm{ID}_i}}\right) \cdot \left(\prod_{i\in \mathrm{I}, i\notin \overline{\mathrm{I}}} u_i^{\mathrm{ID}_i}\right)\right)^r \\ &= \left(g^{\delta} \cdot \left(\prod_{i\in \mathrm{I}, i\in \overline{\mathrm{I}}} g^{\overline{\mathrm{ID}_i}\gamma_i}\right) \cdot \left(\prod_{i\in \mathrm{I}, i\notin \overline{\mathrm{I}}} g^{\mathrm{ID}_i\gamma_i} \cdot y_{N-i+1}^{-\mathrm{ID}_i}\right)\right)^r \\ &= \left(\left(g^{\delta+\sum_{i\in \mathrm{I}} \overline{\mathrm{ID}_i}\gamma_i}\right) \cdot \left(y_{N-m+1}^{-\mathrm{ID}_m}\right)\right)^r = (c_1 \cdot c_2)^r \end{aligned}$$

Algorithm $\mathcal{B}$ can compute c_1 since $\mathcal{B}$ knows δ, γ_i and $\overline{\mathrm{ID}_i}$. Next, we have

$$\begin{aligned} c_2^r &= \left(y_{N-m+1}^{-\mathrm{ID}_m}\right)^{\frac{\alpha^m}{\mathrm{ID}_m} + \widetilde{r}} \\ &= y_{N-m+1}^{-\mathrm{ID}_m \cdot \widetilde{r}} \cdot y_{N-m+1}^{-\alpha^m} = y_{N-m+1}^{-\mathrm{ID}_m \cdot \widetilde{r}} \cdot y_{N+1}^{-1} \end{aligned}$$

Thus, $\mathcal{B}$ can compute a_0 since

$$\begin{aligned} a_0 &= \left(y_{N+1} \cdot y_1^{\gamma}\right) \cdot (c_1)^r \cdot \left(y_{N-m+1}^{-\mathrm{ID}_m \cdot \widetilde{r}} \cdot y_{N+1}^{-1}\right) \\ &= y_1^{\gamma} \cdot (c_1)^r \cdot y_{N-m+1}^{-\mathrm{ID}_m \cdot \widetilde{r}} \end{aligned}$$

$\mathcal{B}$ can compute all the other components in $SK_{\mathbf{ID}_m}$ since they do not involve y_{N+1}. Hence, algorithm $\mathcal{B}$ can generate the secret key for $\mathbf{ID}_m$. Algorithm $\mathcal{B}$ can further use this secret key to derive the secret key for the descendant identity $\mathbf{ID}$ and correctly response to $\mathcal{A}$'s request.

Challenge Adversary $\mathcal{A}$ outputs two equal-length messages $M_0, M_1 \in \mathbb{G}_T$, together with an identity vector set $\mathbf{V}^*$ such that $S_{\mathbf{V}^*} \subseteq \overline{S}$. Denote $\mathrm{I}^* = \{i : \mathrm{ID}_i^* \in S_{\mathbf{V}^*}\}$. Algorithm $\mathcal{B}$ computes the challenge header

$$Hdr^* = (C_0^*, C_1^*) = \left(h, h^{\delta + \sum_{i\in \mathrm{I}^*} \mathrm{ID}_i^* \cdot \gamma_i}\right)$$

where h and T comes from the Decision wBDHI* challenge tuple. Since $h = g^c$, where $c \in \mathbb{Z}_p$ is an unknown value, we have

$$\begin{aligned} C_1 &= h^{\delta + \sum_{i\in \mathrm{I}^*} \mathrm{ID}_i^* \cdot \gamma_i} = \left(g^{\delta + \sum_{i\in \mathrm{I}^*} \mathrm{ID}_i^* \cdot \gamma_i}\right)^c \\ &= \left(g^{\delta} \cdot \prod_{i\in \mathrm{I}^*} g^{\mathrm{ID}_i^* \cdot \gamma_i}\right)^c = \left(g^{\delta} \cdot \prod_{i\in \mathrm{I}^*} u_i^{\mathrm{ID}_i^*}\right)^c \end{aligned}$$

Algorithm $\mathcal{B}$ can compute C_1 since it knows γ_i for all $i \in \mathrm{I}^*$. The challenge session key is computed as

$$K^* = T \cdot e(y_1, h^{\gamma})$$

If $T = e(g, h)^{(\alpha^{N+1})}$, then we have that

$$\begin{aligned} T \cdot e(y_1, h^{\gamma}) &= \left(e(y_1, y_N) \cdot e(y_1, g^{\gamma})\right)^c \\ &= e(y_1, y_N g^{\gamma})^c = e(g_1, g_2)^c \end{aligned}$$

and K^* is a valid session key associating with Hdr^*. Otherwise, K^* is a random element from the session key space.

Phase 2 Adversary $\mathcal{A}$ can further issue $q_2 = q_E - q_1$ queries. $\mathcal{B}$ responds as in **Phase 1**.

Guess Finally, adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. Algorithm $\mathcal{B}$ also outputs b' to answer the Decision wBDHI* problem.

If $T = e(g, h)^{(\alpha^{N+1})}$, then $\mathcal{A}$ can successfully attack the HIBBE system, and the advantage that adversary $\mathcal{A}$ has is $\left|\Pr[b = b'] - \frac{1}{2}\right| \geq \epsilon$. If $T \xleftarrow{R} \mathbb{G}_T$, we have $\left|\Pr[b = b']\right| = \frac{1}{2}$. Therefore

$$\left| \begin{aligned} &\Pr\left[\mathcal{B}\left(g, h, y_1, \ldots, y_N, e(g, h)^{(\alpha^{N+1})}\right) = 0\right] \\ &- \Pr\left[\mathcal{B}\left(g, h, y_1, \ldots, y_N, T\right) = 0\right] \end{aligned} \right| = \epsilon$$

Algorithm $\mathcal{B}$ needs to answer q_E queries from $\mathcal{A}$. The total time that $\mathcal{B}$ takes is $t = t' + \Theta(\tau \cdot N \cdot q_{ID})$. This completes the proof of Theorem 1. □

5 Chosen-ciphertext secure HIBBE with short ciphertexts

In this section, we provide a technique to transform our CPA-secure HIBBE to a CCA2-secure one. We first give an overview of our transformation. We add a "dummy" user in our system with an on-the-fly "identity." The dummy user will be used just for the validity test, and no one is allowed to obtain the secret key for it. A one-time signature scheme with algorithms **SKeyGen**, **Sign**, **Verify** is additionally employed. We require that this scheme is secure in the sense of strong unforgeability. In encryption, the broadcaster first generates a key pair (vk, sk) for a one-time signature schemes. It treats the verification key vk as the on-the-fly "identity" of the "dummy" user. The broadcaster then encapsulates the session key under the identity vector set $\mathbf{V} \cup (vk)$ and generates the header $Hdr = (C_0, C_1)$. The resulting header Hdr is next signed using the signature key sk to obtain a signature σ. The final header consists of the verification key vk, the header Hdr, and the signature σ. When decrypting (vk, Hdr, σ), the receiver verifies the signature on the header Hdr using the verification key vk. If the signature is valid, the receiver can directly use its secret key $SK_{\mathbf{ID}}$ such that $\mathbf{ID} \in Pref(\mathbf{V})$ to decrypt Hdr and recovers the session key K.

This technique is inspired by the Canetti–Halevi–Katz technique [12], which was previously applied in IBE systems [1,5,6,34] to obtain CCA2-secure public key cryptosystems. Canetti et al. also remarked that their technique can be extended to achieve CCA2-secure HIBE from CPA-secure HIBE schemes by adding one extra hierarchy to the underlying HIBE systems. In our transformation, we just add one "dummy" user in the system to support validity text, instead of introducing one extra hierarchy of users to our HIBBE. In this way, CCA2 security is achieved only at a marginal cost of one extra user. Our transformation also implies a unique application of the Canetti–Halevi–Katz technique in the broadcast encryption cryptosystems [14].

5.1 Our IND-ssCIVS-CCA2 secure construction

For ease of description, we write the algorithms in our chosen plaintext secure HIBBE system as **CPA.Setup**, **CPA.KeyGen**, **CPA.Delegate**, **CPA.Encrypt**, and **CPA.Decrypt**. Our chosen ciphertext secure HIBBE system is described as follows.

CCA.Setup(D, n, λ). The system setup algorithm first runs **CPA.Setup**$(D, n+1, \lambda)$ to generate the public parameter and the master secret key

$$PP \leftarrow (g, g_1, g_2, g_3, u_1, \ldots, u_n, u_{n+1}), \quad MSK = g_2^{\alpha}$$

A secure one-time signature scheme with algorithms **SKeyGen**, **Sign**, **Verify** is also included in the public parameter. We stress that the dummy user is associated with the parameter u_{n+1} and no one is allowed to obtain its corresponding secret key.

CCA.KeyGen$(PP, MSK, \mathbf{ID})$ The algorithm directly calls **CPA.KeyGen**$(PP, MSK, \mathbf{ID})$ and returns the generated secret key $SK_{\mathbf{ID}}$.

CCA.Delegate$(PP, SK_{\mathbf{ID}}, ID)$ This algorithm is also identical with the CPA-secure ones. The algorithm runs **CPA. Delegate**$(PP, SK_{\mathbf{ID}}, ID)$ to delegate and return the secret key.

CCA.Encrypt$(PP, \mathbf{V})$ For the given identity vector set $\mathbf{V}$, denote $\mathbb{I} = \{i : \mathrm{ID}_i \in S_{\mathbf{V}}\}$. The encryption algorithm first calls **SKeyGen**(λ) to obtain a verification key vk and a signature key sk. It then picks a random $\beta \xleftarrow{R} \mathbb{Z}_N$ and computes

$$Hdr' = (C_0, C_1) = \left(g^{\beta}, \left(h \cdot u_{n+1}^{vk} \cdot \prod_{i \in \mathbb{I}} u_i^{\mathrm{ID}_i}\right)^{\beta}\right)$$

Finally, the algorithm calls $\sigma \leftarrow$ **Sign**(Hdr', sk) to obtain the signature σ. The final header is output as $Hdr = (vk, Hdr', \sigma)$.

CCA.Decrypt$(PP, Hdr, \mathbf{V}, SK_{\mathbf{ID}}, \mathbf{ID})$ Suppose the algorithm decrypts the header $Hdr = (vk, Hdr', \sigma)$ using the secret key

$$SK_{\mathbf{ID}} = \left(a_0, a_1, \{b_j\}_{j \in [1, n+1] \setminus \mathbb{I}}\right)$$

It first verifies the signature by checking whether the equation **Verify**$(Hdr', \sigma, vk) = 1$ holds. If not, the algorithm simply outputs $\perp$. Otherwise, the algorithm computes

$$K = \textbf{CPA.Decrypt}(PP, Hdr', \mathbf{V} \cap (vk), SK_{\mathbf{ID}}, \mathbf{ID})$$

to recover the session key K.

Consistency If the signature can be verified, then the header is correctly generated by the broadcaster since only the broadcaster holds the signing key sk related to the verification key vk. Note that by the prefix definition, $Pref(\mathbf{V} \cup (vk)) = Pref(\mathbf{V}) \cup (vk)$. Therefore, for any identity vector $\mathbf{ID} \in Pref(\mathbf{V})$, we have that $\mathbf{ID} \in Pref(\mathbf{V}) \cup (vk)$. Therefore, the decryption algorithm can decapsulate the session key by invoking the underlying **CPA.Decrypt**$(PP, Hdr', \mathbf{V} \cup (vk), SK_{\mathbf{ID}}, \mathbf{ID})$.

5.2 Security analysis

We first give the outline of our security proof. We now allow decryption queries in **Phase 1** and **Phase 2** of the security definition. In the decryption query, the simulator first verifies the signature to determine whether the header is valid. If the verification passes, the simulator generates a secret key $SK_{(vk)}$ for the identity vector (vk) associating with the random element u_{n+1}. Note that it is a secret key that will never be really used in the CCA2-secure system since this secret key is for the dummy user so that no one is allowed to obtain it. However, it is a valid key in the underlying CPA-secure system. Also, for any identity vector set $\mathbf{V} \cup (vk)$, we have that $(vk) \in Pref(\mathbf{V} \cup (vk))$. Therefore, the simulator can correctly respond to the decryption query by decrypting the header using $SK_{(vk)}$. In the challenge phase, the simulator generates a new signature and verification key pair $(sk, vk) \leftarrow$ **SKeyGen**(λ) and creates a header $Hdr'^{*} = (C_0^*, C_1^*)$ for the challenge identity vector set $\mathbf{V}^* \cup (vk^*)$. The header is then signed using sk^*, and the final challenge header is output as $Hdr^* = (vk^*, Hdr'^{*}, \sigma^*)$. Since the signature key generation algorithm **SKeyGen** is a randomized algorithm, the adversary is not able to make any valid decryption queries that would require the simulator to use an identity vector set $\mathbf{V} \cup (vk)$ with $vk = vk^*$. Note that the adversary cannot issue secret key query for the dummy user because the dummy user is not available before the simulator produces the challenge ciphertext. Hence, the simulation can be done by invoking the underlying CPA-secure HIBBE scheme.

Formally, the chosen-ciphertext security of the above scheme is guaranteed by the following Theorem.

Theorem 2 *Suppose that the HIBBE scheme proposed in Sect. 4 achieves semi-static and chosen plaintext security. Assume also that the employed signature scheme is a strongly unforgeable one-time signature scheme. Then the HIBBE scheme proposed in Sect. 5 achieves semi-static and chosen-ciphertext security.*

Proof Assume that there exists a polynomial time adversary $\mathcal{A}$ that can break the IND-ssCIVS-CCA2 security of the (D, n)-HIBBE scheme proposed in Sect. 5. We construct a polynomial time algorithm $\mathcal{B}$ breaking the IND-ssCIVS-CPA security of the $(D, n+1)$-HIBBE scheme proposed in Sect. 4 with the help of adversary $\mathcal{A}$. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows. **Init** Adversary $\mathcal{A}$ outputs an identity set $\overline{S}$ containing identities that $\mathcal{A}$ may decide to challenge. Algorithm $\mathcal{B}$ runs $(sk^*, vk^*) \leftarrow$ **SKeyGen**(λ) to generate a challenge signature and verification key pair. It chooses $n \leq |\overline{S}|$, establishes indices $i \in [1, n]$ for the identities in $\overline{S}$, and sets the index $n+1$ for the identity vk^*. Algorithm $\mathcal{B}$ finally sends $\overline{S} \cup vk^*$ to the challenger.

Setup The challenger runs **Setup** algorithm with the inputs $(D, n+1, \lambda)$ and sends the public parameter $PP = (g, g_1, g_2, g_3, u_1, \ldots, u_{n+1})$ to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ directly forwards PP to adversary $\mathcal{A}$.

Phase 1 Adversary $\mathcal{A}$ adaptively issues secret key and decryption queries:

- Secret key queries with the identity vector **ID**. Algorithm $\mathcal{B}$ simply submits **ID** to the challenger to obtain the secret key $SK_{\mathbf{ID}}$ and sends it to adversary $\mathcal{A}$. Since the secret key takes the same form in both schemes, the secret key $SK_{\mathbf{ID}}$ is also a valid key for adversary $\mathcal{A}$.
- Decryption queries with the header Hdr and the identity vector set $\mathbf{V}$. Parses Hdr to be (vk, Hdr', σ). Algorithm $\mathcal{B}$ first verifies the signature by calling **Verify**(Hdr', σ, vk). If the algorithm **Verify** outputs 0, the header is invalid and algorithm $\mathcal{B}$ returns with $\perp$. Otherwise, algorithm $\mathcal{B}$ checks whether $vk = vk^*$. If so, algorithm $\mathcal{B}$ stops the simulation and aborts. If $vk \neq vk^*$, algorithm $\mathcal{B}$ creates an identity vector (vk), assigns the index of the identity vk as $n+1$, and issues a secret key query for (vk) to the challenger. Since $vk \neq vk^*$, we have that $vk \notin \overline{S} \cup vk^*$ so that the challenger can generate a secret key for (vk). Also, $(vk) \in Pref(\mathbf{V} \cup vk)$. Therefore, algorithm $\mathcal{B}$ can use this secret key to decrypt Hdr' and recovers the session key. The session key is finally returned to adversary $\mathcal{A}$.

Challenge Adversary $\mathcal{A}$ outputs the challenge identity vector set $\mathbf{V}^*$ to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ sends $\mathbf{V}^* \cup vk^*$ to the challenger, which returns the challenge header Hdr'^{*} and the challenge session key K^*. Algorithm $\mathcal{B}$ then calls $\sigma^* \leftarrow$ **Sign**(Hdr'^{*}, sk) to obtain the signature, and returns the final challenge header as $Hdr^* = (vk^*, Hdr'^{*}, \sigma^*)$. The challenge session key K^* is sent to adversary $\mathcal{A}$.

Phase 2 Adversary $\mathcal{A}$ continues to make two kinds of queries. Algorithm $\mathcal{B}$ responds as in **Phase 1**.

Guess Finally, adversary $\mathcal{A}$ outputs a guess b'. Algorithm $\mathcal{B}$ also outputs b' as its own guess.

If algorithm $\mathcal{B}$ does not abort during the game, algorithm $\mathcal{B}$ provides a perfect simulation for adversary $\mathcal{A}$. Therefore, if adversary $\mathcal{A}$ can break the CCA2 security of the proposed HIBBE scheme with advantage ϵ, then algorithm $\mathcal{B}$ has the same advantage to break the CPA security of the HIBBE scheme proposed in Sect. 4. The event leading to the abortion is that adversary $\mathcal{A}$ issues a decryption query with the verification key $vk = vk^*$, which may happens with negligible probability. Therefore, the actual advantage of algorithm $\mathcal{B}$ is negligibly close to ϵ. □

6 Performance analysis

6.1 Theoretical analysis

We compare our practical HIBBE schemes with the HIBBE scheme proposed in [27]. Both of them are required to be in

Table 2 (D, n)-HIBBE performance comparisons

	CPA-HIBBE in [27]	CPA-HIBBE in Sect. 4	CCA2-HIBBE in Sect. 5
Order	$N = p_1 p_2 p_3$	p	p
PP	$(n+4)\ell_N$	$(n+4)\ell_p$	$(n+4)\ell_p$
$SK_{\mathbf{ID}}$	$(n+1)\ell_N$	$(n+1)\ell_p$	$(n+1)\ell_p$
Hdr	$2\ell_N$	$2\ell_p$	$(2+\vert vk\vert+\vert\sigma\vert)\ell_p$
Encrypt	$(\vert S_{\mathbf{V}}\vert+2)\cdot\tau_{e_N}+\vert S_{\mathbf{V}}\vert\cdot\tau_{m_N}$	$(\vert S_{\mathbf{V}}\vert+2)\cdot\tau_{e_p}+\vert S_{\mathbf{V}}\vert\cdot\tau_{m_p}$	$(\vert S_{\mathbf{V}}\vert+3)\tau_{e_p}+(\vert S_{\mathbf{V}}\vert+1)\tau_{m_p}+\tau_s$
Decrypt	$(\vert S_{\mathbf{V}}\vert+1)\tau_{e_N}+\vert S_{\mathbf{V}}\vert\tau_{m_N}+2\tau_{p_N}$	$(\vert S_{\mathbf{V}}\vert+1)\tau_{e_p}+\vert S_{\mathbf{V}}\vert\tau_{m_p}+2\tau_{p_p}$	$(\vert S_{\mathbf{V}}\vert+2)\tau_{e_p}+(\vert S_{\mathbf{V}}\vert+1)\cdot\tau_{m_p}+2\tau_{p_p}+\tau_v$

Table 3 Parameters in our experiments

	Composite order	Prime order
Curve	$y^2 = x^3 + x$	$y^2 = x^3 + x$
Curve type	Type A1	Type A
Symmetry	True	True
Order	$N = p_1 p_2 p_3 = 3 \mod 4$	$p = 3 \mod 4$
Security level	$\log p_i \approx 512, i = 1, 2, 3$	$\log p \approx 512$

Table 4 Experiment platform

Test platform	
CPU series	Intel Xeon E5-1620
CPU clock speed	3.70 GHz
RAM	16 GB
Operate system	Ubuntu 13.10
Linux Kernel	3.11.0
JDK version	Open JDK 1.6.0
jPBC version	2.0.0

the KEM setting for ease of our comparisons. The parameter size, the secret key size, and the broadcast header size in our CPA-secure HIBBE scheme remain the same as those of the scheme in [27]. The encryption algorithm and the decryption algorithm also involve the same multiplication and exponentiation operations in the based groups. Recall that element operations in prime-order bilinear groups are nearly 40–50 times faster than those in composite-order bilinear groups in the same security level [17]. Therefore, it is expected that our practical HIBBE schemes is more efficient to be applied in practice.

Our CCA2-secure transformation keeps the resulting HIBBE scheme practical. Compared with the CPA-secure one, the broadcast header additionally involves elements for verification key vk and the signature σ, both of which have constant size in well-designed one-time signature schemes. The encryption algorithm does one more exponentiation operation, one more multiplication operation, and one signing operation for the one-time signature. The decryption algorithm does one more exponentiation operation, one more multiplication operation, and the verification operation to verify the signature.

Table 2 shows efficiency comparisons among HIBBE schemes proposed in [27], in Sect. 4 and in Sect. 5. In Table 2, ℓ_N and ℓ_p are the binary length of elements in groups of order N and p, respectively. We denote τ_{e_N}, τ_{e_p} as the respective time units for one exponentiation in a base group $\mathbb{G}$ of order N and p, similarly, τ_{m_N}, τ_{m_p} as the time unit for one multiplication operation in $\mathbb{G}$ of N and p, and τ_{p_N}, τ_{p_p} as the time for one pairing operation over bas groups of order N and p. For the one-time signature scheme, we denote τ_s as the time for signing, and τ_v as the time for signature verification. It can be seen from Table 3 that: (1) the CPA-secure HIBBE scheme we proposed is more efficient than that of [27]; (2) our CCA2-secure HIBBE transformation is compact and practical.

6.2 Experimental analysis

6.2.1 Parameters and platform

We implemented our two proposed HIBBE schemes and the HIBBE scheme proposed in [27] with the Java Pairing-Based Cryptography Library (JPBC, http://gas.dia.unisa.it/projects/jpbc/index.html) to evaluate their performances. We use elliptic curve Type A for the prime-order bilinear groups, and Type A1 for the composite-order bilinear groups, both of which have the same elliptic curve expression $y^2 = x^3 + x$ for the Tate symmetric pairings. All primes used in our experiments are nearly 512-bit large that are randomly generated by the system. Table 3 concludes the parameters we use in our experiments.

For the experiment platform, we use a 3.70 GHz Intel Xeon E5-1620 platform with 16GB RAM running 32-bit Ubuntu 13.10 and Linux Kernel version 3.11.0. Table 4 summaries the configuration of our experiment platform in details.

6.2.2 Benchmarks

A benchmark comparisons between the composite-order and the prime-order bilinear groups can intuitively show the performance gap between these two types of groups. We test the basic group operations in bilinear groups on our exper-

Table 5 Benchmark on JPBC

Operation	Composite order (ms)	Prime order (ms)	Ratio
#Pairing$(\cdot, \cdot)$	864.42	12.39	69.77
#Mul$(\cdot)$ in $\mathbb{G}_1$	0.41	0.07	5.86
#Exp$(\cdot)$ in $\mathbb{G}_1$	624.05	12.60	49.53
#Mul$(\cdot)$ in $\mathbb{G}_2$	0.48	0.07	6.86
#Exp$(\cdot)$ in $\mathbb{G}_2$	623.43	12.65	49.28
#Mul$(\cdot)$ in $\mathbb{G}_T$	0.09	0.02	4.50
#Exp$(\cdot)$ in $\mathbb{G}_T$	88.36	1.55	57.01

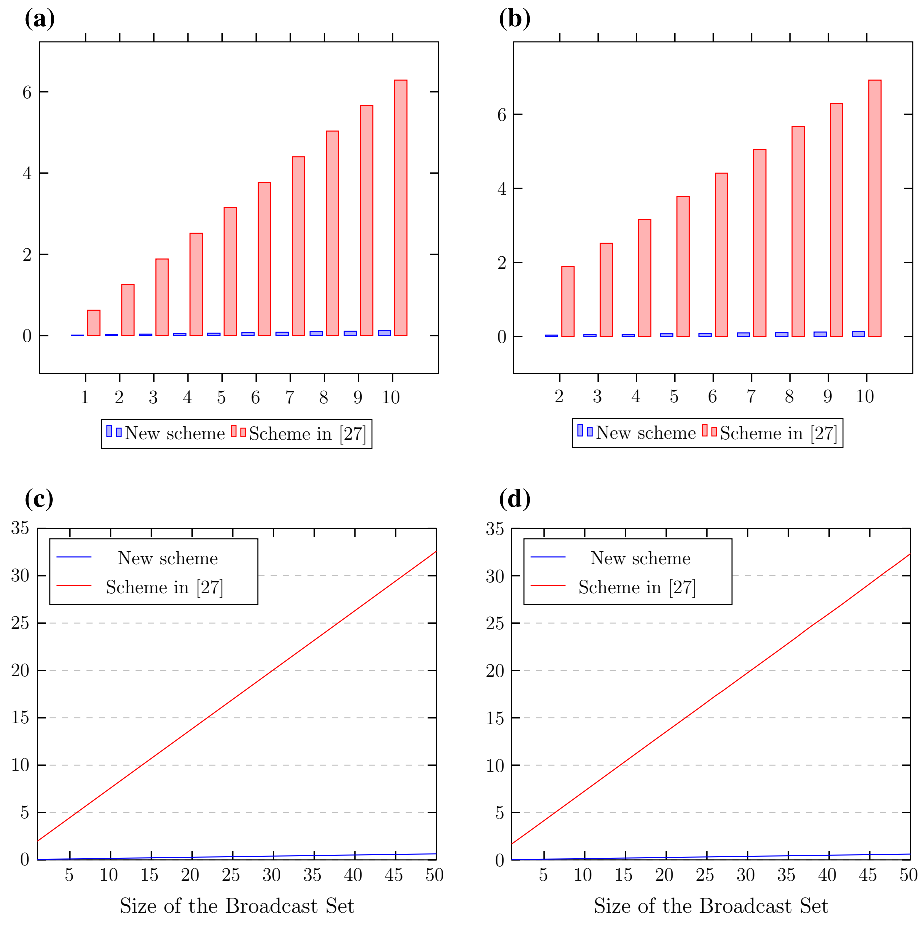

Fig. 4 Comparisons of the new CPA-secure HIBBE scheme and the existing CPA-secure HIBBE scheme in [27]. **a** Secret key generation time (s), **b** secret key delegation time (s), **c** encryption time (s), **d** decryption time (s)

iment platform, including pairing operation, exponentiation operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, multiplication operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Table 5 lists the timing performances of these group operations running on our experiment platform. It can be seen that there is a significant gap between these two types of bilinear groups with the same security level, especially for time-consuming operations. The pairing operation in composite-order bilinear groups is nearly 80 times slower than that in prime-order bilinear groups. The time ratio for running exponentiation operations in these two types of bilinear groups is nearly 50.

6.2.3 Experiment specification

For the HIBBE schemes, the secret key generation, the secret key delegation, the session key encapsulation, and the session key decapsulation involve the identity vectors of the users. To capture this in our experiments, we first generated the a collection of 50 distinct identities of the form $\{\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_{50}\}$. Then, we assign the identity vectors as the form $(\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_d)$, where d is the depth of hierarchy for the user that ranges from 1 to 10. The broadcast identity vector sets $\mathbf{V}$ in encryption and decryp-

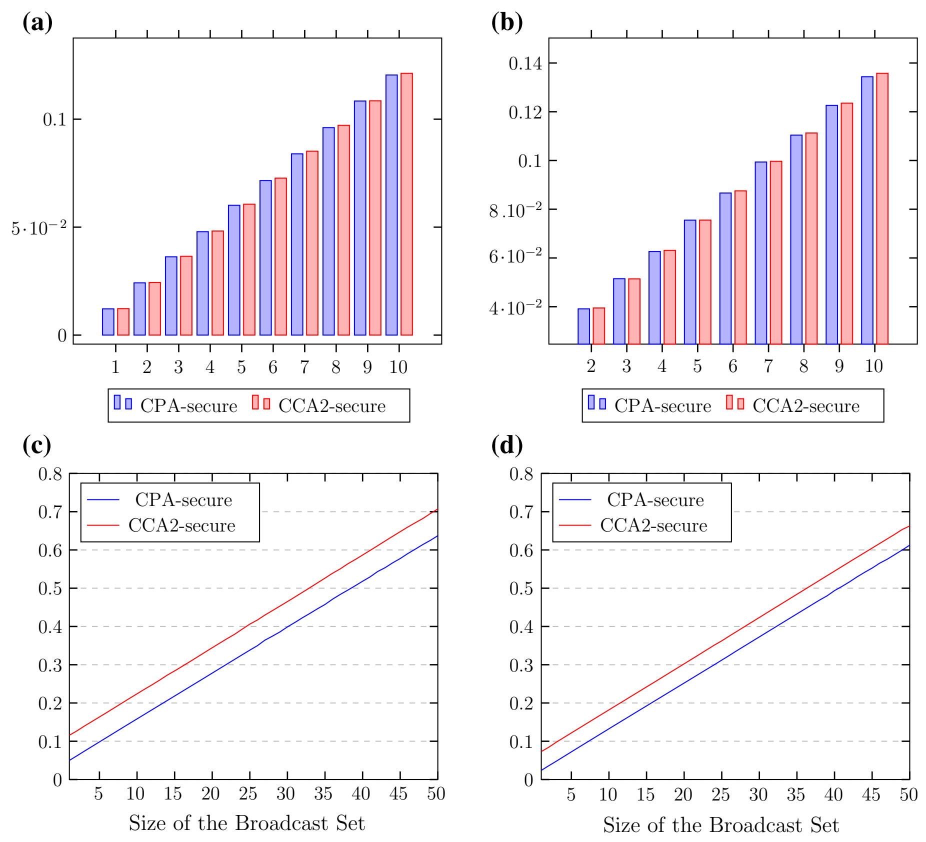

Fig. 5 Efficiency comparisons of new CPA-secure and CCA2-secure HIBBE schemes. **a** Secret key generation time (s), **b** secret key delegation time (s), **c** encryption time (s), **d** decryption time (s)

tion are randomly set with $|S_{\mathbb{V}}|$ ranging from 1 to 50 and $S_{\mathbb{V}} \subseteq \{\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_{50}\}$.

Recall that a secure one-time signature scheme is required in the CCA2-secure HIBBE proposed in Sect. 5. To capture this requirement, we implement the Boneh-Boyen signature scheme [3] and apply it as the building block of our proposed CCA2-secure HIBBE scheme. Such a scheme can be proven secure in the sense of strong unforgeability in the standard model assuming that the strong Diffie–Hellman assumption holds in the based group $\mathbb{G}$. Therefore, leveraging such a signature does not degrade the security level.

We test the time consumptions required for secret key generation with depth of users ranging from 1 to 10, for secret key delegation with depth of users ranging from 2 to 10, for encryption and decryption with size of receivers ranging from 1 to 50, in the HIBBE scheme proposed in [27], in Sect. 4 and in Sect. 5. We repeated each of our experiments 100 times and averaged the result to flatten experimental variances.

6.2.4 Experiment result analysis

We first provide performance results achieved by HIBBE schemes built respectively on prime-order bilinear groups (in Sect. 4) and on composite-order bilinear groups (in [27]). Figure 2 displays measurements of these two schemes. We separately demonstrate **KeyGen** time in Fig. 4a, **Delegate** time in Fig. 4b, **Encrypt** time in Fig. 4c, and **Decrypt** time in Fig. 4d. It can be seen that our schemes have significant improvements on the composite-order HIBBE scheme [27] in performance. Concretely, our CPA-secure HIBBE is about 50 times more efficient than the CPA-secure HIBBE in [27], which is reasonable by considering the benchmark gap in the two types of bilinear groups.

We next consider the costs of achieving CCA2-secure HIBBE. Figure 3 compares the time consumptions for CPA-secure HIBBE (in Sect. 4) and for its CCA2-secure transformation (in Sect. 5). We similarly illustrate **Key-Gen** time in Fig. 5a, **Delegate** time in Fig. 5b, **Encrypt** time in Fig. 5c, and **Decrypt** time in Fig. 5d. The results show that the additional costs for CCA2-secure HIBBE is constant and marginal. In fact, approximately additional 0.5 ms is needed for secret key generation and secret key delegation algorithms that are independent of the depth of hierarchy of the underlying users. Meanwhile, regardless of the size of users to be broadcasted, it additionally takes about 50 ms for CCA2-secure encryption and decryption, which are caused by the one-time signature singing procedures and the signature verification procedures, respectively. Our CCA2-secure transformation is at only a marginal cost in practice.

7 Conclusion

We investigated practical HIBBE with chosen-ciphertext security. We achieved this goal in the standard model in two steps. We first presented an IND-ssCIVS-CPA secure HIBBE scheme that is built from prime-order bilinear groups. Compared with the existing HIBBE scheme based on composite-order bilinear groups, procedures in our construction are more efficient due to the efficiency of group operations in prime-order group settings. Then, we constructed an IND-ssCIVS-CCA2 secure HIBBE scheme from our proposed IND-ssCIVS-CPA secure HIBBE scheme at the cost of one-time signatures. Our CCA2-secure construction is simple and efficient, i.e., only one on-the-fly dummy user is needed. We formally analyzed the security of our HIBBE, followed by theoretical and experimental analyses that illustrated high efficiency of our HIBBE schemes.

Acknowledgments This paper is supported by the National Key Basic Research Program (973 program) through project 2012CB315905, by the Natural Science Foundation through projects 61370190, 61272501, 61173154, 61472429, 61402029, and 61003214, and by the Beijing Natural Science Foundation through project 4132056, the Fundamental Research Funds for the Central Universities, and the Research Funds (No. 14XNLF02) of Renmin University of China, Innovation Fund of China Aerospace Science and Technology Corporation Satellite Application Research Institute through project 2014-CXJJ-TX-10 and the Open Research Fund of Beijing Key Laboratory of Trusted Computing.

References

1. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: EUROCRYPT'04, LNCS, vol. 3494, pp. 223–238. Springer (2004)
2. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: CRYPTO'04, LNCS, vol. 3152, pp. 443–459. Springer (2004)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: EUROCRYPT'04, LNCS, vol. 3027, pp. 56–73. Springer (2004)
4. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: EUROCRYPT'05, LNCS, vol. 3494, pp. 440–456. Springer (2005)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: CRYPTO'01, LNCS, vol. 2139, pp. 213–229. Springer (2001). doi:10.1007/3-540-44647-8_13
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)
7. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: CRYPTO'05, LNCS, vol. 3621, pp. 258–275. Springer (2005)
8. Boneh, D., Hamburg, M.: Generalized identity based and broadcast encryption schemes. In: ASIACRYPT'08, LNCS, vol. 5350, pp. 455–470. Springer (2008)
9. Boneh, D., Katz, J.: Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In: CT-RSA'05, LNCS, vol. 3376, pp. 87–103. Springer (2005)
10. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS'05, pp. 320–329. ACM Press, New York (2005)
11. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: EUROCRYPT'03, LNCS, vol. 2656, pp. 255–271. Springer (2003)
12. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: EUROCRYPT'04, LNCS, vol. 3027, pp. 207–222. Springer (2004)
13. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: CRYPTO'98, LNCS, vol. 1462, pp. 13–25. Springer (1998)
14. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: ASIACRYPT'07, LNCS, vol. 4833, pp. 200–215. Springer (2007)
15. Deng, H., Wu, Q., Qin, B., Chow, S.S.M., Domingo-Ferrer, J., Shi, W.: Tracing and revoking leaked credentials: accountability in leaking sensitive outsourced data. In: ACM AISACCS'14, pp. 425–434. ACM Press, New York (2014)
16. Deng, H., Wu, Q., Qin, B., Mao, J., Liu, X., Zhang, L., Shi, W.: Who is touching my cloud. In: ESORICS'14, LNCS, vol. 8712, pp. 362–379. Springer (2014)
17. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: EUROCRYPT'10, LNCS, vol. 6110, pp. 44–61. Springer (2010)
18. Gentry, C.: Practical identity-based encryption without random oracles. In: EUROCRYPT'06, LNCS, vol. 4004, pp. 445–464. Springer (2006)
19. Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: ASIACRYPT'02, LNCS, vol. 2501, pp. 548–566. Springer (2002)
20. Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: EUROCRYPT'09, LNCS, vol. 5479, pp. 171–188. Springer (2009)
21. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: IEEE HICCS'00. IEEE Press (2000)
22. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: EUROCRYPT'02, LNCS, vol. 2332, pp. 466–481. Springer (2002)
23. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure hibe with short ciphertexts. In: TCC'10, LNCS, vol. 5978, pp. 455–479. Springer (2010)
24. Lewko, A., Waters, B.: Unbounded hibe and attribute-based encryption. In: EUROCRYPT'11, LNCS, vol. 6632, pp. 547–567. Springer (2011)
25. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: CRYPTO'12, LNCS, vol. 7417, pp. 180–198. Springer (2012)
26. Lewko, A., Waters, B.: Why proving hibe systems secure is difficult. In: EUROCRYPT'14, LNCS, vol. 8441, pp. 58–76. Springer (2014)
27. Liu, W., Liu, J., Wu, Q., Qin, B.: Hierarchical identity-based broadcast encryption. In: ACISP'14, LNCS, vol. 8544, pp. 242–257. Springer (2014)
28. Liu, W., Liu, J., Wu, Q., Qin, B., Zhou, Y.: Practical direct chosen ciphertext secure key-policy attribute-based encryption with public ciphertext test. In: ESORICS'14, LNCS, vol. 8713, pp. 91–108. Springer (2014)
29. Liu, W., Liu, X., Wu, Q., Qin, B.: Experimental performance comparisons between (h)ibe schemes over composite-order and prime-order bilinear groups. In: IBCAST'14, pp. 203–209. IEEE Press (2014)
30. Qin, B., Deng, H., Wu, Q., Domingo-Ferrer, J., Naccache, D., Zhou, Y.: Flexible attribute-based encryption applicable to secure e-healthcare records. Int. J. Inf. Secur. **14**(1) (2015). doi:10.1007/s10207-014-0272-7

31. Shamir, A.: Identity-based cryptosystems and signature schemes. In: CRYPTO'84, LNCS, vol. 196, pp. 47–53. Springer (1985)
32. Wang, X., Mu, Y.: A secure mobility support scheme for 6loWPAN wireless sensor networks. Secur. Comm. Netw. **7**(3), 641–652 (2014)
33. Wang, Y., Wu, Q., Wong, D.S., Qin, B., Chow, S.S.M., Liu, Z., Tan, X.: Securely outsourcing exponentiations with single untrusted program for cloud storage. In: ESORICS'14, LNCS, vol. 8712, pp. 326–343. Springer (2014)
34. Waters, B.: Efficient identity-based encryption without random oracles. In: EUROCRYPT'05, LNCS, vol. 3494, pp. 114–127. Springer (2005)
35. Waters, B.: Dual system encryption: realizing fully secure ibe and hibe under simple assumptions. In: CRYPTO'09, LNCS, vol. 5677, pp. 619–636. Springer (2009)
36. Wiangsripanawan, R., Susilo, W., Safavi-Naini, R.: Achieving mobility and anonymity in ip-based networks. In: CANS'07, LNCS, vol. 4856, pp. 60–79. Springer (2007)