Chosen Ciphertext Security for (Hierarchical) Identity-Based Matchmaking Encryption

Sohto Chiku[1*], Keisuke Hara[1,2*] and Junji Shikata[1*]

[1*]Yokohama National University, Japan.
[2*]National Institute of Advanced Industrial Science and Technology (AIST), Japan.

*Corresponding author(s). E-mail(s): chiku-sohto-tw@ynu.jp; hara-keisuke@aist.go.jp; shikata-junji-rb@ynu.ac.jp;

Abstract

Identity-based matchmaking encryption (IB-ME) is an advanced encryption scheme that enables a sender and a receiver to specify each of identity. In general, from the aspect of abilities for adversaries, we have two flavors of security for encryption schemes chosen plaintext attacks (CPA) security and chosen ciphertext attacks (CCA) security. Compared to CPA security, CCA security can capture active adversaries, then it has been recognized as a desirable one.
In this paper, we investigate the CCA security for IB-ME. Concretely, we provide the following three contributions. (1) A method to obtain a CCA secure IB-ME scheme in the standard model based on our new primitive called *hierarchical* IB-ME (HIB-ME) along with strong one-time signature (2) A construction of HIB-ME based on hierarchical identity-based encryption and hierarchical identity-based signature (3) A variant of the first method to get an IB-ME scheme satisfying slightly tweaked CCA security solely based solely on a CPA secure IB-ME scheme (without strong one-time signature). We believe that this new type of CCA security is a reasonable one for IB-ME.

Keywords: identity-based matchmaking encryption, chosen ciphertext security, generic construction, pairing based cryptography

1 Introduction

1.1 Background and Motivation

Identity-based Matchmaking Encryption.

Identity-based matchmaking encryption (IB-ME) proposed by Ateniese et al. [1] is a novel extension of ordinary encryption system that both sender and receiver can specify appropriate identities which the other party should satisfy in order for the message to be revealed. More specifically, in IB-ME, as setup phase, each sender (resp., receiver) is provided a secret encryption (resp., decryption) key associated to its identity σ (resp., ρ) by the authority called key generation center (KGC). Then, when a sender generates a ciphertext CT using encryption key ek_σ, in addition to a plaintext m, it selects the target identity of receiver ρ. Upon receiving a ciphertext CT from the sender with his identity σ, a receiver who has a decryption key of ρ and selects a sender identity σ can decrypt the ciphertext CT. As security requirements, IB-ME should satisfy two properties: *privacy* and *authenticity*. Roughly, if identity requirements by senders and receivers does not match, privacy guarantees that any information of a plaintext and an identity does not leak from a ciphertext. Also, authenticity ensures that only the sender who has an encryption key associated with his identity σ can generate a ciphertext associated with σ. To show the usefulness of IB-ME, Ateniese et al. [1] demonstrates that a privacy-preserving bulletin board system (over a Tor network) can be realized based on IB-ME. In that system, users (who might belong to different organizations) can communicate secretly through this bulletin board or collect information from anonymous sources.

Security against Chosen Ciphertext Attacks.

From the aspect of abilities for adversaries, we have two flavors of security for encryption schemes(e.g., public key encryption and identity-based encryption): chosen plaintext attacks (CPA) security and chosen ciphertext attacks (CCA) security [2–4]. CCA security is stronger than CPA security in the sense that an adversary is given an ability of getting decryption results of ciphertexts (except for target ciphertext). In general, CCA security is more desirable in practice since it takes active adversaries and implies non-malleability [5]. Actually, CCA security notions for various advanced encryption primitives [6–10] have been proposed so far.

Motivation.

Following the previous works, in this paper, we focus on how to achieve CCA security on IB-ME.[1] One might think that we can get an efficient CCA secure IB-ME scheme (in the random oracle model) easily by extending the Fujisaki-Okamoto (FO) conversion [12] in a generic manner. However, this approach does not work immediately. Roughly, this is because a receiver cannot execute the re-encryption check when decrypting a ciphertext since he does not have a (secret) encryption key of a sender. Thus, we have to explore another approach to obtain CCA secure IB-ME schemes.

[1] Note that this paper is the first work to consider CCA security for IB-ME, and we have a follow-up study [11] on another approach to obtain more efficient CCA secure IB-ME schemes.

1.2 Our Contribution

Based on the above motivation, this paper gives the following three technical contributions.

A New Primitive: Hierarchical Identity-based Matchmaking Encryption.
Toward a CCA secure IB-ME scheme in the *standard* model, we devise a new extension of IB-ME called *hierarchical identity-based matchmaking encryption* (HIB-ME). Roughly, HIB-ME is an extension of IB-ME in the sense that it enables senders (resp., receivers) to generate encryption keys (resp., decryption keys) for their children's identities. We show that a CPA secure HIB-ME scheme can be obtained by combining hierarchical identity-based encryption (HIBE) and hierarchical identity-based signature (HIBS) [13].

CCA Secure (H)IB-ME in the Standard Model.
Depending on the above contribution, we provide the first CCA secure (H)IB-ME scheme in the standard model. Our construction is obtained by extending the Canetti-Halevi-Katz (CHK) conversion technique [14] based on CPA secure HIB-ME and strong one-time signature. Regarding the efficiency, for example, the ciphertext size is just almost twice of the underlying CPA secure (H)IB-ME scheme. From the previous works [15–20], we can realize our generic construction over bilinear groups or lattices. Actually, based on the Blazy-Kiltz-Pan anonymous HIBE scheme [16], we give a concrete instantiation of our generic construction over bilinear groups.

A Tweaked CCA Security for IB-ME.
As the final contribution, we introduce a slightly weak but reasonable CCA security notion, called *tweaked CCA security*, for IB-ME. Roughly, tweaked CCA security is the same as (standard) CCA security except that the (secret) encryption key used in generating challenge ciphertexts is not allowed to leaked. As an advantage of tweaked CCA security, we show that a tweaked CCA secure IB-ME scheme can be constructed solely based on a CPA secure IB-ME scheme (without strong one-time signature) by leveraging privacy and authenticity of the underlying IB-ME scheme. (That is, regarding the ciphertext size, our tweaked CCA secure IB-ME scheme does not have an overhead occurred by strong one-time signature.)

1.3 Related Work

Identity-based Matchmaking Encryption.
After seminal work [1, 21], research on various flavors is being carried out. Francati et al. [22] proposed a mismatch-cases privacy and gave a construction from the q-type assumption in the plain model. Chen et al. [23] dismantle q-type assumption and proposed first IB-ME construction from the standard assumption in the plain model. Also, from the viewpoint of matchmaking encryption that is a generalization of IB-ME, Francati et al. [24] showed that ME for general policies can be constructed from standard assumptions (LWE) and without iO/FE.

Concurrent Works.

This paper is the first paper to consider CCA security for IB-ME, and there exists a follow-up study [11] another approach to achieve CCA secure IB-ME. Also, there exists a consideration of CCA security for IB-ME [25] using CHK conversion technique. This study applies CHK technique to IBE and constructs a pairing-based CCA secure IB-ME construction.

2 Preliminaries

In this section, we provide some notations and recall definitions of some cryptographic primitives used in this paper.

2.1 Notations

In this paper, we use the following notations. For $n \in \mathbb{N}$, we denote $[n] = \{1, .., n\}$. $x \leftarrow X$ denotes the operation of sampling an element x from a finite set X. $y \leftarrow \mathsf{A}(x; r)$ denotes that a probabilistic Turing machine A outputs y for an input x using a randomness r, and we simply denote $y \leftarrow \mathsf{A}(x)$ when we need not write an internal randomness explicitly. PPT stands for probabilistic polynomial time. $x \coloneqq y$ denotes that x is defined by y. We say a function $\varepsilon(\lambda)$ is negligible in λ, if $\varepsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{Z}$, and we write $negl(\lambda)$ to denote a negligible function in λ. $\emptyset$ denotes the empty set. If O is a function or an algorithm and A is an algorithm, A^{O} means A has oracle access to O. For a bit string x, $\mathsf{len}(x)$ denotes the length of x.

2.2 Digital Signature

Let Sig denote a digital signature scheme. Sig consists of the following three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$*:* The key generation algorithm takes the security parameter 1^λ as input, and outputs a verification key vk and signing key sk.
$\mathsf{Sign}(\mathsf{sk}, \mathsf{M}) \to \Sigma$*:* The signing algorithm takes a sk and plaintext $\mathsf{M} \in \mathcal{M}$ as input, and outputs a signature Σ.
$\mathsf{Verify}(\mathsf{vk}, \Sigma) \to \top/\bot$*:* The verifying algorithm takes vk, and Σ as input, and outputs $\top$ (meaning "accept") or $\bot$ (meaning "reject").

Correctness. The correctness for Sig requires that for all $\lambda \in \mathbb{N}, (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\mathsf{M} \in \mathcal{M}$, it holds that $\top \leftarrow \mathsf{Verify}(\mathsf{vk}, \Sigma, \mathsf{M})$ with overwhelming probability after executing $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{M})$.

Security. Next, we define one-time $\mathsf{sEUF\text{-}CMA}$ security for a digital signature scheme.

Definition 1 (One-time $\mathsf{sEUF\text{-}CMA}$ Security)**.** *Let* Sig *be a digital signature scheme. We say that* Sig *satisfies one-time* $\mathsf{sEUF\text{-}CMA}$ *security if for all PPT adversaries* A*,*

$$\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{A}}(\lambda)$$

$$:= \Pr\left[((\Sigma^*, \mathsf{M}^*) \notin \mathcal{Q}_{\mathsf{O}_S}) \wedge (\mathsf{Verify}(\mathsf{vk}^*, \Sigma^*, \mathsf{M}^*) \neq \bot) \;\middle|\; \begin{array}{l} \mathcal{Q}_{\mathsf{O}_S} := \emptyset; \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda); \\ (\Sigma^*, \mathsf{M}^*) \leftarrow \mathsf{A}^{\mathsf{O}_S}(\mathsf{vk}); \end{array}\right]$$
$$= negl(\lambda)$$

holds, where signature generation oracle O_S *is implemented by* $\mathsf{Sig.Sign}(\mathsf{sk}, \cdot)$. *Also, when* A *makes a signature generation query of* (Σ, M), *the challenger adds* (Σ, M) *to the list* $\mathcal{Q}_{\mathsf{O}_S}$. A *is allowed to make only one query to* O_S.

2.3 Hierarchical Identity-based Encryption

Let l-level HIBE denote a hierarchical identity-based encryption (HIBE) scheme with a maximum depth l. l-level HIBE with a plaintext space $\mathcal{M}$ consists of the following four algorithms $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$:

$\mathsf{Setup}(1^\lambda, l) \to (\mathsf{mpk}, \mathsf{msk})$*:* The setup algorithm takes the security parameter 1^λ and the maximum hierarchical depth l as input, and outputs a master public key mpk and a master secret key msk.
$\mathsf{KeyDer}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$*:* The secret key derivation algorithm takes a mpk, the secret key of ID' $\mathsf{sk}_{\mathsf{ID}'}$, and $\mathsf{ID} \in \mathcal{ID}^{\leq l}$ as input, and outputs a secret key $\mathsf{HIBE.sk}_{\mathsf{ID}}$. The algorithm can take msk as input in place of $\mathsf{sk}_{\mathsf{ID}'}$.
$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M}) \to \mathsf{CT}$*:* The encryption algorithm takes a mpk, $\mathsf{ID} \in \mathcal{ID}^{\leq l}$, and plaintext $\mathsf{M} \in \mathcal{M}$ as input, and outputs a ciphertext CT.
$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{CT}) \to \mathsf{M}/\bot$*:* The decryption algorithm takes mpk, $\mathsf{sk}_{\mathsf{ID}}$, and CT as input, and outputs M or $\bot$.

Correctness. We define correctness for HIBE. Firstly, the correctness for KeyDer requires that for all $\lambda \in \mathbb{N}, l \in \mathbb{N}, (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l), \mathsf{ID}, \mathsf{ID}' \in \mathcal{ID}^{\leq l}$ such that $\mathsf{ID}' \in \mathtt{prefix}(\mathsf{ID})$, it holds that $\mathsf{KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}) = \mathsf{KeyDer}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID})$. Secondly, the correctness for Dec requires that for all $\lambda \in \mathbb{N}, l \in \mathbb{N}, (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l), \mathsf{M} \in \mathcal{M}$, and $\mathsf{ID} \in \mathcal{ID}^{\leq l}$, it holds that $\Pr[\mathsf{M} = \mathsf{M}'] = 1 - negl(\lambda)$, where $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{M}), \mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$, and $\mathsf{M}' \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{CT})$.

Security. Next, we define $\mathsf{IND\text{-}hID\text{-}CPA}$ security for an HIBE scheme.

Definition 2 ($\mathsf{IND\text{-}hID\text{-}CPA}$ Security)**.** *Let* HIBE *be an l-level HIBE scheme. We say that* HIBE *satisfies* $\mathsf{IND\text{-}hID\text{-}CPA}$ *security if for all PPT adversaries* A*,*

$$\mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE},\mathsf{A}}(\lambda)$$
$$:= \left| \Pr\left[\mathsf{coin} = \widehat{\mathsf{coin}} \;\middle|\; \begin{array}{l} \mathcal{Q}_{\mathsf{O}_K} := \emptyset; \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l); \\ \mathsf{coin} \leftarrow \{0,1\}; \\ (\mathsf{ID}^*, \mathsf{M}^*) \leftarrow \mathsf{A}^{\mathsf{O}_K}(\mathsf{mpk}); \\ \textit{If } (\mathsf{coin} = 0) \quad \mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}^*, \mathsf{M}^*); \\ \textit{Else} \quad \mathsf{CT}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk}); \\ \widehat{\mathsf{coin}} \leftarrow \mathsf{A}^{\mathsf{O}_K}(\mathsf{CT}^*); \end{array}\right] - \frac{1}{2} \right|$$
$$= negl(\lambda),$$

where $\mathsf{CTSamp}(\mathsf{mpk})$ *is an algorithm that outputs a uniformly random element from a ciphertext space. Secret key derivation oracle* O_K *is implemented by* $\mathsf{KeyDer}(\mathsf{mpk}, \mathsf{msk}, \cdot)$*. Also, when* A *makes a secret key derivation query* ID*, the challenger adds* ID *to* $\mathcal{Q}_{\mathsf{O}_K}$*. We require that* A *is not allowed to make a secret key derivation query* ID^* *to* O_K *and* $\mathsf{ID}' \notin \mathtt{prefix}(\mathsf{ID}^*)$ *holds for all* $\mathsf{ID}' \in \mathcal{Q}_{\mathsf{O}_K}$*.*

2.4 Hierarchical Identity-based Signature

Let l-level HIBS denote a hierarchical identity-based signature (HIBS) with a maximum depth l. l-level HIBS with a message space $\mathcal{M}$ consists of the following four algorithms $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Sign}, \mathsf{Verify})$:

$\mathsf{Setup}(1^\lambda, l) \to (\mathsf{mpk}, \mathsf{msk})$*:* The setup algorithm takes the security parameter 1^λ and the maximum hierarchical depth l as input, and outputs a master public key mpk and a master secret key msk.
$\mathsf{KeyDer}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$*:* The secret key derivation algorithm takes a mpk, the secret key of ID' $\mathsf{sk}_{\mathsf{ID}'}$, and $\mathsf{ID} \in \mathcal{ID}^{\leq l}$ as input, and outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$. The algorithm can take msk as input in place of $\mathsf{sk}_{\mathsf{ID}'}$.
$\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{M}) \to \Sigma$*:* The signing algorithm takes a mpk, $\mathsf{sk}_{\mathsf{ID}}$, and a message $\mathsf{M} \in \mathcal{M}$ as input, and outputs a signature Σ.
$\mathsf{Verify}(\mathsf{mpk}, \mathsf{ID}, \Sigma) \to \top/\bot$*:* The verifying algorithm takes mpk, $\mathsf{ID} \in \mathcal{ID}^{\leq l}$, and Σ as input, and outputs $\top$ (meaning "accept") or $\bot$ (meaning "reject").

Correctness. We define correctness for HIBS. Firstly, the correctness for KeyDer requires that for all $\lambda \in \mathbb{N}, l \in \mathbb{N}, (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l), \mathsf{ID}, \mathsf{ID}' \in \mathcal{ID}^{\leq l}$ such that $\mathsf{ID}' \in \mathtt{prefix}(\mathsf{ID})$, $\mathsf{KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}) = \mathsf{KeyDer}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID})$ holds. Secondly, the correctness for Verify requires that for all $\lambda \in \mathbb{N}, l \in \mathbb{N}, (\mathsf{HIBS.mpk}, \mathsf{HIBS.msk}) \leftarrow \mathsf{Setup}(1^\lambda, l), \mathsf{M} \in \mathcal{M}$, and $\mathsf{ID} \in \mathcal{ID}^{\leq l}$, it holds that $\Pr[\mathsf{Verify}(\mathsf{mpk}, \mathsf{ID}, \Sigma) = \top] = 1 - negl(\lambda)$, where $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$ and $\Sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{M})$.

Security. Next, we define $\mathsf{EUF\text{-}hID\text{-}CMA}$ security for an HIBS scheme.

Definition 3 ($\mathsf{EUF\text{-}hID\text{-}CMA}$ Security)**.** *Let* HIBS *be an* l*-level HIBS scheme. We say that* HIBS *satisfies* $\mathsf{EUF\text{-}hID\text{-}CMA}$ *security if for all PPT adversaries* A*,*

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{EUF\text{-}hID\text{-}CMA}}_{\mathsf{HIBS},\mathsf{A}}(\lambda) \\
&:= \Pr\left[\begin{array}{c|l}
(\forall \mathsf{ID}' \in \mathcal{Q}_{\mathsf{O}_K} : \mathsf{ID}' \notin \mathtt{prefix}(\mathsf{ID}^*))\wedge & \mathcal{Q}_{\mathsf{O}_K}, \mathcal{Q}_{\mathsf{O}_S} := \emptyset; \\
((\mathsf{ID}^*, \mathsf{M}^*) \notin \mathcal{Q}_{\mathsf{O}_S})\wedge & (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l); \\
(\mathsf{Verify}(\mathsf{mpk}, \mathsf{ID}^*, \Sigma^*) = \top) & (\mathsf{ID}^*, \Sigma^*, \mathsf{M}^*) \leftarrow \mathsf{A}^{\mathsf{O}_K, \mathsf{O}_S}(\mathsf{mpk});
\end{array}\right] \\
&= negl(\lambda)
\end{aligned}
$$

holds, where signing key derivation oracle O_K *is implemented by* $\mathsf{KeyDer}(\mathsf{mpk}, \mathsf{msk}, \cdot)$*. Also, when* A *makes a signing key derivation query* ID*, the challenger adds* ID *to* $\mathcal{Q}_{\mathsf{O}_K}$*. Signature generation oracle* O_S *is implemented by* $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \cdot)$ *where* $\mathsf{sk}_{\mathsf{ID}} = \mathsf{KeyDer}(\mathsf{mpk}, \mathsf{msk}, \cdot)$*. Also, when* A *makes a signature generation query* $(\mathsf{ID}, \mathsf{M})$*, the challenger adds* $(\mathsf{ID}, \mathsf{M})$ *to* $\mathcal{Q}_{\mathsf{O}_S}$*.*

2.5 Affine Message Authentication

Affine message authentication codes (MACs) over $\mathbb{Z}_q^n$ are group-based MACs with a specific algebraic structure.

Definition 4 (Affine MACs)**.** *Let par be system parameters containing a group* $\mathbb{G} = (\mathbb{G}_2, q, g_2)$ *of prime-order* q *and let* $n \in \mathbb{N}$*. We say that* $\mathsf{MAC} = (\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Verify})$ *is affine over* $\mathbb{Z}_q^n$ *if the following conditions hold:*

- $\mathsf{KeyGen}(1^\lambda)$ *returns* $\mathsf{sk}_{\mathsf{MAC}}$ *containing* $(\mathbf{B}, \mathbf{x}_0, ..., \mathbf{x}_l, \mathbf{x}'_0, ..., \mathbf{x}'_{l'})$*, where* $\mathbf{B} \in \mathbb{Z}_q^{n \times n'}$*,* $\mathbf{x}_i \in \mathbb{Z}_q^n$*,* $\mathbf{x}'_j \in \mathbb{Z}_q$*, for some* $n', l, l' \in \mathbb{N}$*. We assume* $\mathbf{B}$ *has rank at least one.*
- $\mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{M} \in \mathbf{B})$ *returns a tag* $\tau = ([\mathbf{t}]_2, [u]_2) \in \mathbb{G}_2^n \times \mathbb{G}_2$*, computed as*

$$\mathbf{t} = \mathbf{B}\mathbf{s} \in \mathbb{Z}_q^n \quad \textit{for } \mathbf{s} \leftarrow \mathbb{Z}_q^{n'}$$

$$u = \sum_{i=0}^{l} f_i(\mathsf{M})\mathbf{x}_i^\top \mathbf{t} + \sum_{i=0}^{l'} f'_i(\mathsf{M})x'_i \in \mathbb{Z}_q \tag{1}$$

  for some public defining functions $f_i : \mathsf{M} \to \mathbb{Z}_q$ *and* $f'_i : \mathsf{M} \to \mathbb{Z}_q$*. Vector* $\mathbf{s}$ *can be generated either pseudorandomly or randomly and* u *is the (deterministic).*
- $\mathsf{Verify}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{M}, \tau = ([\mathbf{t}]_2, [u]_2))$ *verifies if eq. (1) holds.*

Next, we recall pseudorandomness against chosen-message attacks (PR-CMA) for affine MACs.

Definition 5 (PR-CMA)**.** *An affine* MAC *over* $\mathbb{Z}_q^n$ *is* $\mathsf{PR\text{-}CMA}$ *secure if for all PPT* A*,*

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{PR\text{-}CMA}}_{\mathsf{A},\mathsf{MAC}}(\lambda)\\
&:= \left| \Pr\left[\mathsf{coin} = \widehat{\mathsf{coin}} \;\middle|\; \begin{array}{l} \mathcal{Q}_{\mathsf{O}_M} := \emptyset; \\ \mathsf{sk}_{\mathsf{MAC}} \leftarrow \mathsf{KeyGen}(1^\lambda); \\ \mathsf{coin} \leftarrow \{0,1\}; \\ \mathsf{M}^* \leftarrow \mathsf{A}^{\mathsf{O}_M}(1^\lambda); \\ \textit{If } (\mathsf{coin} = 0) \quad \tau^* \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{M}^*); \\ \textit{Else} \quad \tau^* \leftarrow \mathsf{Rand}(1^\lambda); \\ \widehat{\mathsf{coin}} \leftarrow \mathsf{A}^{\mathsf{O}_M}(\tau^*); \end{array} \right] - \frac{1}{2} \right|\\
&= negl(\lambda).
\end{aligned}$$

3 Hierarchical Identity-based Matchmaking Encryption

In this section, we introduce a new cryptographic primitive called *hierarchical identity-based matchmaking encryption* (HIB-ME). In Section 3.1, we provide the formalization of HIB-ME. Then, in Section 3.2, we give a generic construction of HIB-ME based on HIBE and HIBS. Finally, in Section 3.3, we provide the security proofs for our HIB-ME scheme.

3.1 Formalization of HIB-ME

In this section, we provide the syntax, correctness, and security definitions for HIB-ME. Informally, HIB-ME is an extension of IB-ME in the sense that it enables senders (resp., receivers) to generate encryption keys (resp., decryption keys) for their children's identities.

Let (k, l)-level $\mathsf{HIB\text{-}ME}$ denote an HIB-ME scheme with a maximum depth k for sender keys and depth l for receiver keys. (k, l)-level $\mathsf{HIB\text{-}ME}$ consists of the following five algorithms $(\mathsf{Setup}, \mathsf{SKDer}, \mathsf{RKDer}, \mathsf{Enc}, \mathsf{Dec})$:

$\mathsf{Setup}(1^\lambda, k, l) \to (\mathsf{mpk}, \mathsf{msk})$*:* The setup algorithm takes the security parameter 1^λ and the maximum hierarchical sender depth k and receiver depth l as input, and outputs a master public key mpk and a master secret key msk.

$\mathsf{SKDer}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma) \to \mathsf{ek}_\sigma$*:* The sender key derivation algorithm takes a master public key mpk, an encryption key of σ' $\mathsf{ek}_{\sigma'}$, and a sender's identity $\sigma \in \mathcal{ID}^{\leq l}$ as input, and outputs an encryption key ek_σ. The algorithm can take msk as input in place of $\mathsf{ek}_{\sigma'}$.

$\mathsf{RKDer}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho) \to \mathsf{dk}_\rho$*:* The receiver key derivation algorithm takes a master public key mpk, a decryption key of ρ' $\mathsf{dk}_{\rho'}$, and a receiver's identity $\rho \in \mathcal{ID}^{\leq l}$ as input, and outputs a decryption key dk_ρ. The algorithm can take msk as input in place of $\mathsf{dk}_{\rho'}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{M}) \to \mathsf{CT}$*:* The encryption algorithm takes a master public key mpk, an encryption key ek_σ, a receiver's identity $\mathsf{rcv} \in \mathcal{ID}^{\leq l}$, and a plaintext $\mathsf{M} \in \mathcal{M}$ as input, and outputs a ciphertext CT.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{CT}) \to \mathsf{M}/\perp$*:* The decryption algorithm takes a master public key mpk, a decryption key $\mathsf{dk}_{\mathsf{ID}}$, a sender's identity $\mathsf{snd} \in \mathcal{ID}^{\leq l}$, and a ciphertext CT as input, and outputs a plaintext M or $\perp$.

Correctness. We define correctness for $\mathsf{HIB\text{-}ME}$. Firstly, the correctness for SKDer requires that for all $\lambda \in \mathbb{N}, l \in \mathbb{N}, (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l), \sigma, \sigma' \in \mathcal{ID}^{\leq l}$ such that $\sigma' \in \mathtt{prefix}(\sigma)$, $\mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \sigma) = \mathsf{SKDer}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$ holds. Secondly, the correctness for RKDer requires that for all $\lambda \in \mathbb{N}, l \in \mathbb{N}, (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l), \rho, \rho' \in \mathcal{ID}^{\leq l}$ such that $\rho' \in \mathtt{prefix}(\rho)$, $\mathsf{RKDer}(\mathsf{mpk}, \mathsf{msk}, \rho) = \mathsf{RKDer}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$ holds. Thirdly, the correctness for Dec requires that for all $\lambda \in \mathbb{N}, l \in \mathbb{N}, (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l), \mathsf{M} \in \mathcal{M}$, and $\sigma, \rho \in \mathcal{ID}^{\leq l}$, it holds that $\Pr[\mathsf{M} = \mathsf{M}'] = 1 - negl(\lambda)$ where $\mathsf{ek}_\sigma \leftarrow \mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \sigma)$, $\mathsf{dk}_\rho \leftarrow \mathsf{RKDer}(\mathsf{mpk}, \mathsf{msk}, \rho)$, $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv} = \rho, \mathsf{M})$, and $\mathsf{M}' \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd} = \sigma, \mathsf{CT})$.

Security. Next, we define security requirements ($\mathsf{hib\text{-}cca\text{-}priv}$ security, $\mathsf{hib\text{-}cpa\text{-}priv}$ security, and $\mathsf{hib\text{-}auth}$ security) for HIB-ME.

Definition 6 (Security of HIB-ME)**.** *Let* $\mathsf{HIB\text{-}ME}$ *be an HIB-ME scheme. We say that* $\mathsf{HIB\text{-}ME}$ *satisfies:*

- $\mathsf{hib\text{-}cca\text{-}priv}$ *security if for all PPT adversaries* A *in* $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME,A}}(\lambda)$ *(which is defined as in Fig. 1 (left)),* $\mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME,A}}(\lambda) := |\Pr[\mathsf{coin} = \widehat{\mathsf{coin}}] - 1/2| = negl(\lambda)$ *holds.*
- $\mathsf{hib\text{-}cpa\text{-}priv}$ *security if for all PPT adversaries* A *in* $\mathsf{Exp}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME,A}}(\lambda)$ *(this is a variant of* $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME,A}}(\lambda)$ *where there is no decryption oracle),* $\mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME,A}}(\lambda) := |\Pr[\mathsf{coin} = \widehat{\mathsf{coin}}] - 1/2| = negl(\lambda)$ *holds.*

$\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{(k,l)\text{-}\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$	$\mathsf{Exp}^{\mathsf{hib\text{-}auth}}_{(k,l)\text{-}\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$
$\mathcal{Q}_{\mathsf{O}_R}, \mathcal{Q}_{\mathsf{O}_D} := \emptyset$	$\mathcal{Q}_{\mathsf{O}_S}, \mathcal{Q}_{\mathsf{O}_E} := \emptyset$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, k, l)$	$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, k, l)$
$\mathsf{coin} \leftarrow \{0,1\}$	$(\mathsf{CT}^*, \rho^*, \mathsf{snd}^*) \leftarrow \mathsf{A}^{\mathsf{O}_S,\mathsf{O}_R,\mathsf{O}_E}(\mathsf{mpk})$
$(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*) \leftarrow \mathsf{A}^{\mathsf{O}_S,\mathsf{O}_R,\mathsf{O}_D}(\mathsf{mpk})$	$\mathsf{dk}_{\rho^*} \leftarrow \mathsf{RKDer}(\mathsf{mpk}, \mathsf{msk}, \rho^*)$
$\mathsf{ek}_{\sigma^*} \leftarrow \mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \sigma^*)$	$\mathsf{M}^* \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{\rho^*}, \mathsf{snd}^*, \mathsf{CT}^*)$
If $(\mathsf{coin} = 0)$	If $(\forall \sigma \in \mathcal{Q}_{\mathsf{O}_S} : \sigma \neq \mathsf{snd}^*) \wedge (\forall \sigma' \in \mathcal{Q}_{\mathsf{O}_S} : \sigma' \notin$
$\quad \mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_{\sigma^*}, \mathsf{rcv}^*, \mathsf{M}^*)$	$\mathtt{prefix}(\sigma^*)) \wedge ((\mathsf{snd}^*, \rho^*, \mathsf{M}^*) \notin \mathcal{Q}_E) \wedge (\mathsf{M}^* \neq \bot)$
Else $\quad \mathsf{CT}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$	$\quad$ Return 1
$\widehat{\mathsf{coin}} \leftarrow \mathsf{A}^{\mathsf{O}_S,\mathsf{O}_R,\mathsf{O}_D}(\mathsf{CT}^*)$	Else $\quad$ Return 0

Fig. 1 Security experiments for HIB-ME: CCA privacy experiment and authenticity experiment. Let $\mathsf{CTSamp}(\mathsf{mpk})$ be an algorithm that outputs a uniformly random element from a ciphertext space. Sender key derivation oracle O_S is implemented by $\mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \cdot)$. Also, when the adversary makes a sender key derivation query about σ, the challenger adds σ to $\mathcal{Q}_{\mathsf{O}_S}$. Receiver key derivation oracle O_R is implemented by $\mathsf{RKDer}(\mathsf{mpk}, \mathsf{msk}, \cdot)$. Also, when the adversary makes a receiver key derivation query about ρ, the challenger adds ρ to $\mathcal{Q}_{\mathsf{O}_R}$. Encryption oracle O_E is implemented by $\mathsf{HIB\text{-}ME.Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \cdot, \cdot)$, where $\mathsf{ek}_\sigma \leftarrow \mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \cdot)$. Also, when the adversary makes encryption query about $(\sigma, \mathsf{rcv}, \mathsf{M})$, the challenger adds $(\sigma, \mathsf{rcv}, \mathsf{M})$ to $\mathcal{Q}_{\mathsf{O}_E}$ Decryption oracle O_D is implemented by $\mathsf{HIB\text{-}ME.Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \cdot, \cdot)$, where $\mathsf{dk}_\rho \leftarrow \mathsf{RKDer}(\mathsf{mpk}, \mathsf{msk}, \cdot)$. Also, when the adversary makes decryption query about $(\rho, \mathsf{snd}, \mathsf{CT})$, the challenger adds $(\rho, \mathsf{snd}, \mathsf{CT})$ to $\mathcal{Q}_{\mathsf{O}_D}$. In $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{(k,l)\text{-}\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$, we require that A is not allowed to make a receiver key derivation query ρ^* to O_R, $\rho' \notin \mathtt{prefix}(\rho^*)$ holds for all $\forall \rho' \in \mathcal{Q}_{\mathsf{O}_R}$, and $(\rho^*, \mathsf{snd}^*, \mathsf{CT}^*) \notin \mathcal{Q}_{\mathsf{O}_D}$ holds.

- $\mathsf{hib\text{-}auth}$ *security if for all PPT adversaries* A, $\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = 1] = negl(\lambda)$, *where* $\mathsf{Exp}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ *is defined as in Fig. 1 (right).*

3.2 Generic Construction of HIB-ME

In this section, we provide a generic construction of HIB-ME. One might see that this construction is an extension of the construction by Wang et al. [26] Specifically, we show that by using two $l+1$-level HIBE schemes and a k-level HIBS scheme, we can construct a HIB-ME scheme with k-level sender hierarchies and l-level receiver hierarchies.

Construction. We give a formal description of our HIB-ME scheme with a message space $\mathcal{M}$ and an identity space $\mathcal{ID}$.

- Let HIBS be an k-level HIBS scheme with an identity space $\mathcal{ID}$, a message space $\mathcal{ID}|\mathcal{M}$, and a signature space $\mathcal{S}$.
- Let HIBE^1 be an $l+1$-level HIBE scheme with a message space $\mathcal{M}$ and an identity space $\mathcal{ID}|\mathcal{ID}$.
- Let HIBE^2 be an $l+1$-level HIBE scheme with a message space $\mathcal{S}$ and an identity space $\mathcal{ID}|\mathcal{ID}$.

Based on the above primitives, we construct our (k,l)-level HIB-ME scheme as follows:

$\mathsf{Setup}(1^\lambda, k, l)$*:* On input a security parameter 1^λ, a maximum hierarchical depth k for senders, and a maximum hierarchical depth l for receivers, the setup algorithm runs $(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{msk}) \leftarrow \mathsf{HIBE}^1.\mathsf{Setup}(1^\lambda, l+1)$, $(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{msk}) \leftarrow \mathsf{HIBE}^2.\mathsf{Setup}(1^\lambda, l+1)$, $(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{msk}) \leftarrow \mathsf{HIBS}.\mathsf{Setup}(1^\lambda, k)$, and outputs $\mathsf{mpk} = (\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{mpk})$ and $\mathsf{msk} = (\mathsf{HIBE}^1.\mathsf{msk}, \mathsf{HIBE}^2.\mathsf{msk}, \mathsf{HIBS}.\mathsf{msk})$.

$\mathsf{SKDer}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$*:* On input a master public key mpk, an encryption key of σ' $\mathsf{ek}_{\sigma'}$, and an identity σ, the sender key derivation algorithm computes a signing key $\mathsf{HIBS}.\mathsf{sk}_\sigma \leftarrow \mathsf{HIBS}.\mathsf{KeyDer}(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{sk}_{\sigma'}, \sigma)$ and outputs $\mathsf{ek}_\sigma = \mathsf{HIBS}.\mathsf{sk}_\sigma$

$\mathsf{RKDer}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$*:* On input a master public key mpk, a decryption key of ρ' $\mathsf{dk}_{\rho'}$ and an identity ρ, the receiver key derivation algorithm computes $\mathsf{HIBE}^1.\mathsf{sk}_\rho \leftarrow \mathsf{HIBE}^1.\mathsf{KeyDer}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{sk}_{\rho'}, \rho)$ and $\mathsf{HIBE}^2.\mathsf{sk}_\rho \leftarrow \mathsf{HIBE}^2.\mathsf{KeyDer}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{sk}_{\rho'}, \rho)$ and outputs $\mathsf{dk}_\rho = (\mathsf{HIBE}^1.\mathsf{sk}_\rho, \mathsf{HIBE}^2.\mathsf{sk}_\rho)$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{M})$*:* On input a master public key mpk, a secret encryption key ek_σ, a target identity rcv, and a message $\mathsf{M} \in \mathcal{M}$, the encryption algorithm firstly generates a signature $\Sigma \leftarrow \mathsf{HIBS}.\mathsf{Sign}(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{sk}_\sigma, \mathsf{rcv}|\mathsf{M})$. It then computes ciphertexts (for HIBE^1 and HIBE^2) $\mathsf{CT}^1 \leftarrow \mathsf{HIBE}^1.\mathsf{Enc}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{rcv}|\sigma, \mathsf{M})$ and $\mathsf{CT}^2 \leftarrow \mathsf{HIBE}^2.\mathsf{Enc}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{rcv}|\sigma, \Sigma)$ under an identity $\mathsf{rcv}|\sigma$. Finally, it outputs ciphertext $\mathsf{CT} = (\mathsf{CT}^1, \mathsf{CT}^2)$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{CT})$*:* On input a master public key mpk, a secret decryption key dk_ρ, a selected sender's identity snd, and a ciphertext CT, the decryption algorithm firstly delegates a decryption key $\mathsf{dk}^2_{\rho|\mathsf{snd}} \leftarrow \mathsf{HIBE}^2.\mathsf{KeyDer}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{sk}_\rho, \rho|\mathsf{snd})$ and recovers $\Sigma \leftarrow \mathsf{HIBE}^2.\mathsf{Dec}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{dk}^2_{\rho|\mathsf{snd}}, \mathsf{CT}^2)$. Then, if $\mathsf{HIBS}.\mathsf{Verify}(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{snd}, \rho|\mathsf{M}, \Sigma) = \perp$ holds, it returns $\perp$. Otherwise, it delegates a decryption key $\mathsf{dk}^1_{\rho|\mathsf{snd}} \leftarrow \mathsf{HIBE}^1.\mathsf{KeyDer}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{sk}_\rho, \rho|\mathsf{snd})$ and recovers a message $\mathsf{M} \leftarrow \mathsf{HIBE}^1.\mathsf{Dec}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{dk}^1_{\rho|\mathsf{snd}}, \mathsf{CT}^1)$.

Correctness. Here, we show that our HIB-ME scheme in section 3.2 satisfies correctness. Firstly, the correctness of SKDer is derived by the correctness of $\mathsf{HIBS}.\mathsf{KeyDer}$ Secondly, the correctness of RKDer is derived by the correctness of $\mathsf{HIBE}^1.\mathsf{KeyDer}$ and $\mathsf{HIBE}^2.\mathsf{KeyDer}$. Thirdly, the correctness of Dec is follows: Fix a message $\mathsf{M} \in \mathcal{M}$, a sender's identity $\sigma \in \mathcal{ID}$, and a target receiver's identity $\mathsf{rcv} \in \mathcal{ID}$. Moreover, fix a receiver's identity $\rho \in \mathcal{ID}$, and a target sender's identity $\mathsf{snd} \in \mathcal{ID}$. Assumes that $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{ek}_\sigma \leftarrow \mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \sigma)$, $\mathsf{dk}_\rho \leftarrow \mathsf{RKDer}(\mathsf{mpk}, \mathsf{msk}, \rho)$, and $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{M})$. In this case, we have $\mathsf{mpk} = (\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{mpk})$, $\mathsf{msk} = (\mathsf{HIBE}^1.\mathsf{msk}, \mathsf{HIBE}^2.\mathsf{msk}, \mathsf{HIBS}.\mathsf{msk})$, ek_σ is the signing key output by $\mathsf{HIBS}.\mathsf{KeyDer}(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{msk}, \sigma)$, dk_ρ is the tuple of the decryption keys obtained by $\mathsf{HIBE}^1.\mathsf{sk}_\rho \leftarrow \mathsf{HIBE}^1.\mathsf{KeyDer}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{msk}, \rho)$ and $\mathsf{HIBE}^2.\mathsf{sk}_\rho \leftarrow \mathsf{HIBE}^2.\mathsf{KeyDer}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{msk}, \rho)$, and $\mathsf{CT} = (\mathsf{CT}^1, \mathsf{CT}^2)$ is the tuple of ciphertexts $\mathsf{CT}^1 \leftarrow \mathsf{HIBE}^1.\mathsf{Enc}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{rcv}|\sigma, \mathsf{M})$ and $\mathsf{CT}^2 \leftarrow \mathsf{HIBE}^2.\mathsf{Enc}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{rcv}|\sigma, \Sigma)$, where $\Sigma \leftarrow \mathsf{HIBS}.\mathsf{Sign}(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{sk}_\sigma, \mathsf{rcv}|\mathsf{M})$. It is obvious that, when $\rho = \mathsf{rcv} \wedge \sigma = \mathsf{snd}$, $\mathsf{rcv}|\sigma = \rho|\mathsf{snd}$ holds. Then, the receiver with the identity ρ can generate a decryption key for $\rho|\mathsf{snd}$ using its key dk_ρ and recovers the signature Σ and the message M correctly due to the correctness of HIBE^1 and HIBE^2. Moreover, since the signature Σ is computed by the sender with an identity σ for $\mathsf{rcv}|\mathsf{M}$, we have $\top \leftarrow \mathsf{HIBS}.\mathsf{Verify}(\mathsf{HIBS}.\mathsf{mpk}, \sigma, \Sigma, \rho|\mathsf{M})$ with overwhelming probability due to the correctness of HIBS. Thus, the correctness of Dec holds.

3.3 Security Proofs

In this section, we show that our scheme satisfies security requirements.

Theorem 1. *If* $l+1$*-level* HIBE^1 *and* $l+1$*-level* HIBE^2 *are* $\mathsf{IND\text{-}hID\text{-}CPA}$ *secure, then our HIB-ME scheme* $\mathsf{HIB\text{-}ME}$ *satisfies* $\mathsf{hib\text{-}cpa\text{-}priv}$.

Proof. Let $\star.\mathsf{CTSamp}(\star.\mathsf{mpk})$ be an algorithm that outputs a random element from ciphertext space of $\star$ scheme and A a PPT adversary against the $\mathsf{hib\text{-}cpa\text{-}priv}$ security of $\mathsf{HIB\text{-}ME}$.

Game_0*:* This is an original experiment $\mathsf{Exp}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ conditioned on $\mathsf{coin} = 0$. Namely, the challenger responds to the challenge query $(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*)$ with the ciphertext $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_{\sigma^*}, \mathsf{rcv}^*, \mathsf{M}^*)$, where $\mathsf{ek}_{\sigma^*} \leftarrow \mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \sigma^*)$.

Game_1*:* Same as Game_0, except that the challenger generates the challenge ciphertext as $\mathsf{CT}^* \leftarrow (\mathsf{CT}^1, \mathsf{CT}^\star)$, where the first part CT^1 is the same as in Game_0, but the second part is generated as $\mathsf{CT}^\star \leftarrow \mathsf{HIBE}^2.\mathsf{CTSamp}(\mathsf{HIBE}^2.\mathsf{mpk})$.

Game_2*:* Same as Game_1, except that the challenger also generates the first part as $\mathsf{CT}^1 \leftarrow \mathsf{HIBE}^1.\mathsf{CTSamp}(\mathsf{HIBE}^1.\mathsf{mpk})$. Here, the challenge ciphertext generated by the challenger in Game_2 is a tuple of random elements in the ciphertext space. That is, Game_2 is exactly the same as an original experiment $\mathsf{Exp}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ conditioned on $\mathsf{coin} = 1$.

Let X_i denote an event that $\mathsf{coin} = \widehat{\mathsf{coin}}$ holds in Game_i for $i \in \{0,1,2\}$. Then, we can estimate the advantage $\mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = 2\cdot|\Pr[\mathsf{coin} = \widehat{\mathsf{coin}}] - \frac{1}{2}| = |\Pr[X_0] - \Pr[X_2]| \leq |\Pr[X_0] - \Pr[X_1]| + |\Pr[X_1] - \Pr[X_2]|$.

Lemma 1. *There exists an adversary* B^2 *against the* $\mathsf{IND\text{-}hID\text{-}CPA}$ *security of* HIBE^2 *such that* $|\Pr[X_0] - \Pr[X_1]| = \mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE}^2,\mathsf{B}^2}(\lambda)$.

Proof. We construct a PPT adversary B^2 who attacks the $\mathsf{IND\text{-}hID\text{-}CPA}$ security of HIBE^2 so that $|\Pr[X_0] - \Pr[X_1]| = \mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE}^2,\mathsf{B}^2}(\lambda)$, using the adversary A.

1. At the beginning of the $\mathsf{IND\text{-}hID\text{-}CPA}$ security game of HIBE^2, B^2 firstly receives the public parameter $\mathsf{HIBE}^2.\mathsf{mpk}$ from its challenger. Then, B^2 picks $\mathsf{coin}' \leftarrow \{0,1\}$ and generates $(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{msk}) \leftarrow \mathsf{HIBE}^1.\mathsf{Setup}(1^\lambda, l+1)$ and $(\mathsf{HIBS.mpk}, \mathsf{HIBS.msk}) \leftarrow \mathsf{HIBS.Setup}(1^\lambda, k)$. Then, B^2 sets $\mathsf{mpk} := (\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBS.mpk})$ and sends it to A.
2. Whenever A makes a decryption key query ρ, B^2 makes a key generation query ρ to its challenger to obtain a key $\mathsf{HIBE}^2.\mathsf{sk}_\rho$ and generates $\mathsf{HIBE}^1.\mathsf{sk}\rho \leftarrow \mathsf{HIBE}^1.\mathsf{KeyDer}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{msk}, \rho)$ by itself. It then sets $\mathsf{dk}_\rho := (\mathsf{HIBE}^1.\mathsf{sk}_\rho, \mathsf{HIBE}^2.\mathsf{sk}_\rho)$ and returns dk_ρ to A.
3. Whenever A makes an encryption key query σ, B^2 generates $\mathsf{HIBS.sk}_\sigma \leftarrow \mathsf{HIBS.KeyDer}(\mathsf{HIBS.mpk}, \mathsf{HIBS.msk}, \sigma)$, sets $\mathsf{ek}_\sigma := \mathsf{HIBS.sk}_\sigma$, and returns ek_σ to A.
4. Whenever A makes a challenge query of $(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*)$, B^2 firstly computes $\mathsf{HIBS.sk}_{\sigma^*} \leftarrow \mathsf{HIBS.KeyDer}(\mathsf{HIBS.mpk}, \mathsf{HIBS.msk}, \sigma^*)$, $\Sigma \leftarrow \mathsf{HIBS.Sign}(\mathsf{HIBS.mpk}, \mathsf{HIBS.sk}_{\sigma^*}, \mathsf{rcv}^*|\mathsf{M}^*)$, and $\mathsf{CT}^{1*} \leftarrow \mathsf{HIBE}^1.\mathsf{Enc}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{rcv}^*|\sigma^*, \mathsf{M}^*)$. Then, B^2 makes a challenge query of

$(\mathsf{rcv}^*|\mathsf{M}^*, \Sigma)$ to its challenger. Upon receiving a challenge ciphertext $\mathsf{CT}^\star$, B^2 sets the challenge ciphertext as $\mathsf{CT}^* := (\mathsf{CT}^{1*}, \mathsf{CT}^\star)$ and sends it to A.

5. Finally, when A outputs a guess $\widehat{\mathsf{coin}}'$, B^2 checks whether $\mathsf{coin}' = \widehat{\mathsf{coin}}'$ holds. If this is the case, then B^2 outputs $\widehat{\mathsf{coin}} := 1$ to its challenger. Otherwise, B^2 outputs $\widehat{\mathsf{coin}} := 0$ to its challenger.

We firstly argue that B^2 is admissible for the $\mathsf{IND\text{-}hID\text{-}CPA}$ security game of HIBE^2. Since A is admissible for the $\mathsf{hib\text{-}cpa\text{-}priv}$ game, this means that, for all decryption key generation queries $\rho \in \mathcal{ID}$ made by A, it must satisfy that $\rho \neq \mathsf{rcv}^*$ holds. Since the challenge query submitted by B^2 forms of $(\mathsf{rcv}^*|\sigma^*, \Sigma)$ and all key generation queries it issued are exactly identities ρ queried by A, this means that B^2 never asked a secret key for the challenge identity rcv^* or its prefix. Thus, B^2 is admissible for the $\mathsf{IND\text{-}hID\text{-}CPA}$ security game of HIBE^2. Here, let coin be the challenge bit of the $\mathsf{IND\text{-}hID\text{-}CPA}$ security game of HIBE^2. Due to the above construction, if $\mathsf{coin} = 0$ (that is, $\mathsf{CT}^\star \leftarrow \mathsf{HIBE}^2.\mathsf{Enc}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{rcv}^*|\sigma^*, \Sigma)$) holds, B^2 perfectly simulates $\eth_0$ for A. Then, $\Pr[X_0] = \Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 0]$ holds. If $\mathsf{coin} = 1$ (that is, $\mathsf{CT}^\star \leftarrow \mathsf{HIBE}^2.\mathsf{CTSamp}(\mathsf{HIBE}^2.\mathsf{mpk})$) holds, then B^2 perfectly simulates Game_1 for A. Then, $\Pr[X_1] = \Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 1]$ holds. Now, we have

$$\begin{aligned}|\Pr[X_0] - \Pr[X_1]| &= |\Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 0] - \Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 1]| \\ &= \mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE}^2,\mathsf{B}^2}(\lambda).\end{aligned}$$

$\square$

Lemma 2. *There exists an adversary* B^1 *against the* $\mathsf{IND\text{-}hID\text{-}CPA}$ *security of* HIBE^1 *such that* $|\Pr[X_1] - \Pr[X_2]| = \mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE}^1,\mathsf{B}^1}(\lambda)$.

The proof is similar to one of Lemma 1.

Proof. We use A to construct an adversary B^1 for the $\mathsf{IND\text{-}hID\text{-}CPA}$ security of HIBE^1.

1. At the beginning of the $\mathsf{IND\text{-}hID\text{-}CPA}$ security game, adversary B^1 receives the public parameters $\mathsf{HIBE}^1.\mathsf{mpk}$ from the $\mathsf{IND\text{-}hID\text{-}CPA}$ security challenger. Then, B^1 picks $\mathsf{coin}' \leftarrow \{0,1\}$. In addition, it runs $(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{msk}) \leftarrow \mathsf{HIBE}^2.\mathsf{Setup}(1^\lambda, l+1)$ and $(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{msk}) := \mathsf{HIBS}.\mathsf{Setup}(1^\lambda, k)$. Then, B^1 sets $\mathsf{mpk} := (\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{mpk})$ and sends it to the adversary A.
2. Whenever A makes a decryption key query on an identity ρ, B^1 makes a key generation query to $\mathsf{IND\text{-}hID\text{-}CPA}$ challenger on ρ to obtain a key $\mathsf{HIBE}^1.\mathsf{sk}_\rho$, and generates $\mathsf{HIBE}^2.\mathsf{sk}_\rho \leftarrow \mathsf{HIBE}^2.\mathsf{KeyDer}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{msk}, \rho)$ by itself. It then sets $\mathsf{dk}_\rho := (\mathsf{HIBE}^1.\mathsf{sk}_\rho, \mathsf{HIBE}^2.\mathsf{sk}_\rho)$ and returns dk_ρ to A.
3. Whenever A makes an encryption key query on an identity σ, B^1 computes $\mathsf{HIBS}.\mathsf{sk}_\sigma \leftarrow \mathsf{HIBS}.\mathsf{KeyDer}(\mathsf{HIBS}.\mathsf{mpk}, \mathsf{HIBS}.\mathsf{msk}, \sigma)$. It sets $\mathsf{ek}_\sigma = \mathsf{HIBS}.\mathsf{sk}_\sigma$ and returns ek_σ to A.
4. Whenever A makes a challenge query on input $(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*)$, B^1 first computes random $\mathsf{CT}^{2*} \leftarrow \mathsf{HIB\text{-}ME}^2.\mathsf{CTSamp}(\mathsf{HIBE}^2.\mathsf{mpk})$. Then it submits the challenge query $(\mathsf{rcv}^*|\sigma^*, \mathsf{M}^*)$ to the $\mathsf{IND\text{-}hID\text{-}CPA}$ challenger. The challenger replies to B^1 with $\mathsf{CT}^\star$. Then, the algorithm B^1 sets the challenge ciphertext as $\mathsf{CT} := (\mathsf{CT}^\star, \mathsf{CT}^{2*})$ and sends it to A.

5. Finally, when A outputs guess $\widehat{\mathsf{coin}'}$, B^1 checks $\mathsf{coin}' = \widehat{\mathsf{coin}'}$. If so, B^2 outputs $\widehat{\mathsf{coin}} := 1$. Otherwise, B^2 outputs $\widehat{\mathsf{coin}} := 0$.

Similarly to Lemma 1, B^1 is admissible for the $\mathsf{IND\text{-}hID\text{-}CPA}$ game. By above construction, if $\mathsf{CT}^{1*} \leftarrow \mathsf{HIBE}^1.\mathsf{Enc}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{rcv}^*|\sigma^*, \mathsf{M}^*)$, B^1 perfectly simulated $\eth_1$ for A, and then $\Pr[X_1] = \Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 0]$ holds. If $\mathsf{CT}^{1*} \leftarrow \mathsf{HIBE}^1.\mathsf{CTSamp}(\mathsf{HIBE}^1.\mathsf{mpk})$, then B^1 perfectly simulated Game_2 for A, and then $\Pr[X_2] = \Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 1]$ holds. Now, we have

$$
\begin{aligned}
|\Pr[X_1] - \Pr[X_2]| &= |\Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 0] - \Pr[\widehat{\mathsf{coin}} = 1|\mathsf{coin} = 1]| \\
&= \mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE}^1,\mathsf{B}^1}(\lambda).
\end{aligned}
$$

□

Combining everything together, we conclude

$$
\mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE}^1,\mathsf{B}^1}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}hID\text{-}CPA}}_{\mathsf{HIBE}^2,\mathsf{B}^2}(\lambda).
$$

Now, since both HIBE^1 and HIBE^2 satisfy $\mathsf{IND\text{-}hID\text{-}CPA}$ security, $\mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = negl(\lambda)$ holds. This concludes that $\mathsf{HIB\text{-}ME}$ is $\mathsf{hib\text{-}cpa\text{-}priv}$ secure.

□

Theorem 2. *If k-level HIBS scheme* HIBS *is* $\mathsf{EUF\text{-}hID\text{-}CMA}$ *secure, then our HIB-ME scheme* $\mathsf{HIB\text{-}ME}$ *satisfies* $\mathsf{hib\text{-}auth}$ *security.*

Proof. Let A be a PPT adversary which can break $\mathsf{hib\text{-}auth}$ of our $\mathsf{HIB\text{-}ME}$. Then, we could build an algorithm B that breaks $\mathsf{EUF\text{-}hID\text{-}CMA}$ of HIBS as follows:

1. At the beginning, algorithm B receives $\mathsf{HIBS.mpk}$ from the $\mathsf{EUF\text{-}hID\text{-}CMA}$ challenger. Then, it executes $(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{msk}) \leftarrow \mathsf{HIBE}^1.\mathsf{Setup}(1^\lambda, l+1)$ and $(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{msk}) \leftarrow \mathsf{HIBE}^2.\mathsf{Setup}(1^\lambda, l+1)$, and sends $\mathsf{mpk} := (\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBS.mpk})$ to A.
2. For the queries made by A, B proceeds as follows:
   - When A makes encryption key queries for σ, B queries $\mathsf{EUF\text{-}hID\text{-}CMA}$ challenger for secret signing key on σ. B sets the $\mathsf{ek}_\sigma := \mathsf{sk}_\sigma$ received from the challenger, and returns ek_σ to A.
   - When A makes decryption key queries for ρ, B runs $\mathsf{HIBE}^1.\mathsf{sk}_\rho \leftarrow \mathsf{HIBE}^1.\mathsf{KeyDer}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{msk}, \rho)$ and $\mathsf{HIBE}^2.\mathsf{sk}_\rho \leftarrow \mathsf{HIBE}^2.\mathsf{KeyDer}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{msk}, \rho)$. Then B sets $\mathsf{dk}_\rho := (\mathsf{HIBE}^1.\mathsf{sk}_\rho, \mathsf{HIBE}^2.\mathsf{sk}_\rho)$ and returns dk_ρ to A.
   - When A makes ciphertext queries for $(\sigma, \mathsf{rcv}, \mathsf{M})$, B first queries $\mathsf{EUF\text{-}hID\text{-}CMA}$ challenger for signature on input $(\sigma, \mathsf{rcv}|\mathsf{M})$ and receives Σ, then it runs the encryption algorithm to obtain CT^1 and CT^2. Finally, it sends $\mathsf{CT} = (\mathsf{CT}^1, \mathsf{CT}^2)$ to A.
3. Once B receives the forgery output $(\mathsf{CT}^* = (\mathsf{CT}^{1*}, \mathsf{CT}^{2*}), \rho^*, \mathsf{snd}^*)$ from A, B executes in the following way:

(a) Computes $\mathsf{sk}^2_{\rho^*|\mathsf{snd}^*} \leftarrow \mathsf{HIBE}^2.\mathsf{KeyDer}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{HIBE}^2.\mathsf{sk}_\rho, \rho^*|\mathsf{snd}^*)$, $\Sigma^* \leftarrow \mathsf{HIBE}^2.\mathsf{Dec}(\mathsf{HIBE}^2.\mathsf{mpk}, \mathsf{sk}^2_{\rho^*|\mathsf{snd}^*}, \mathsf{CT}^{*2})$, $\mathsf{sk}^1_{\rho|\mathsf{snd}} \leftarrow \mathsf{HIBE}^1.\mathsf{KeyDer}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{HIBE}^1.\mathsf{sk}_\rho, \rho|\mathsf{snd})$ and $\mathsf{M} \leftarrow \mathsf{HIBE}^1.\mathsf{Dec}(\mathsf{HIBE}^1.\mathsf{mpk}, \mathsf{dk}^1_{\rho|\mathsf{snd}}, \mathsf{CT}^{*1})$.

(b) If $\mathsf{M}^* \neq \perp$, outputs $(\mathsf{snd}^*, \rho^*|\mathsf{M}^*, \Sigma)$ to EUF-hID-CMA as forgery. Otherwise, B halts.

All the oracle queries of A are perfectly simulated by B. It is obvious that Dec outputs $\mathsf{M}^* \neq \perp$ only if Σ is valid. (i.e. if $\mathsf{M}^* \neq \perp$, $\mathsf{HIBS.Verify}(\cdot) = 1$ holds.) If A never makes a forbidden query, it is also clear that B never makes a forbidden query. Thus, it holds that

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = \mathsf{Adv}^{\mathsf{EUF\text{-}hID\text{-}CMA}}_{\mathsf{HIBS},\mathsf{B}}(\lambda).$$

Now, HIBS satisfies EUF-hID-CMA security, $\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = negl(\lambda)$. That implies HIB-ME is hib-auth secure.

□

3.4 Instantiation

In this section, we give an instantiation over bilinear groups for our generic construction in Section 3.2. Here, we instantiate our generic construction by combining the Blazy-Kiltz-Pan anonymous HIBE (BKP-AHIBE) scheme [16] and an HIBS scheme which is obtained by applying the Naor transformation to the BKP-AHIBE scheme.

Let $\mathsf{KDF} : \mathbb{G}_T \to \mathbb{G}_2^3$ be a key derivation function. Let $\mathsf{MAC} := (\mathsf{KeyGen}_{\mathsf{MAC}}, \mathsf{Tag}, \mathsf{Verify})$ be an affine MAC with the message space $\mathcal{B}^{\leq m}$ for some finite set $\mathcal{B}$. Then, an instantiation of our HIB-ME scheme $(k, l) - \mathsf{HIB\text{-}ME} = (\mathsf{Setup}, \mathsf{SKDer}, \mathsf{RKDer}, \mathsf{Enc}, \mathsf{Dec})$ with the message space $\mathcal{M} = \mathbb{G}_T$ is provided as follows.

$\mathsf{Setup}(1^\lambda, k, l)$*:* The setup algorithm first picks $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_s \leftarrow \mathcal{D}_k$ and computes $(\mathbf{B}_i, \mathbf{x}_{i,0}, \ldots, \mathbf{x}_{i,l+1}, x'_{i,0}) = \mathsf{sk}_{\mathsf{MAC},i} \leftarrow \mathsf{KeyGen}_{\mathsf{MAC}}(1^\lambda)$ for $i \in \{1, 2, s\}$. Next, for $i \in \{1, 2\}$ and $j \in \{0, \ldots, l+1\}$, it computes

$$\mathbf{Y}_{i,j} \leftarrow \mathbb{Z}_q^{n \times m}, \quad \mathbf{Z}_{i,j} \leftarrow (\mathbf{Y}_{i,j}^\top | \mathbf{x}_{i,j}) \cdot \mathbf{A}_i \in \mathbb{Z}_q^{m \times n},$$

and, for $j = \{0, \ldots, k\}$,

$$\mathbf{Y}_{s,j} \leftarrow \mathbb{Z}_q^{n \times m}, \quad \mathbf{Z}_{s,j} \leftarrow (\mathbf{Y}_{s,j}^\top | \mathbf{x}_{s,j}) \cdot \mathbf{A}_s \in \mathbb{Z}_q^{m \times n}.$$

Furthermore, it picks $\mathbf{y}_{1,0}, \mathbf{y}_{2,0}, \mathbf{y}_{s,0} \leftarrow \mathbb{Z}_q^n$, and computes

$$\mathbf{z}'_{i,0} = (\mathbf{y}'_{i,0} | x'_{i,0}) \cdot \mathbf{A}_i \in \mathbb{Z}_q^{1 \times n} \quad (\text{for } i = \{1, 2, s\}).$$

Then, it sets

$$\mathsf{mpk} := (\mathcal{G}, [\mathbf{A}_1]_1, [\mathbf{A}_2]_1, [\mathbf{A}_s]_1, ([\mathbf{Z}_{1,j}]_1, [\mathbf{Z}_{2,j}]_1)_{0 \leq j \leq l+1}, ([\mathbf{Z}_{s,j}]_1)_{0 \leq j \leq k}, [\mathbf{z}'_{1,0}]_1, [\mathbf{z}'_{2,0}]_1, [\mathbf{z}'_{s,0}]_1),$$

and

msk
$\coloneqq ((\mathsf{sk}_{\mathsf{MAC},i})_{i\in\{1,2,s\}}, ([\mathbf{Y}_{1,j}]_1, [\mathbf{Y}_{2,j}]_1)_{0\le j\le l+1}, ([\mathbf{Y}_{s,j}]_1)_{0\le j\le k}, [\mathbf{y}'_{1,0}]_1, [\mathbf{y}'_{2,0}]_1, [\mathbf{y}'_{s,0}]_1)$.

Finally, it returns a master public/secret key pair $(\mathsf{mpk}, \mathsf{msk})$.
$\mathsf{SKDer}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$*:* The sender key derivation algorithm works as follows:

If $\mathsf{ek}_{\sigma'} = \mathsf{msk}$, it first computes $([\mathbf{t}_\sigma]_2, [u_\sigma]_2) \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC},s}, \sigma)$ and $\mathbf{v}_\sigma = \sum_{j=0}^{\mathsf{len}(\sigma)} f_j(\sigma)\mathbf{Y}_{s,j}\mathbf{t}_\sigma + \mathbf{y}'_{s,j}$. Next, it picks $\mathbf{S}_\sigma \leftarrow \mathbb{Z}_q^{m'\times\mu}$ and computes $\mathbf{T} = \mathbf{B}\cdot\mathbf{S} \in \mathbb{Z}_q^{m'\times\mu}$. Then, it computes

$$\mathbf{u}_\sigma = \sum_{j=0}^{\mathsf{len}(\sigma)} f_j(\sigma)\mathbf{x}_{s,j}^\top \mathbf{T}_\sigma \in \mathbb{Z}_q^{1\times\mu}, \quad \mathbf{V}_\sigma = \sum_{j=0}^{\mathsf{len}(\sigma)} f_j(\sigma)\mathbf{Y}_{s,j}\mathbf{T}_\sigma \in \mathbb{Z}_q^{n\times\mu}.$$

For $j = \mathsf{len}(\sigma)+1, \ldots, k$, it computes

$$d_{\sigma,j} = \mathbf{x}_{s,j}^\top \mathbf{t} \in \mathbb{Z}_q, \quad \mathbf{D}_{\sigma,j} = \mathbf{x}_{s,j}^\top \mathbf{T} \in \mathbb{Z}_q^{1\times\mu},$$
$$\mathbf{e}_{\sigma,j} = \mathbf{Y}_{s,j}\mathbf{t} \in \mathbb{Z}_q^n, \quad \mathbf{E}_{\sigma,j} = \mathbf{Y}_{s,j}\mathbf{T} \in \mathbb{Z}_q^{n\times\mu}.$$

Then, it sets $\mathsf{ek}'_\sigma \coloneqq ([\mathbf{t}_\sigma]_2, [u_\sigma]_2, [\mathbf{v}_\sigma]_2) \in \mathbb{G}_2^m \times \mathbb{G}_2^1 \times \mathbb{G}_2^n$ and $\mathsf{ek}''_\sigma \coloneqq ([\mathbf{T}_\sigma]_2, [\mathbf{u}_\sigma]_2, [\mathbf{V}_\sigma]_2, ([d_{\sigma,j}]_2, [\mathbf{D}_{\sigma,j}]_2, [\mathbf{e}_{\sigma,j}]_2, [\mathbf{E}_{\sigma,j}]_2)_{\mathsf{len}(\sigma)\le j\le k}) \in \mathbb{G}_2^{m\times\mu} \times \mathbb{G}_2^{1\times\mu} \times \mathbb{G}_2^{n\times\mu} \times (\mathbb{G}_2 \times \mathbb{G}_2^{1\times\mu} \times \mathbb{G}_2^n \times \mathbb{G}_2^{n\times\mu})^{k-\mathsf{len}(\sigma)}$. Finally, it returns $\mathsf{ek}_\sigma \coloneqq (\mathsf{ek}'_\sigma, \mathsf{ek}''_\sigma)$.

Else, it first parses $\hat\sigma \coloneqq (\sigma', \sigma) \in \mathcal{B}^{p+1}$. Next, it computes

$$\hat u_{\hat\sigma} = u_{\sigma'} + \sum_{j=\mathsf{len}(\sigma')+1}^{\mathsf{len}(\hat\sigma)} f_j(\hat\sigma) d_j \in \mathbb{Z}_q, \quad \hat{\mathbf{v}}_{\hat\sigma} = \mathbf{v}_{\sigma'} + \sum_{j=\mathsf{len}(\sigma')+1}^{\mathsf{len}(\hat\sigma)} f_j(\hat\sigma)\mathbf{e}_j \in \mathbb{Z}_q^n,$$
$$\hat{\mathbf{u}}_{\hat\sigma} = \mathbf{u}_{\sigma'} + \sum_{j=\mathsf{len}(\sigma')+1}^{\mathsf{len}(\hat\sigma)} f_j(\hat\sigma)\mathbf{D}_j \in \mathbb{Z}_q^{1\times\mu}, \quad \hat{\mathbf{V}}_{\hat\sigma} = \mathbf{V}_{\sigma'} + \sum_{j=\mathsf{len}(\sigma')+1}^{\mathsf{len}(\hat\sigma)} f_j(\hat\sigma)\mathbf{E}_j \in \mathbb{Z}_q^{n\times\mu}.$$

Then, it picks $\mathbf{s}' \leftarrow \mathbb{Z}_q^\mu, \mathbf{S} \leftarrow \mathbb{Z}_q^{\mu\times\mu}$ and computes

$$\mathbf{t}'_{\hat\sigma} = \mathbf{t}_\sigma + \mathbf{T}\mathbf{s}' \in \mathbb{Z}_q^m, \quad \mathbf{T}'_{\hat\sigma} = \hat{\mathbf{T}}\dot{\mathbf{S}}' \in \mathbb{Z}_q^{m\times\mu},$$
$$u'_{\hat\sigma} = \hat u_{\hat\sigma} + \hat{\mathbf{u}}_{\hat\sigma}\cdot\mathbf{s}' \in \mathbb{Z}_q, \quad \mathbf{u}'_{\hat\sigma} = \hat{\mathbf{u}}_{\hat\sigma}\cdot\mathbf{S}' \in \mathbb{Z}_q^{1\times\mu},$$
$$\mathbf{v}'_{\hat\sigma} = \hat{\mathbf{v}}_{\hat\sigma} + \hat{\mathbf{V}}_{\hat\sigma}\cdot\mathbf{s}' \in \mathbb{Z}_q^n, \quad \mathbf{V}'_{\hat\sigma} = \hat{\mathbf{V}}_{\hat\sigma}\cdot\mathbf{S}' \in \mathbb{Z}_q^{n\times\mu}.$$

For $j = \mathsf{len}(\hat\sigma), \ldots, k:$

$$d'_{\hat\sigma,j} = d_{\sigma',j} + \mathbf{D}_{\sigma',j}\mathbf{s}' \in \mathbb{Z}_q, \quad \mathbf{D}'_{\hat\sigma,j} = \mathbf{D}_{\sigma',j}\cdot\mathbf{S}' \in \mathbb{Z}_q^{1\times\mu},$$
$$\mathbf{e}'_{\hat\sigma,j} = \mathbf{e}_{\sigma',j} + \mathbf{E}_{\sigma',j}\mathbf{s}' \in \mathbb{Z}_q^n, \quad \mathbf{E}'_{\hat\sigma,j} = \mathbf{E}_{\sigma',j}\cdot\mathbf{S}' \in \mathbb{Z}_q^{n\times\mu}.$$

Then, it sets $\mathsf{ek}'_{\hat{\sigma}} = ([\mathbf{t}'_{\hat{\sigma}}]_2, [u'_{\hat{\sigma}}]_2, [\mathbf{v}'_{\hat{\sigma}}]_2)$ and $\mathsf{ek}''_{\hat{\sigma}} = ([\mathbf{T}'_{\hat{\sigma}}]_2, [\mathbf{u}'_{\hat{\sigma}}]_2, [\mathbf{V}'_{\hat{\sigma}}]_2, ([d'_{\hat{\sigma},j}]_2, [\mathbf{D}'_{\hat{\sigma},j}]_2, [\mathbf{e}'_{\hat{\sigma},j}]_2, [\mathbf{E}'_{\hat{\sigma},j}]_2)_{k(\hat{\sigma})<j\leq k})$. Finally, it returns $\mathsf{ek}_{\hat{\sigma}} := (\mathsf{ek}'_{\hat{\sigma}}, \mathsf{ek}''_{\hat{\sigma}})$.

$\mathsf{RKDer}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$*:* The receiver key derivation algorithm works as follows:

If $\mathsf{dk}_{\rho'} = \mathsf{msk}$, for $i \in \{1,2\}$, it first computes $([\mathbf{t}_{i,\rho}]_2, [u_{i,\rho}]_2) \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC},i}, \rho)$ and $\mathbf{v}_{i,\rho} = \sum_{j=0}^{\mathsf{len}(\rho)} f_j(\rho)\mathbf{Y}_{i,j}\mathbf{t}_{i,\rho} + \mathbf{y}'_{i,j}$. Next, it picks $\mathbf{S}_{1,\rho}, \mathbf{S}_{2,\rho} \leftarrow \mathbb{Z}_q^{m'\times\mu}$ and computes $\mathbf{T}_{1,\rho} = \mathbf{B}\cdot\mathbf{S}_{1,\rho} \in \mathbb{Z}_q^{m'\times\mu}$ and $\mathbf{T}_{2,\rho} = \mathbf{B}\cdot\mathbf{S}_{2,\rho} \in \mathbb{Z}_q^{m'\times\mu}$. Then, for $i \in \{1,2\}$, it computes

$$\mathbf{u}_{i,\rho} = \sum_{j=0}^{\mathsf{len}(\rho)} f_j(\rho)\mathbf{x}_{i,j}^\top\mathbf{T}_{i,\rho} \in \mathbb{Z}_q^{1\times\mu}, \quad \mathbf{V}_{i,\rho} = \sum_{j=0}^{\mathsf{len}(\rho)} f_j(\rho)\mathbf{Y}_{i,j}\mathbf{T}_{i,\rho} \in \mathbb{Z}_q^{n\times\mu}.$$

For $i = \{1,2\}, j = \mathsf{len}(\rho)+1, \ldots, l+1$, it computes

$$d_{\rho,i,j} = \mathbf{x}_{i,j}^\top\mathbf{t} \in \mathbb{Z}_q, \quad \mathbf{D}_{\rho,i,j} = \mathbf{x}_{i,j}^\top\mathbf{T} \in \mathbb{Z}_q^{1\times\mu},$$
$$\mathbf{e}_{\rho,i,j} = \mathbf{Y}_{i,j}\mathbf{t} \in \mathbb{Z}_q^{n}, \quad \mathbf{E}_{\rho,i,j} = \mathbf{Y}_{i,j}\mathbf{T} \in \mathbb{Z}_q^{n\times\mu}.$$

Then, it sets $\mathsf{dk}'_\rho := ([\mathbf{t}_{i,\rho}]_2, [u_{i,\rho}]_2, [\mathbf{v}_{i,\rho}]_2)_{i=\{1,2\}} \in (\mathbb{G}_2^m \times \mathbb{G}_2^1 \times \mathbb{G}_2^n)^2$ and $\mathsf{dk}''_\rho := (([\mathbf{T}_{i,\rho}]_2, [\mathbf{u}_{i,\rho}]_2, [\mathbf{V}_{i,\rho}]_2, ([d_{\rho,i,j}]_2, [\mathbf{D}_{\rho,i,j}]_2, [\mathbf{e}_{\rho,i,j}]_2, [\mathbf{E}_{\rho,i,j}]_2)_{\mathsf{len}(\rho)\leq j\leq l+1}))_{i=\{1,2\}} \in (\mathbb{G}_2^{m\times\mu} \times \mathbb{G}_2^{1\times\mu} \times \mathbb{G}_2^{n\times\mu} \times (\mathbb{G}_2 \times \mathbb{G}_2^{1\times\mu} \times \mathbb{G}_2^{n} \times \mathbb{G}_2^{n\times\mu})^{(l+1)-\mathsf{len}(\rho)})^2$. Finally, it returns $\mathsf{dk}_\rho := (\mathsf{dk}'_\rho, \mathsf{dk}''_\rho)$.

Else, it first parses $\hat{\rho} := (\rho', \rho) \in \mathcal{B}^{p+1}$. Next, for $i = \{1,2\}$ it computes

$$\hat{u}_{i,\hat{\rho}} = u_{i,\rho'} + \sum_{j=\mathsf{len}(\rho')+1}^{\mathsf{len}(\hat{\rho})} f_j(\hat{\rho})d_{\rho',i,j} \in \mathbb{Z}_q, \quad \hat{\mathbf{v}}_{i,\hat{\rho}} = \mathbf{v}_{i,\rho'} + \sum_{j=\mathsf{len}(\rho')+1}^{\mathsf{len}(\hat{\rho})} f_j(\hat{\rho})\mathbf{e}_{\rho',i,j} \in \mathbb{Z}_q^{n},$$
$$\hat{\mathbf{u}}_{i,\hat{\rho}} = \mathbf{u}_{i,\rho'} + \sum_{j=\mathsf{len}(\rho')+1}^{\mathsf{len}(\hat{\rho})} f_j(\hat{\rho})\mathbf{D}_{\rho,i,j} \in \mathbb{Z}_q^{1\times\mu}, \quad \hat{\mathbf{V}}_{i,\hat{\rho}} = \mathbf{V}_{i,\rho'} + \sum_{j=\mathsf{len}(\rho')+1}^{\mathsf{len}(\hat{\rho})} f_j(\hat{\rho})\mathbf{E}_{\rho,i,j} \in \mathbb{Z}_q^{n\times\mu}.$$

Then, it picks $\mathbf{s}'_1, \mathbf{s}'_2 \leftarrow \mathbb{Z}_q^{\mu}, \mathbf{S}'_1, \mathbf{S}'_2 \leftarrow \mathbb{Z}_q^{\mu\times\mu}$ and for $i = \{1,2\}$, it computes

$$\mathbf{t}'_{i,\hat{\rho}} = \mathbf{t}_{i,\rho} + \mathbf{T}\mathbf{s}'_i \in \mathbb{Z}_q^{m}, \quad \mathbf{T}'_{i,\hat{\rho}} = \hat{\mathbf{T}}_{i,\rho}\dot{\mathbf{S}}'_i \in \mathbb{Z}_q^{m\times\mu},$$
$$u'_{i,\hat{\rho}} = \hat{u}_{i,\hat{\rho}} + \hat{\mathbf{u}}_{i,\hat{\rho}} \cdot \mathbf{s}'_i \in \mathbb{Z}_q, \quad \mathbf{u}'_{i,\hat{\rho}} = \hat{\mathbf{u}}_{i,\hat{\rho}} \cdot \mathbf{S}'_i \in \mathbb{Z}_q^{1\times\mu},$$
$$\mathbf{v}'_{i,\hat{\rho}} = \hat{\mathbf{v}}_{i,\hat{\rho}} + \hat{\mathbf{V}}_{i,\hat{\rho}} \cdot \mathbf{s}'_i \in \mathbb{Z}_q^{n}, \quad \mathbf{V}'_{i,\hat{\rho}} = \hat{\mathbf{V}}_{i,\hat{\rho}} \cdot \mathbf{S}'_i \in \mathbb{Z}_q^{n\times\mu}.$$

For $i = \{1,2\}, j = \mathsf{len}(\hat{\sigma}), \ldots, l+1:$

$$d'_{\hat{\rho},i,j} = d_{\rho',i,j} + \mathbf{D}_{\rho',i,j}\mathbf{s}'_i \in \mathbb{Z}_q, \quad \mathbf{D}'_{\hat{\rho},i,j} = \mathbf{D}_{\rho',i,j} \cdot \mathbf{S}'_i \in \mathbb{Z}_q^{1\times\mu},$$
$$\mathbf{e}'_{\hat{\rho},i,j} = \mathbf{e}_{\rho',i,j} + \mathbf{E}_{\rho',i,j}\mathbf{s}'_i \in \mathbb{Z}_q^{n}, \quad \mathbf{E}'_{\hat{\rho},i,j} = \mathbf{E}_{\rho',i,j} \cdot \mathbf{S}'_i \in \mathbb{Z}_q^{n\times\mu}.$$

Then, it sets $\mathsf{dk}'_{\hat{\rho}} = ([\mathbf{t}'_{i,\hat{\rho}}]_2, [u'_{i,\hat{\rho}}]_2, [\mathbf{v}'_{i,\hat{\rho}}]_2)_{i=\{1,2\}}$ and $\mathsf{dk}''_{\hat{\rho}} = ([\mathbf{T}'_{i,\hat{\rho}}]_2, [\mathbf{u}'_{i,\hat{\rho}}]_2, [\mathbf{V}'_{i,\hat{\rho}}]_2, ([d'_{\hat{\rho},i,j}]_2, [\mathbf{D}'_{\hat{\rho},i,j}]_2, [\mathbf{e}'_{\hat{\rho},i,j}]_2, [\mathbf{E}'_{\hat{\rho},i,j}]_2)_{\mathsf{len}(\hat{\rho})<j\leq l+1})_{i=\{1,2\}}$. Finally, it returns $\mathsf{dk}_{\hat{\rho}} := (\mathsf{dk}'_{\hat{\rho}}, \mathsf{dk}''_{\hat{\rho}})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \rho, \mathsf{M} \in \mathbb{G}_T)$*:* The encryption algorithm first sets $\hat{\mathsf{M}} = \mathsf{rcv}|\mathsf{M}$, runs the SKDer algorithm with input ek_σ and $\hat{\mathsf{M}}$, and retrieves $\mathsf{ek}_{\hat{\mathsf{M}}} = ([\mathbf{t}_{\hat{\mathsf{M}}}]_2, [u_{\hat{\mathsf{M}}}]_2, [\mathbf{v}_{\hat{\mathsf{M}}}]_2)$. Next, it picks $\mathbf{r}_1, \mathbf{r}_2 \leftarrow \mathbb{Z}_q^n$ and computes

$$c_{0,1} = \mathbf{A}_1 \mathbf{r}_1 \in \mathbb{Z}_q^{n+1}, \quad c_{0,2} = \mathbf{A}_2 \mathbf{r}_2 \in \mathbb{Z}_q^{n+1},$$
$$c_{1,1} = \left(\sum_{j=0}^{\mathsf{len}(\mathsf{rcv}|\sigma)} f_j(\mathsf{rcv}|\sigma) \mathbf{Z}_j \right) \cdot \mathbf{r}_0, \quad c_{1,2} = \left(\sum_{j=0}^{\mathsf{len}(\mathsf{rcv}|\sigma)} f_j(\mathsf{rcv}|\sigma) \mathbf{Z}_j \right) \cdot \mathbf{r}_1.$$

Then, it computes $K_1 = \mathbf{z}'_{1,0} \cdot \mathbf{r}_1 \in \mathbb{Z}_q$ and $K_2 = \mathbf{z}'_{2,0} \cdot \mathbf{r}_2 \in \mathbb{Z}_q$. Finally, it computes $\mathsf{CT}_1 = [K_1]_T \cdot \mathsf{M}$, $(R_1, R_2, R_3) \leftarrow \mathsf{KDF}([K_2]_T)$ and

$$\mathsf{CT}_{2,1} = R_1 \cdot [\mathbf{t}_{\hat{\mathsf{M}}}]_2, \quad \mathsf{CT}_{2,2} = R_2 \cdot [u_{\hat{\mathsf{M}}}]_2, \quad \mathsf{CT}_{2,3} = R_3 \cdot [\mathbf{v}_{\hat{\mathsf{M}}}]_2,$$

and outputs $\mathsf{CT} = ([c_{0,1}]_1, [c_{0,2}]_1, [c_{1,1}]_1, [c_{1,2}]_1, [\mathsf{CT}_1]_T, [\mathsf{CT}_{2,1}]_2, [\mathsf{CT}_{2,2}]_2, [\mathsf{CT}_{2,3}]_2)$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \sigma, \mathsf{CT})$*:* The decryption algorithm first runs the RKDer algorithm with input dk_ρ and $\rho|\mathsf{snd}$ and retrieves $\mathsf{dk}_{\rho|\mathsf{snd}} := ([\mathbf{t}_{i,\rho|\mathsf{snd}}]_2, [u_{i,\rho|\mathsf{snd}}]_2, [\mathbf{v}_{i,\rho|\mathsf{snd}}]_2)_{i=\{1,2\}}$. Next, it computes

$$K_1 = e\left([c_{0,1}]_1, \begin{bmatrix} \mathbf{v}_{1,\rho|\mathsf{snd}} \\ u_{1,\rho|\mathsf{snd}} \end{bmatrix}_2\right) - e([c_{1,1}]_1, [\mathbf{t}_{1,\rho|\mathsf{snd}}]_2),$$
$$K_2 = e\left([c_{0,2}]_1, \begin{bmatrix} \mathbf{v}_{2,\rho|\mathsf{snd}} \\ u_{2,\rho|\mathsf{snd}} \end{bmatrix}_2\right) - e([c_{1,2}]_1, [\mathbf{t}_{2,\rho|\mathsf{snd}}]_2).$$

Next, it computes $\mathsf{M} = \mathsf{CT}_1/[K_1]_T$,$(R_1, R_2, R_3) \leftarrow \mathsf{KDF}([K_2]_T)$ and

$$[\mathbf{t}_{\mathsf{rcv}|\sigma}]_2 = [\mathsf{CT}_{2,1}]_2/R_1, \quad [u_{\mathsf{rcv}|\sigma}]_2 = [\mathsf{CT}_{2,2}]_2/R_2, \quad [\mathbf{v}_{\mathsf{rcv}|\sigma}]_2 = [\mathsf{CT}_{2,3}]_2/R_3.$$

Then, it picks $\mathbf{r} \leftarrow \mathbb{Z}_q^n$ and computes

$$c_0 = \mathbf{A}_2 \mathbf{r} \in \mathbb{Z}_q^{n+1}, \quad c_1 = \left(\sum_{j=0}^{\mathsf{len}(\mathsf{rcv}|\sigma)} f_j(\mathsf{rcv}|\sigma) \mathbf{Z}_j \right) \cdot \mathbf{r},$$

and $K = \mathbf{z}'_{2,0} \cdot \mathbf{r} \in \mathbb{Z}_q$. Then, it computes

$$K' = e\left([c_0]_1, \begin{bmatrix} \mathbf{v}_{\rho|\mathsf{snd}} \\ u_{\rho|\mathsf{snd}} \end{bmatrix}_2\right) - e([c_1]_1, [\mathbf{t}_{\rho|\mathsf{snd}}]_2).$$

If $K = K'$ holds, then it returns M. If not, it returns $\perp$.

4 Our CCA Secure (H)IB-ME Scheme

In this section, we provide our CCA secure (H)IB-ME scheme. In Section 4.1, we give the formal description of our l-level CCA secure (H)IB-ME scheme based on an $l+1$-level CPA secure (H)IB-ME scheme and a strong one-time signature scheme. Then, in Section 4.2, we show that our scheme satisfies $\mathsf{hib\text{-}cca\text{-}priv}$ security and $\mathsf{hib\text{-}auth}$ security.

4.1 Description

In this section, we give the formal description of our CCA secure l-level HIB-ME scheme from an $l+1$-level HIB-ME scheme and a strong one-time signature scheme. Roughly, toward CCA security, we extend the technique by Canetti et al. [14].

Construction. Fix integers $k \geq 0$ and $l \geq 1$. Let $\mathsf{HIB\text{-}ME}' = (\mathsf{Setup}', \mathsf{SKDer}', \mathsf{RKDer}', \mathsf{Enc}', \mathsf{Dec}')$ be a $(k, l+1)$-level HIB-ME scheme with a sender identity space $\mathcal{ID}$ and a receiver identity space $\mathcal{ID}' = \{0,1\}|\mathcal{ID}$. Let $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a strong one-time signature scheme with a verification key space $\mathcal{ID}$. Then, our (k,l)-level HIB-ME scheme $\mathsf{HIB\text{-}ME} = (\mathsf{Setup}, \mathsf{SKDer}, \mathsf{RKDer}, \mathsf{Enc}, \mathsf{Dec})$ scheme is described as follows:

$\mathsf{Setup}(1^\lambda, k, l)$*:* On the input a security parameter 1^λ, a maximum hierarchical level of sender k, and a maximum hierarchical level of receiver l, the setup algorithm runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}'(1^\lambda, k, l+1)$ and outputs $(\mathsf{mpk}, \mathsf{msk})$.
$\mathsf{SKDer}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$*:* On the input a master public key mpk, an encryption key $\mathsf{ek}_{\sigma'}$ associated to σ', and a sender identity σ, the sender key derivation algorithm runs $\mathsf{ek}_\sigma \leftarrow \mathsf{SKDer}'(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$ and outputs an encryption key ek_σ.
$\mathsf{RKDer}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$*:* On the input a master public key mpk, a decryption key $\mathsf{dk}_{\rho'}$ associated to ρ', and a receiver identity ρ, the receiver key derivation algorithm runs $\mathsf{dk}_\rho \leftarrow \mathsf{RKDer}'(\mathsf{mpk}, \mathsf{dk}_{\rho'}, 0|\rho)$ and outputs a decryption key dk_ρ.
$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{M})$*:* On the input a master public key mpk, an encryption key ek_σ, a target receiver identity rcv, and a plaintext M, the encryption algorithm firstly runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Next, it sets $\widehat{\mathsf{rcv}} := 0|\mathsf{rcv}.1|\mathsf{vk}$ and computes $\mathsf{CT}_1 \leftarrow \mathsf{Enc}'(\mathsf{mpk}, \mathsf{ek}_\sigma, \widehat{\mathsf{rcv}}, \mathsf{M})$ and $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{CT}_1)$. Finally, it outputs $\mathsf{CT} := (\mathsf{CT}_1, \mathsf{vk}, \Sigma)$.
$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{CT})$*:* On the input a master public key mpk, a decryption key dk_ρ, a target sender identity snd, and a ciphertext CT, the decryption algorithm checks whether $\perp = \mathsf{Verify}(\mathsf{vk}, \mathsf{CT}_1, \Sigma)$ holds. If this is the case, then it returns $\perp$. Otherwise, it sets $\hat{\rho} = 0|\rho.1|\mathsf{vk}$ and generates $\mathsf{dk}_{\hat{\rho}} \leftarrow \mathsf{RKDer}'(\mathsf{mpk}, \mathsf{dk}_\rho, \hat{\rho})$. Finally, it runs $\mathsf{M} \leftarrow \mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_{\hat{\rho}}, \mathsf{snd}, \mathsf{CT}_1)$ and outputs the plaintext M.

Correctness. It is obvious that the correctness of $\mathsf{HIB\text{-}ME}$ holds due to the correctness of $\mathsf{HIB\text{-}ME}'$ and Sig.

4.2 Security Proofs

In this section, we show that our scheme satisfies security requirements.
Theorem 3. *If* $\mathsf{HIB\text{-}ME}'$ *is* $\mathsf{hib\text{-}cpa\text{-}priv}$ *secure and* Sig *is* $\mathsf{sEUF\text{-}CMA}$ *secure, then* $\mathsf{HIB\text{-}ME}$ *is* $\mathsf{hib\text{-}cca\text{-}priv}$ *secure.*

Proof. Let A be a PPT adversary against the $\mathsf{hib\text{-}cca\text{-}priv}$ security of $\mathsf{HIB\text{-}ME}$. Within the experiment $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$, let $(\mathsf{CT}_1^*, \mathsf{vk}^*, \Sigma^*)$ be a challenge ciphertext. We define the events Forge and Succ in $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ as follows:

Forge*:* The adversary A makes at least one decryption query $(\rho^*, (\mathsf{CT}_1, \mathsf{vk}^*, \Sigma))$ satisfying $\top = \mathsf{Verify}(\mathsf{vk}^*, \mathsf{CT}_1, \Sigma))$, where ρ^* is the challenge receiver identity of A.
Succ*:* The adversary A outputs $\widehat{\mathsf{coin}}$ satisfying $\widehat{\mathsf{coin}} = \mathsf{coin}$, where $\mathsf{coin} \in \{0, 1\}$ is a challenge bit.

Using Forge and Succ, we can evaluate the advantage of A in $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ as

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)\\
&= 2 \cdot |\Pr[\mathsf{Succ}] - \frac{1}{2}|\\
&= 2 \cdot |\Pr[\mathsf{Succ} \wedge \mathsf{Forge}] + \Pr[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] - 1/2\Pr[\mathsf{Forge}] + 1/2\Pr[\mathsf{Forge}] - 1/2|\\
&\leq 2 \cdot (|\Pr[\mathsf{Succ} \wedge \mathsf{Forge}] - 1/2\Pr[\mathsf{Forge}]| + |\Pr[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + 1/2\Pr[\mathsf{Forge}] - 1/2])\\
&\leq \Pr[\mathsf{Forge}] + |2 \cdot \Pr[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \Pr[\mathsf{Forge}] - 1|\\
&= \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda) + \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}}(\lambda)
\end{aligned}
$$

Lemma 3. *There exists an adversary* $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ *against the* $\mathsf{sEUF\text{-}CMA}$ *security of* Sig *such that* $\Pr[\mathsf{Forge}] = \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}}(\lambda)$.

Proof. We construct a PPT adversary $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ that attacks the $\mathsf{sEUF\text{-}CMA}$ security of Sig so that $\Pr[\mathsf{Forge}] = \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}}(\lambda)$, using the adversary A.

1. As the setup, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ first receives vk^* from its challenger, runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}'(1^\lambda, k, l+1)$, and sends mpk to A. Furthermore, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ picks $\mathsf{coin} \leftarrow \{0, 1\}$.
2. When A makes oracle queries, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ responds as follows:
   - When A makes a sender key derivation query σ, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ runs $\mathsf{ek}_\sigma \leftarrow \mathsf{SKDer}'(\mathsf{mpk}, \mathsf{msk}, \sigma)$ and returns ek_σ to A.
   - When A makes a receiver key derivation query ρ, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ runs $\mathsf{dk}_\rho \leftarrow \mathsf{RKDer}'(\mathsf{mpk}, \mathsf{msk}, \rho)$ and return dk_ρ to A.
   - When A makes a decryption query of $(\mathsf{snd}, \rho, \mathsf{CT} = (\mathsf{CT}_1, \Sigma, \mathsf{vk}))$, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ checks whether $\mathsf{vk} = \mathsf{vk}^*$ and $\top = \mathsf{Verify}(\mathsf{vk}^*, \mathsf{CT}_1, \Sigma)$ hold. If this is the case, then it outputs (CT_1, Σ) as its forgery and terminates. Otherwise, it checks whether $\perp = \mathsf{Verify}(\mathsf{vk}, \mathsf{CT}_1, \Sigma)$ holds. If this is the case, then it returns $\perp$ to A. Otherwise, it sets $\hat{\rho} = 0|\rho.1|\mathsf{vk}$, runs $\mathsf{dk}_\rho \leftarrow \mathsf{RKDer}'(\mathsf{mpk}, \mathsf{msk}, \hat{\rho})$ and $\mathsf{M} \leftarrow \mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{CT}_1)$, and returns M to A.
   - When A makes a challenge query of $(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*)$, if $\mathsf{coin} = 0$ holds, then $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ runs $\mathsf{ek}_{\sigma^*} \leftarrow \mathsf{SKDer}'(\mathsf{mpk}, \mathsf{msk}, \sigma^*)$ and $\mathsf{CT}_1^* \leftarrow \mathsf{Enc}'(\mathsf{mpk}, \mathsf{ek}_{\sigma^*}, 0|\rho.1|\mathsf{vk}^*, \mathsf{M}^*)$. Otherwise (that is, $\mathsf{coin} = 1$ holds), $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ runs $\mathsf{CT}_1^* \leftarrow \mathsf{HIB\text{-}ME}'.\mathsf{CTSamp}(\mathsf{mpk})$. Then, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ makes a signing query CT_1^* and gets a signature Σ^*. Finally, B returns $(\mathsf{vk}^*, \mathsf{CT}_1^*, \Sigma^*)$ to A.

3. When A outputs $\widehat{\mathsf{coin}}$ and terminates, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ halts.

From the above construction, $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ perfectly simulates $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ for A. Now, we assume that Forge happens, that is, A makes a decryption query $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{vk}^*, \Sigma)$ that satisfies $\mathsf{Verify}(\mathsf{vk}^*, \mathsf{CT}_1, \Sigma) = \top$ at least once. Since $(\mathsf{CT}_1^*, \Sigma^*) \neq (\mathsf{CT}_1, \Sigma)$ holds from the requirements for decryption queries by A, the tuple of a message and a signature (CT_1, Σ) output by $\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}$ satisfies the winning conditions of the experiment $\mathsf{Exp}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}}(\lambda)$. Therefore, $\Pr[\mathsf{Forge}] = \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}}(\lambda)$ holds. □

Lemma 4. *There exists an adversary* $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ *against the* $\mathsf{hib\text{-}cpa\text{-}priv}$ *security of* $\mathsf{HIB\text{-}ME}'$ *such that* $|2 \cdot \Pr[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \Pr[\mathsf{Forge}] - 1| = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda)$.

Proof. We construct a PPT adversary $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ that attacks the $\mathsf{hib\text{-}cpa\text{-}priv}$ security of $\mathsf{HIB\text{-}ME}'$ so that $|2 \cdot \Pr[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \Pr[\mathsf{Forge}] - 1| = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda)$, using the adversary A.

1. As a setup, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ receives mpk from its challenger, runs $(\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends mpk to A.
2. When A makes oracle queries, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ responds as follows:
   - When A makes a sender key derivation query σ, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ also makes a sender key derivation query σ to its challenger to obtain ek_σ and returns ek_σ to A.
   - When A makes a receiver key derivation query ρ, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ also makes a receiver key derivation query ρ to its challenger to obtain dk_ρ and returns dk_ρ to A.
   - When A makes a decryption query of $(\mathsf{snd}, \rho, \mathsf{CT})$, depending on whether $\rho = \rho^*$ or $\mathsf{vk} = \mathsf{vk}^*$, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ proceeds as follows:

     $\rho = \rho^* \wedge \mathsf{vk} = \mathsf{vk}^*$*:* If $\top = \mathsf{Verify}(\mathsf{vk}^*, \mathsf{CT}_1, \Sigma)$ holds, then $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ outputs $\widehat{\mathsf{coin}} \leftarrow \{0, 1\}$ and terminates. Otherwise, if $\perp = \mathsf{Verify}(\mathsf{vk}^*, \mathsf{CT}_1, \Sigma)$, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ returns $\perp$ to A.

     $(\rho = \rho^* \wedge \mathsf{vk} \neq \mathsf{vk}^*) \vee \rho \neq \rho^*$*:* $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ makes a decryption key derivation query on $\hat{\rho} = 0|\rho.1|\mathsf{vk}$ to its challenger to obtain $\mathsf{dk}_{\hat{\rho}}$, computes $\mathsf{M} \leftarrow \mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_{\hat{\rho}}, \mathsf{snd}, \mathsf{CT}_1)$, and returns M to A. (Note that $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ is allowed to make a query $0|\rho.1|\mathsf{vk}$ since it is not a prefix of the target receiver identity ρ^*).
   - When A makes a challenge query of $(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*)$, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ also makes a challenge query of $(\sigma^*, 0|\mathsf{rcv}^*.1|\mathsf{vk}^*, \mathsf{M}^*)$ to its challenger and gets CT_1^*. Then, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ computes $\Sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}^*, \mathsf{CT}_1^*)$ and returns a challenge ciphertext $\mathsf{CT}^* := (\mathsf{vk}^*, \mathsf{CT}_1^*, \Sigma^*)$ to A.
3. Finally, when A outputs $\widehat{\mathsf{coin}}$ and terminates, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ outputs the same $\widehat{\mathsf{coin}}$ and terminates.

Since $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ does not get the secret key corresponding to the target identity $0|\mathsf{rcv}^*.1|\mathsf{vk}^*$, unless Forge occurs, $\mathsf{Exp}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ is perfectly simulated to A and the challenge bits of $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ and A correspond. On the other hand, when Forge occurs, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ outputs a random bit $\widehat{\mathsf{coin}}$. Here, let $\mathsf{Succ}^{\mathsf{B}}$ be an event that $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$

breaks the $\mathsf{hib\text{-}cpa\text{-}priv}$ security of $\mathsf{HIB\text{-}ME}'$. Then, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda) &= |2\cdot\Pr[\mathsf{Succ}^{\mathsf{B}}]-1| \\ &= |2\cdot(\Pr[\mathsf{Succ}\wedge\overline{\mathsf{Forge}}]+\Pr[\mathsf{Forge}]\cdot\frac{1}{2})-1| \\ &= |2\cdot\Pr[\mathsf{Succ}\wedge\overline{\mathsf{Forge}}]+\Pr[\mathsf{Forge}]-1|.\end{aligned}$$

□

From Lemma 3 and Lemma 4, it holds that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) &= |\Pr[\mathsf{Succ}\wedge\mathsf{Forge}]+\Pr[\mathsf{Succ}\wedge\overline{\mathsf{Forge}}] \\ &\quad -1/2\Pr[\mathsf{Forge}]+1/2\Pr[\mathsf{Forge}]-1/2| \\ &\leq |\Pr[\mathsf{Succ}\wedge\mathsf{Forge}]-1/2\Pr[\mathsf{Forge}]| \\ &\quad +|\Pr[\mathsf{Succ}\wedge\overline{\mathsf{Forge}}]+1/2\Pr[\mathsf{Forge}]-1/2| \\ &\leq \Pr[\mathsf{Forge}]+|\Pr[\mathsf{Succ}\wedge\overline{\mathsf{Forge}}]+\Pr[\mathsf{Forge}]-1| \\ &\leq \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda)+\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathsf{B}^{\mathsf{sEUF\text{-}CMA}}}(\lambda).\end{aligned}$$

Now, since we assume that $\mathsf{HIB\text{-}ME}'$ is $\mathsf{hib\text{-}cpa\text{-}priv}$ secure and Sig is $\mathsf{sEUF\text{-}CMA}$ secure, we obtain $\mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)=negl(\lambda)$, that is, $\mathsf{HIB\text{-}ME}$ satisfies $\mathsf{hib\text{-}cca\text{-}priv}$ security. □

Theorem 4. *If* $\mathsf{HIB\text{-}ME}'$ *is* $\mathsf{hib\text{-}auth}$ *secure, then* $\mathsf{HIB\text{-}ME}$ *is* $\mathsf{hib\text{-}auth}$ *secure.*

Proof. Let A be a PPT adversary against the $\mathsf{hib\text{-}auth}$ security of $\mathsf{HIB\text{-}ME}$. Then, using A, we construct a PPT adversary B against the $\mathsf{hib\text{-}auth}$ security of $\mathsf{HIB\text{-}ME}'$ as follows:

1. At the beginning, when B receives mpk from the challenger, it sends mpk to A.
2. When A makes oracle queries, B answers as follows:
   - When A makes an encryption key query σ, B also makes an encryption key query σ to its challenger. Upon receiving ek_σ from the challenger, B returns it to A.
   - When A makes a decryption key query ρ, B also makes an decryption key query ρ to its challenger. Upon receiving dk_ρ from the challenger, B returns it to A.
   - When A makes a ciphertext query $(\sigma,\mathsf{rcv},\mathsf{M})$, B generates $(\mathsf{sk},\mathsf{vk})\leftarrow\mathsf{Sig.KeyGen}(1^\lambda)$, makes a ciphertext query $(\sigma, 0|\mathsf{rcv}.1|\mathsf{vk},\mathsf{M})$, and receives CT_1 from the challenger. Then, B computes $\Sigma\leftarrow\mathsf{Sig.Sign}(\mathsf{sk},\mathsf{CT}_1)$ and returns $\mathsf{CT}=(\mathsf{CT}_1,\mathsf{vk},\Sigma)$ to A.
3. When A outputs a forgery $(\mathsf{CT}^*,\rho^*,\mathsf{snd}^*)$, B firstly checks whether $\mathsf{Sig.Verify}(\mathsf{vk},\Sigma)=\top$ holds. If this is not the case, then B halts. Otherwise, B outputs $(\mathsf{CT}_1^*, 0|\rho^*.1|\mathsf{vk},\mathsf{snd}^*)$ and terminates.

From the above construction, we can see that B perfectly simulates the game of $\mathsf{hib\text{-}auth}$ security for A. Also, if A never makes a forbidden query, it is also clear that B never makes a forbidden query. Thus, it holds that

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)=\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}}(\lambda).$$

Since $\mathsf{HIB\text{-}ME}'$ satisfies $\mathsf{hib\text{-}auth}$ security, $\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = negl(\lambda)$. That is, $\mathsf{HIB\text{-}ME}$ is $\mathsf{hib\text{-}auth}$ secure. □

5 Tweaked CCA Security for (H)IB-ME

In this section, we introduce a slightly weak but reasonable CCA security notion, called *tweaked CCA security*, for (H)IB-ME. Informally, tweaked CCA security is the same as (standard) CCA security except that the (secret) encryption key used in generating challenge ciphertexts is not allowed to leaked.

5.1 Formalization of Tweaked CCA Security

In this section, we provide the formal definition of tweaked CCA security for (H)IB-ME.

Definition 7 (Tweaked CCA Security for HIB-ME). *Let* (k,l)-$\mathsf{HIB\text{-}ME}$ *be an HIB-ME scheme. We say that* $\mathsf{HIB\text{-}ME}$ *satisfies* $\mathsf{hib\text{-}tcca\text{-}priv}$ *security if for all PPT adversaries* A,

$$\mathsf{Adv}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) := \left| \Pr\left[\mathsf{coin} = \widehat{\mathsf{coin}} \;\middle|\; \begin{array}{l} \mathcal{Q}_{\mathsf{O}_S}, \mathcal{Q}_{\mathsf{O}_R}, \mathcal{Q}_{\mathsf{O}_D} := \emptyset; \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{coin} \leftarrow \{0,1\}; \\ (\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*) \leftarrow \mathsf{A}^{\mathsf{O}_S,\mathsf{O}_R,\mathsf{O}_D}(\mathsf{mpk}); \\ \mathsf{ek}_{\sigma^*} \leftarrow \mathsf{SKDer}(\mathsf{mpk}, \mathsf{msk}, \sigma^*); \\ \textit{If } (\mathsf{coin} = 0) \quad \mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_{\sigma^*}, \mathsf{rcv}^*, \mathsf{M}^*); \\ \textit{Else} \quad \mathsf{CT}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk}); \\ \widehat{\mathsf{coin}} \leftarrow \mathsf{A}^{\mathsf{O}_S,\mathsf{O}_R,\mathsf{O}_D}(\mathsf{CT}^*); \end{array} \right] - \frac{1}{2} \right|$$
$$= negl(\lambda) holds, where$$

each oracle and list update is the same in Fig.1. We require that A *is not allowed to make a sender key derivation query* $\forall \sigma', \sigma' \in \mathtt{prefix}(\sigma^*)$ *to* O_S*, a receiver key derivation query* $\forall \mathsf{rcv}', \mathsf{rcv}' \in \mathtt{prefix}(\mathsf{rcv}^*)$ *to* O_R *and* $(\sigma^*, \mathsf{rcv}^*, \mathsf{CT}^*)$ *to* O_D*.*

5.2 Description

In this section, we provide a construction of tweaked CCA secure (H)IB-ME. Our tweaked CCA secure (H)IB-ME scheme can be obtained solely based on a CPA secure (H)IB-ME scheme. Notably, compared to the previous CCA secure (H)IB-ME scheme in Section 4.1, our tweaked CCA secure (H)IB-ME scheme does not need a strong one-time signature scheme which incurs a ciphertext overhead (with the length of a verification key and a signature). Note that, in the (ordinary) IBE setting, the non-adaptive CCA security (a.k.a. the CCA1 security) can only be achieved with a similar construction, while in the IB-ME setting, we can achieve more reasonable security notion (adaptive security but with query limitations).

Construction. Fix integers $k \geq 0$ and $l \geq 1$. Let $\mathsf{HIB\text{-}ME}' = (\mathsf{Setup}', \mathsf{SKDer}', \mathsf{RKDer}', \mathsf{Enc}', \mathsf{Dec}')$ be a $(k, l+1)$-level HIB-ME scheme with a sender identity space $\mathcal{ID}$ and a receiver identity space $\mathcal{ID}' = \{0,1\}|\mathcal{ID}$. Then, we show how to construct (k,l)-level HIB-ME scheme $\mathsf{HIB\text{-}ME} = (\mathsf{Setup}, \mathsf{SKDer}, \mathsf{RKDer}, \mathsf{Enc}, \mathsf{Dec})$. Setup algorithm Setup, sender key derivation algorithm SKDer, and receiver key derivation algorithm RKDer is the same as construction in section 4.1. Now, we show encryption algorithm Enc and decryption algorithm Dec as follows:

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{M})$*:* On the input a master public key mpk, an encryption key ek_σ, a target receiver identity rcv, and a plaintext M, the encryption algorithm sets $\widehat{\mathsf{rcv}} := 0|\mathsf{rcv}.1|\sigma$ and computes $\mathsf{CT} \leftarrow \mathsf{Enc}'(\mathsf{mpk}, \mathsf{ek}_\sigma, \widehat{\mathsf{rcv}}, \mathsf{M})$. Finally, it outputs CT.
$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{CT})$*:* On the input a master public key mpk, a decryption key dk_ρ, a target sender identity snd, and a ciphertext CT, the decryption algorithm sets $\hat{\rho} = 0|\rho.1|\mathsf{snd}$ and generates $\mathsf{dk}_{\hat{\rho}} \leftarrow \mathsf{RKDer}'(\mathsf{mpk}, \mathsf{dk}_\rho, \hat{\rho})$. Finally, it runs $\mathsf{M} \leftarrow \mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_{\hat{\rho}}, \mathsf{snd}, \mathsf{CT})$ and outputs the plaintext M.

5.3 Security Proofs

In this section, we show that our (H)IB-ME scheme given in Section 5.2 satisfies hib-tcca-priv security.

Theorem 5. *If* HIB-ME′ *satisfies* hib-cpa-priv *security and* hib-auth *security, then* HIB-ME *is* hib-tcca-priv *secure.*

Proof. Let A be a PPT adversary against the hib-tcca-priv security of HIB-ME. We introduce the following games.

Game_0*:* This is an original game of $\mathsf{Exp}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$.
Game_1*:* Same as Game_0, except that, when A makes a decryption query $(\rho^*, \mathsf{snd}^*, \cdot)$, the challenger returns $\perp$ to A.

In the following, for $i \in \{0,1\}$, let X_i denote an event that $\mathsf{coin} = \widehat{coin}$ in Game_i. Then, we can estimate the advantage

$$\begin{aligned}\mathsf{Adv}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) &= 2 \cdot |\Pr[\mathsf{coin} = \widehat{\mathsf{coin}}] - \frac{1}{2}| \\ &\leq |\Pr[X_0] - \Pr[X_1]| + |\Pr[X_1] - \frac{1}{2}|.\end{aligned}$$

Lemma 5. *There exists an adversary* $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ *against the* hib-auth *security of* HIB-ME′ *such that* $|\Pr[X_0] - \Pr[X_1]| = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}auth}}}(\lambda)$.

Proof. First of all, for $i \in \{0,1\}$, we define an event Bad_i that A makes a decryption query of $(\rho^*, \mathsf{snd}^*, \mathsf{CT})$ satisfying $\perp \neq \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{\rho^*}, \mathsf{snd}^*, \mathsf{CT})$ in Game_i, where ρ^* and snd^* are challenge receiver identity and sender identity, respectively. Game_0 proceeds identically to Game_1 unless Bad_0 occurs. That is, $|\Pr[X_0]-\Pr[X_1]| \leq \Pr[\mathsf{Bad}_0]$ holds. In the following, we show that one can construct a PPT adversary B against the hib-auth security of HIB-ME′ so that $\Pr[\mathsf{Bad}] = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}auth}}}(\lambda)$, using the adversary A.

1. As the setup of the hib-auth game, upon receiving a master public key mpk from the challenger, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ sends mpk to A and picks $\mathsf{coin} \leftarrow \{0,1\}$.
2. When A makes oracle queries, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ answers them as follows:

- When A makes a sender key derivation query σ, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ also makes a sender key derivation query σ to its challenger and gets ek_σ. Then, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ returns ek_σ to A and updates $\mathcal{Q}_{\mathsf{O}_S} \leftarrow \mathcal{Q}_{\mathsf{O}_S} \cup \{\sigma\}$.
- When A makes a receiver key derivation query ρ, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ also makes a receiver key derivation query ρ to its challenger and gets dk_ρ. Then, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ returns dk_ρ to A.
- When A makes a decryption query of $(\rho, \mathsf{snd}, \mathsf{CT})$, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ makes a receiver key derivation query ρ to its challenger, gets dk_ρ, and computes $\mathsf{M} \leftarrow \mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{CT})$. Then, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ checks that $\mathsf{snd} \notin \mathcal{Q}_{\mathsf{O}_S} \wedge \mathsf{M} \neq \perp$ holds. If this is the case, then $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ outputs $(\rho, \mathsf{snd}, \mathsf{CT})$ as its forgery and terminates. Otherwise, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ returns M to A.
- When A makes a challenge query of $(\sigma^*, \rho^*, \mathsf{M}^*)$, if $\mathsf{coin} = 0$ holds, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ make encryption query with $(\sigma^*, \rho^*, \mathsf{M}^*)$ to its challenger, gets CT^*, and returns CT^* to A. Otherwise, $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ samples $\mathsf{CT}^* \leftarrow \mathsf{CTSamp}(\mathsf{mpk})$ and returns CT^* to A.

3. When A outputs $\widehat{\mathsf{coin}}$, B halts.

From the above construction, we can see that $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ perfectly simulates Game_0 for A. Now, we assume that Bad happens, that is, A makes a decryption query $(\rho^*, \mathsf{snd}^*, \mathsf{CT})$ satisfying $\mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_{\rho^*}, \mathsf{snd}^*, \mathsf{CT}) \neq \perp$ at least once. Since $(\rho^*, \mathsf{snd}^*, \mathsf{CT}^*) \neq (\rho, \mathsf{snd}, \mathsf{CT})$ and $\mathtt{prefix}(\mathsf{snd}^*) \notin \mathcal{Q}_{\mathsf{O}_S}$ hold from the requirements for A, the forgery $(\rho, \mathsf{snd}, \mathsf{CT})$ output by $\mathsf{B}^{\mathsf{hib\text{-}auth}}$ satisfies the winning conditions of the $\mathsf{hib\text{-}auth}$ security game. Therefore, $\Pr[\mathsf{Bad}] = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}', \mathsf{B}^{\mathsf{hib\text{-}auth}}}(\lambda)$ holds. $\square$

Lemma 6. *There exists an adversary* $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ *against the* $\mathsf{hib\text{-}cpa\text{-}priv}$ *security of* $\mathsf{HIB\text{-}ME}'$ *such that* $|\Pr[X_1] - \frac{1}{2}| = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}', \mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda)$.

Proof. We construct a PPT adversary $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ who attacks the $\mathsf{hib\text{-}cpa\text{-}priv}$ security of $\mathsf{HIB\text{-}ME}'$ so that $|\Pr[X_1] - \frac{1}{2}| = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}', \mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda)$, using the adversary A, as follows:

1. As a setup, upon receiving mpk from the challenger, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ sends mpk to A.
2. When A makes oracle queries, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ answers as follows:
   - When A makes a sender key derivation query σ, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ also makes a sender key derivation query σ to its challenger to obtain ek_σ and returns ek_σ to A.
   - When A makes a receiver key derivation query ρ, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ also makes a receiver key derivation query ρ to its challenger to obtain dk_ρ and returns dk_ρ to A.
   - When A makes a decryption query of $(\mathsf{snd}, \rho, \mathsf{CT})$, depending on whether $\rho = \rho^*$ or $\mathsf{snd} = \mathsf{snd}^*$, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ proceeds as follows:

     $\rho = \rho^* \wedge \mathsf{snd} = \mathsf{snd}^*$*:* $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ just returns $\perp$ to A.
     $(\rho = \rho^* \wedge \mathsf{snd} \neq \mathsf{snd}^*) \vee \rho \neq \rho^*$*:* $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ makes a decryption key derivation query $\hat{\rho} = 0|\rho.1|\mathsf{snd}$ to its challenger to obtain $\mathsf{dk}_{\hat{\rho}}$, computes $\mathsf{M} \leftarrow \mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_{\hat{\rho}}, \mathsf{snd}, \mathsf{CT})$, and returns M to A. (Note that $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ is allowed to make a query $0|\rho.1|\mathsf{snd}$ since it is not a prefix of the target identity ρ^*.)

- When A makes a challenge query of $(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*)$, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ also makes a challenge query of $(\sigma^*, \mathsf{rcv}^*, \mathsf{M}^*)$ to its challenger to obtain CT^* and returns CT^* to A.

3. When A outputs $\widehat{\mathsf{coin}}$ and terminates, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ outputs the same $\widehat{\mathsf{coin}}$ and terminates.

Since $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ does not obtain the secret key corresponding to the target identity $0|\rho^*.1|\mathsf{snd}^*$, $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$, $\mathsf{Exp}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda)$ is perfectly simulated to A and the challenge bit of $\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}$ and A correspond. Thus, it holds that

$$|\Pr[X_1] - 1/2| = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda).$$

□

Putting everything together, it holds that

$$\mathsf{Adv}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}} \leq \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}cpa\text{-}priv}}}(\lambda) + \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}^{\mathsf{hib\text{-}auth}}}(\lambda).$$

Now, since $\mathsf{HIB\text{-}ME}'$ satisfies $\mathsf{hib\text{-}cpa\text{-}priv}$ security and $\mathsf{hib\text{-}auth}$ security, we obtain $\mathsf{Adv}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = negl(\lambda)$. That is, $\mathsf{HIB\text{-}ME}$ satisfies $\mathsf{hib\text{-}tcca\text{-}priv}$ security.

□

Theorem 6. *If* $\mathsf{HIB\text{-}ME}'$ *satisfies* $\mathsf{hib\text{-}auth}$ *security, then* $\mathsf{HIB\text{-}ME}$ *is* $\mathsf{hib\text{-}auth}$ *secure.*

Proof. Let A be a PPT adversary against the $\mathsf{hib\text{-}auth}$ security of $\mathsf{HIB\text{-}ME}$. Then, using A, we construct a PPT adversary B against the $\mathsf{hib\text{-}auth}$ security of $\mathsf{HIB\text{-}ME}'$ as follows:

1. At the beginning, when B receives mpk from the challenger, it sends mpk to A.
2. When A makes oracle queries, B answers as follows:
   - When A makes an encryption key query σ, B also makes an encryption key query σ to its challenger. Upon receiving ek_σ from the challenger, B returns it to A.
   - When A makes a decryption key query ρ, B also makes a decryption key query ρ to its challenger. Upon receiving dk_ρ from the challenger, B returns it to A.
   - When A makes a ciphertext query $(\sigma, \mathsf{rcv}, \mathsf{M})$, B also makes a ciphertext query $(\sigma, 0|\mathsf{rcv}.1|\sigma, \mathsf{M})$ to its challenger. Upon receiving a ciphertext CT from the challenger, B returns it to A.
3. When A outputs a forgery $(\mathsf{CT}^*, \rho^*, \mathsf{snd}^*)$, B outputs a forgery $(\mathsf{CT}^*, 0.\rho^*.1|\mathsf{snd}^*, \mathsf{snd}^*)$ and terminates.

From the above construction, we can see that B perfectly simulates the game of $\mathsf{hib\text{-}auth}$ security for A. Also, if A never makes a forbidden query, it is also clear that B never makes a forbidden query. Thus, it holds that

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathsf{B}}(\lambda).$$

Since $\mathsf{HIB\text{-}ME}'$ satisfies $\mathsf{hib\text{-}auth}$ security, $\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathsf{A}}(\lambda) = negl(\lambda)$ holds. That is, $\mathsf{HIB\text{-}ME}$ is $\mathsf{hib\text{-}auth}$ secure. □

Declarations

Conflict of interest The authors declare no conflict of interest.
Ethical approval The authors declare full compliance with ethical standards. This article does not contain any studies involving humans or animals performed by any of the authors.
Data availability The used data are publicly available.

Acknowledgement

This research was in part conducted under a contract of "Research and development on new generation cryptography for secure wireless communication services" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan. This work also was in part supported by JSPS KAKENHI Grant Numbers JP22H03590 and JP21H03395, JST-CREST JPMJCR22M1, and JST-AIP JPMJCR22U5.

References

[1] Ateniese, G., Francati, D., Nuñez, D. & Venturi, D. Boldyreva, A. & Micciancio, D. (eds) *Match me if you can: Matchmaking encryption and its applications.* (eds Boldyreva, A. & Micciancio, D.) *Advances in Cryptology – CRYPTO 2019, Part II*, Vol. 11693 of *Lecture Notes in Computer Science*, 701–731 (Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 2019).

[2] Attrapadung, N. *et al.* Relations among notions of security for identity based encryption schemes. Cryptology ePrint Archive, Report 2005/258 (2005). https://eprint.iacr.org/2005/258.

[3] Dolev, D., Dwork, C. & Naor, M. Nonmalleable cryptography. *SIAM Journal on Computing* **30**, 391–437 (2000).

[4] Galindo, D. & Hasuo, I. Security notions for identity based encryption. Cryptology ePrint Archive, Report 2005/253 (2005). https://eprint.iacr.org/2005/253.

[5] Bellare, M., Desai, A., Pointcheval, D. & Rogaway, P. Krawczyk, H. (ed.) *Relations among notions of security for public-key encryption schemes.* (ed.Krawczyk, H.) *Advances in Cryptology – CRYPTO'98*, Vol. 1462 of *Lecture Notes in Computer Science*, 26–45 (Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 1998).

[6] Boneh, D. & Franklin, M. K. Kilian, J. (ed.) *Identity-based encryption from the Weil pairing.* (ed.Kilian, J.) *Advances in Cryptology – CRYPTO 2001*, Vol. 2139 of *Lecture Notes in Computer Science*, 213–229 (Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 2001).

[7] Canetti, R. & Goldwasser, S. Stern, J. (ed.) *An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack.* (ed.Stern, J.) *Advances in Cryptology – EUROCRYPT'99*, Vol. 1592 of *Lecture Notes in Computer Science*, 90–106 (Springer, Heidelberg, Germany, Prague, Czech Republic, 1999).

[8] Canetti, R. & Hohenberger, S. Ning, P., De Capitani di Vimercati, S. & Syverson, P. F. (eds) *Chosen-ciphertext secure proxy re-encryption.* (eds Ning, P., De Capitani di Vimercati, S. & Syverson, P. F.) *ACM CCS 2007: 14th Conference on Computer and Communications Security*, 185–194 (ACM Press, Alexandria, Virginia, USA, 2007).

[9] Koppula, V. & Waters, B. Boldyreva, A. & Micciancio, D. (eds) *Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption.* (eds Boldyreva, A. & Micciancio, D.) *Advances in Cryptology – CRYPTO 2019, Part II*, Vol. 11693 of *Lecture Notes in Computer Science*, 671–700 (Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 2019).

[10] Yamada, S., Attrapadung, N., Hanaoka, G. & Kunihiro, N. Catalano, D., Fazio, N., Gennaro, R. & Nicolosi, A. (eds) *Generic constructions for chosen-ciphertext secure attribute based encryption.* (eds Catalano, D., Fazio, N., Gennaro, R. & Nicolosi, A.) *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, Vol. 6571 of *Lecture Notes in Computer Science*, 71–89 (Springer, Heidelberg, Germany, Taormina, Italy, 2011).

[11] Chiku, S., Hashimoto, K., Hara, K. & Shikata, J. Identity-based matchmaking encryption, revisited: Strong security and practical constructions from standard classical and post-quantum assumptions. Cryptology ePrint Archive, Paper 2023/1435 (2023). https://eprint.iacr.org/2023/1435.

[12] Fujisaki, E. & Okamoto, T. Imai, H. & Zheng, Y. (eds) *How to enhance the security of public-key encryption at minimum cost.* (eds Imai, H. & Zheng, Y.) *PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography*, Vol. 1560 of *Lecture Notes in Computer Science*, 53–68 (Springer, Heidelberg, Germany, Kamakura, Japan, 1999).

[13] Gentry, C. & Silverberg, A. Zheng, Y. (ed.) *Hierarchical ID-based cryptography.* (ed.Zheng, Y.) *Advances in Cryptology – ASIACRYPT 2002*, Vol. 2501 of *Lecture Notes in Computer Science*, 548–566 (Springer, Heidelberg, Germany, Queenstown, New Zealand, 2002).

[14] Canetti, R., Halevi, S. & Katz, J. Cachin, C. & Camenisch, J. (eds) *Chosen-ciphertext security from identity-based encryption.* (eds Cachin, C. & Camenisch, J.) *Advances in Cryptology – EUROCRYPT 2004*, Vol. 3027 of *Lecture Notes in Computer Science*, 207–222 (Springer, Heidelberg, Germany, Interlaken, Switzerland, 2004).

[15] Agrawal, S., Boneh, D. & Boyen, X. Gilbert, H. (ed.) *Efficient lattice (H)IBE in the standard model.* (ed.Gilbert, H.) *Advances in Cryptology – EUROCRYPT 2010*, Vol. 6110 of *Lecture Notes in Computer Science*, 553–572 (Springer, Heidelberg, Germany, French Riviera, 2010).

[16] Blazy, O., Kiltz, E. & Pan, J. Garay, J. A. & Gennaro, R. (eds) *(Hierarchical) identity-based encryption from affine message authentication.* (eds Garay, J. A. & Gennaro, R.) *Advances in Cryptology – CRYPTO 2014, Part I*, Vol. 8616 of *Lecture Notes in Computer Science*, 408–425 (Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 2014).

[17] Boyen, X. & Waters, B. Dwork, C. (ed.) *Anonymous hierarchical identity-based encryption (without random oracles).* (ed.Dwork, C.) *Advances in Cryptology – CRYPTO 2006*, Vol. 4117 of *Lecture Notes in Computer Science*, 290–307 (Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 2006).

[18] Cash, D., Hofheinz, D., Kiltz, E. & Peikert, C. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology* **25**, 601–639 (2012).

[19] Kiltz, E. & Neven, G. *Identity-Based Signatures*, 31–44. Cryptology and Information Security Series on Identity-Based Cryptography (I, 2008).

[20] Seo, J. H., Kobayashi, T., Ohkubo, M. & Suzuki, K. Jarecki, S. & Tsudik, G. (eds) *Anonymous hierarchical identity-based encryption with constant size ciphertexts.* (eds Jarecki, S. & Tsudik, G.) *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, Vol. 5443 of *Lecture Notes in Computer Science*, 215–234 (Springer, Heidelberg, Germany, Irvine, CA, USA, 2009).

[21] Ateniese, G., Francati, D., Nuñez, D. & Venturi, D. Match me if you can: Matchmaking encryption and its applications. *Journal of Cryptology* **34**, 16 (2021).

[22] Francati, D., Guidi, A., Russo, L. & Venturi, D. Adhikari, A., Küsters, R. & Preneel, B. (eds) *Identity-based matchmaking encryption without random oracles.* (eds Adhikari, A., Küsters, R. & Preneel, B.) *INDOCRYPT 2021*, Vol. 13143 of *LNCS*, 415–435 (Springer, Heidelberg, 2021).

[23] Chen, J., Li, Y., Wen, J. & Weng, J. Agrawal, S. & Lin, D. (eds) *Identity-based matchmaking encryption from standard assumptions.* (eds Agrawal, S. & Lin, D.) *Advances in Cryptology – ASIACRYPT 2022, Part III*, Vol. 13793 of *Lecture Notes in Computer Science*, 394–422 (Springer, Heidelberg, Germany, Taipei, Taiwan, 2022).

[24] Francati, D., Friolo, D., Malavolta, G. & Venturi, D. Hazay, C. & Stam, M. (eds) *Multi-key and multi-input predicate encryption from learning with errors.* (eds Hazay, C. & Stam, M.) *Advances in Cryptology – EUROCRYPT 2023*, 573–604

(Springer Nature Switzerland, Cham, 2023).

[25] Lin, S., Li, Y. & Chen, J. Ge, C. & Yung, M. (eds) *Cca-secure identity-based matchmaking encryption from standard assumptions.* (eds Ge, C. & Yung, M.) *Information Security and Cryptology*, 253–273 (Springer Nature Singapore, Singapore, 2024).

[26] Wang, Y., Wang, B., Lai, Q. & Zhan, Y. Identity-based matchmaking encryption with stronger security and instantiation on lattices. Cryptology ePrint Archive, Report 2022/1718 (2022). https://eprint.iacr.org/2022/1718.