Hierarchical Identity-Based Matchmaking Encryption

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

[1] Yokohama National University, Yokohama, Japan chiku-sohto-tw@ynu.jp, shikata-junji-rb@ynu.ac.jp
[2] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
hara-keisuke@aist.go.jp

Abstract. Identity-based matchmaking encryption (IB-ME) is an advanced encryption scheme that allows both the sender and the receiver to specify their respective identities. We study the notion of *hierarchical IB-ME (HIB-ME)*, which augments IB-ME with delegation capabilities.
Specifically, we first formalize HIB-ME and construct it based on hierarchical identity-based encryption and hierarchical identity-based signature. Moreover, as applications of the HIB-ME, we show two chosen ciphertext secure (H)IB-ME constructions for different security levels.

1 Introduction

1.1 Background and Motivation

Identity-based Matchmaking Encryption. Identity-based matchmaking encryption (IB-ME), proposed by Ateniese et al. [AFNV19], is a novel extension of the ordinary encryption system. In this system, both the sender and receiver can specify appropriate identities, which must be satisfied for the message to be revealed. More specifically, in IB-ME, as a 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 the 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 do 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. [AFNV19] demonstrated that a privacy-preserving bulletin board system [3] (over a Tor network) could 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.

Prior Works. Following the seminal work [AFNV19, AFNV21], research on various flavors of IB-ME has been carried out. Francati et al. [FGRV21] proposed a mismatch-cases privacy and gave a construction from a q-type assumption in the plain model. Chen et al. [CLWW22] dismantle q-type assumption and proposed the first IB-ME construction from the standard assumption in the standard model. Wang et al. [WWLZ25] proposed a generic construction of IB-ME based on a 2-level anonymous HIBE and an identity-based signature scheme. Boyen and Li [BL23] constructed IB-ME that satisfy enhanced privacy under standard assumptions using an anonymous IBE, an identity-based signature, an average-case randomness extractor, and a reusable computational extractor. Belfiore et al. [BCF24] constructed IB-ME that achieves the notion of enhanced privacy using as building blocks an anonymous IBE, a homomorphic signature, and a reusable computational extractor. There exists some works [CHHS23, LLC24, WWLZ25] by taking another approach to achieve CCA secure IB-ME as follows. Chiku et al. [CHHS23] towards CCA secure IB-ME schemes by taking two other approaches. One is based on combining Boneh and Franklin IBE scheme [BF01], Sakai et al.'s identity-based

[3] A bulletin board is an online platform where users can post messages, share information, and engage in discussions on various topics. It allows for asynchronous communication, enabling users to interact at their convenience.

Table 1. Comparison between our IB-ME scheme and the existing CCA secure IB-ME schemes. "Anon IBE" stands for anonymous identity-based encryption, "IBS" stands for identity-based signature, "2-HIB-ME" stands for HIB-ME with 2-level receiver delegation, "OTS" stands for one-time strong signature, and "(Q)ROM" (resp., "StdM") stands for (quantum) random oracle model (resp., standard model).

Schemes	Privacy Level	Crypto. Primitives	Assumption	Model
Chiku et al. [CHHS23]	CCA	Anon IBE + IBS	BDH	ROM
Chiku et al. [CHHS23] + [DLP14, PFH+22]	CCA		NTRU	QROM
Lin et al. [LLC24]	CCA		SXDH	StdM
CHK w/ OTS (§ 4.1) + [BKP14]	CCA	2-HIB-ME + OTS	k-lin	StdM
CHK w/ OTS (§ 4.1) + [ABB10]	CCA	2-HIB-ME + OTS	LWE	StdM
CHK w/o OTS (§ 4.2) + [BKP14]	tweaked CCA	2-HIB-ME	k-lin	StdM
CHK w/o OTS (§ 4.2) + [ABB10]	tweaked CCA	2-HIB-ME	LWE	StdM

non-interactive key exchange protocol [SOK00], and Fujisaki-Okamoto transformation [FO99]. Another one is depending on a CCA secure IBE scheme, an IBS scheme, and a hash function modeled as a random oracle. These constructions are toward not only CCA security but also enhanced privacy. Lin et al. [LLC24] towards a CCA secure IB-ME scheme using the CHK conversion technique. This study applies the CHK technique to specific pairing-based IBE and constructs a pairing-based CCA secure IB-ME construction. Wang et al. [WWLZ25] insist on their IB-ME towards CCA security by replacing CPA secure HIBE with CCA secure HIBE, but there is no security proof for CCA privacy.

Motivation. In this paper, we aim to formulate a model and give a general construction for *hierarchical IB-ME (HIB-ME)*, thereby providing a basis for further exploration of its theoretical applications. Incorporating hierarchical structures into identity-based primitives is a natural and important step in the evolution of advanced cryptographic primitives with centralized authority. This extension enables the distribution of key generation tasks across multiple levels, reducing the burden on central authorities and improving overall efficiency. Additionally, this extension has a potential for an application to achieve CCA security (e.g., [CHK04]).

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. 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. Regarding security aspects, we define CPA privacy, CCA privacy, tweaked CCA privacy, and authenticity. 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) by extending the Wang et al.'s CPA secure IB-ME construction [WWLZ25].

Extension of CHK conversion in HIB-ME. As an application of HIB-ME, we propose a CCA secure (H)IB-ME scheme in the standard model. Specifically, we offer a systematic analysis of the Canetti-Halevi-Katz (CHK) conversion technique [CHK04] within the (H)IB-ME framework and propose two distinct variants of the CHK conversion tailored for (H)IB-ME (one is for CCA security and the other is for tweaked CCA security). The resulting schemes are summarized in Table 1.

First, we show a natural extension of the CHK conversion, where a CCA secure (H)IB-ME is derived from CPA secure HIB-ME and one-time signatures. Regarding the efficiency, for example, the ciphertext size is almost the same as the underlying CPA secure (H)IB-ME scheme. From the previous works [ABB10, BKP14, BW06, CHKP12, KN08, SKOS09], we can realize our generic construction over bilinear groups or lattices.

Moreover, we introduce a slightly weak but reasonable CCA security notion, called *tweaked CCA security*, for (H)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 be leaked. As an advantage

of tweaked CCA security, we show that a tweaked CCA secure (H)IB-ME scheme can be constructed solely based on a CPA secure HIB-ME scheme (without strong one-time signature) by leveraging privacy and authenticity of the underlying HIB-ME scheme. (That is, regarding the ciphertext size, our tweaked CCA secure (H)IB-ME scheme does not have an overhead occurred by strong one-time signature.)

2 Preliminaries

Notations. In this paper, we use the following notations. For $n \in \mathbb{N}$, we denote $[n] = \{1, .., n\}$. Let $\lambda \in \mathbb{N}$ denote the security parameter. $x \leftarrow X$ denotes the operation of sampling an element x from a finite set X. $y \leftarrow \mathcal{A}(x; r)$ denotes that a probabilistic Turing machine $\mathcal{A}$ outputs y for an input x using a randomness r, and we simply denote $y \leftarrow \mathcal{A}(x)$ when we do not need to write the internal randomness explicitly. PPT stands for probabilistic polynomial time. $x := y$ 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 $\mathcal{O}$ is a function or an algorithm and $\mathcal{A}$ is an algorithm, $\mathcal{A}^{\mathcal{O}}$ means $\mathcal{A}$ has oracle access to $\mathcal{O}$. For a bit string x, $\mathsf{len}(x)$ denotes the length of x.

2.1 One-Time Digital Signature

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

$\mathsf{KGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$**:** 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.

$\mathsf{Ver}(\mathsf{vk}, \Sigma, \mathsf{m}) \to 1/0$**:** The verifying algorithm takes vk, and Σ as input, and outputs 1 (meaning "accept") or 0 (meaning "reject").

Correctness. The correctness for Sig requires that for all $\lambda \in \mathbb{N}, (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$ and $\mathsf{m} \in \mathcal{M}$, it holds that

$$\Pr[1 = \mathsf{Ver}(\mathsf{vk}, \Sigma, \mathsf{m}) \mid \Sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m})] = 1.$$

Security. Next, we define a one-time strong unforgeability ($\mathsf{sEUF\text{-}CMA}$ 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 $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{l|l} ((\mathsf{m}, \Sigma) \neq (\mathsf{m}^*, \Sigma^*)) \wedge & (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda); \\ (\mathsf{Ver}(\mathsf{vk}, \Sigma, \mathsf{m}) = 1) & \mathsf{m}^* \leftarrow \mathcal{A}(\mathsf{vk}); \\ & \Sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m}^*); \\ & (\Sigma, \mathsf{m}) \leftarrow \mathcal{A}(\mathsf{vk}, \Sigma^*); \end{array}\right] \leq \mathsf{negl}(\lambda).$$

2.2 Hierarchical Identity-Based Encryption

Let HIBE denote a hierarchical identity-based encryption (HIBE) [GS02, HL02, SW08]. HIBE for identity space $\mathcal{ID}$ and message space $\mathcal{M}$ consists of following five algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{KDel}, \mathsf{Enc}, \mathsf{Dec})$:

$\mathsf{Setup}(1^\lambda, l) \to (\mathsf{mpk}, \mathsf{msk})$**:** 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.

$\mathsf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$**:** The key generation algorithm takes mpk, msk, and a user identity ID, and outputs a user secret key $\mathsf{sk}_{\mathsf{ID}}$.

$\mathsf{KDel}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}}$**:** The key delegation algorithm takes mpk, $\mathsf{sk}_{\mathsf{ID}'}$, and a user identity ID, and outputs a user secret key $\mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}}$ for the $d+1$ depth identity $\mathsf{ID}'|\mathsf{ID}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{m}) \to \mathsf{ct}$**:** The encryption algorithm takes mpk, ID, and a plaintext m as input, and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \to \mathsf{m}/\perp$**:** The decryption algorithm takes mpk, $\mathsf{sk}_{\mathsf{ID}}$, and ct as input, and outputs m or $\perp$.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l)$, $\mathsf{ID} \in \mathcal{ID}$, $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$, and $\mathsf{m} \in \mathcal{M}$,

$$\Pr[\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}, \mathsf{m})) = \mathsf{m}] = 1.$$

Moreover, we also require the distribution of $\mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}} \leftarrow \mathsf{KDel}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID})$ is identical to the one from $\mathsf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}'|\mathsf{ID})$.

Security. Next, we define IND-hID-CPA security for a HIBE scheme.

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

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}hid\text{-}cpa}}_{\mathsf{HIBE},\mathcal{A}}(\lambda, l) &:= \left| \Pr\left[b = b' \;\middle|\; \begin{array}{l} b \leftarrow\!\$\ \{0,1\}, \mathcal{L}_{\mathsf{sk}} := \emptyset; \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l); \\ (\mathsf{ID}^*_0, \mathsf{ID}^*_1, \mathsf{m}^*_0, \mathsf{m}^*_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KGen}}, \mathcal{O}_{\mathsf{KDel}}, \mathcal{O}_{\mathsf{sk}}}(\mathsf{mpk}); \\ \mathsf{ct}^*_b \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ID}^*_b, \mathsf{m}^*_b); \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KGen}}, \mathcal{O}_{\mathsf{KDel}}, \mathcal{O}_{\mathsf{sk}}}(\mathsf{ct}^*_b); \end{array} \right] - \frac{1}{2} \right| \\ &= \mathsf{negl}(\lambda),\end{aligned}$$

with restriction $(\mathsf{ID}^*_b, \cdot) \in \mathcal{L}_{\mathsf{sk}} \wedge (\mathtt{prefix}(\mathsf{ID}^*_b), \cdot) \in \mathcal{L}_{\mathsf{sk}}$ for $b \in \{0,1\}$, and $\mathcal{O}_{\mathsf{KGen}}$ is queried only once for the same ID*. Now, we define three type of oracles that $\mathcal{A}$ can access as follows:*

- **Key Generation** $\mathcal{O}_{\mathsf{KGen}}(\mathsf{mpk}, \mathsf{msk}, \cdot)$**:** On input ID, the challenger runs $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$ and updates $\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \cup \{(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})\}$.
- **Key Delegate** $\mathcal{O}_{\mathsf{KDel}}(\mathsf{mpk}, \cdot, \cdot)$**:** On input $(\mathsf{ID}', \cdot) \in \mathcal{L}_{\mathsf{sk}}$, ID, the challenger extracts $\mathsf{sk}_{\mathsf{ID}'}$ from $\mathcal{L}_{\mathsf{sk}}$, runs $\mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}} \leftarrow \mathsf{KDel}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID})$, and updates $\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \cup \left\{(\mathsf{ID}'|\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}})\right\}$.
- **Key Reveal** $\mathcal{O}_{\mathsf{sk}}(\cdot)$**:** On input $(\mathsf{ID}, \cdot) \in \mathcal{L}_{\mathsf{sk}}$, the challenger extracts $\mathsf{sk}_{\mathsf{ID}}$ from $\mathcal{L}_{\mathsf{sk}}$, and returns $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$. In addition, the challenger updates $\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \setminus \{(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}})\}$.

2.3 Hierarchical Identity-Based Signature

Let HIBS denote a hierarchical identity-based signature (HIBS) [GS02,SW08]. HIBS consists of following five algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{KDel}, \mathsf{Sign}, \mathsf{Ver})$:

$\mathsf{Setup}(1^\lambda, l) \to (\mathsf{mpk}, \mathsf{msk})$**:** 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.

$\mathsf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$**:** The key generation algorithm takes mpk, msk, and a user identity ID, and outputs a user secret key $\mathsf{sk}_{\mathsf{ID}}$.

$\mathsf{KDel}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}}$**:** The key delegation algorithm takes mpk, $\mathsf{sk}_{\mathsf{ID}'}$, and a user identity ID, and outputs a user secret key $\mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}}$ for the $d+1$ depth identity $\mathsf{ID}'|\mathsf{ID}$.

$\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{m}) \to \Sigma$**:** 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.

$\mathsf{Ver}(\mathsf{mpk}, \mathsf{ID}, \Sigma, \mathsf{m}) \to 1/0$**:** The verifying algorithm takes mpk, $\mathsf{ID} \in \mathcal{ID}$, Σ, and m as input, and outputs 1 (meaning "accept") or 0 (meaning "reject").

Correctness. For corectness, we require that for all $\lambda \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l)$, $\mathsf{ID} \in \mathcal{ID}$, $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID})$, $\mathsf{m} \in \mathcal{M}$, and $\Sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{m})$,

$$\Pr[\mathsf{Ver}(\mathsf{mpk}, \mathsf{ID}, \Sigma, \mathsf{m}) = 1] = 1.$$

Moreover, we also require the distribution of $\mathsf{sk}_{\mathsf{ID}'|\mathsf{ID}} \leftarrow \mathsf{KDel}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}'}, \mathsf{ID})$ is identical to the one from $\mathsf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ID}'|\mathsf{ID})$.

Security. Next, we define EUF-hID-CMA security for a HIBS scheme.

Definition 3. *Let* HIBS *be an l-level HIBS scheme. We say that* HIBS *satisfies* EUF-hID-CMA *security if for all* PPT *adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{euf\text{-}hid\text{-}cma}}_{\mathsf{HIBS},\mathcal{A}}(\lambda, l) := \Pr\left[\begin{array}{c|l} (\mathsf{ID}^*, \cdot) \in \mathcal{L}_{\mathsf{sk}} \wedge & \mathcal{L}_{\mathsf{sk}}, \mathcal{L}_\Sigma := \emptyset; \\ (\mathtt{prefix}(\mathsf{ID}^*), \cdot) \in \mathcal{L}_{\mathsf{sk}} \wedge & (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, l); \\ ((\mathsf{ID}^*, \mathsf{m}^*, \cdot) \notin \mathcal{L}_\Sigma) \wedge & (\mathsf{ID}^*, \Sigma^*, \mathsf{m}^*) \\ (\mathsf{Ver}(\mathsf{mpk}, \mathsf{ID}^*, \Sigma^*, \mathsf{m}^*) \neq 0) & \quad \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KGen}}, \mathcal{O}_{\mathsf{KDel}}, \mathcal{O}_{\mathsf{sk}}, \mathcal{O}_{\mathsf{Sign}}}(\mathsf{mpk}); \end{array}\right]$$
$$= \mathsf{negl}(\lambda),$$

with restriction $\mathcal{O}_{\mathsf{KGen}}$ is queried only once for the same ID*. Now, we define four type of oracles that $\mathcal{A}$ can access as follows:*

- **Key Generation** $\mathcal{O}_{\mathsf{KGen}}(\mathsf{mpk}, \mathsf{msk}, \cdot)$**:** Same as Definition 2.
- **Key Delegate** $\mathcal{O}_{\mathsf{KDel}}(\mathsf{mpk}, \cdot, \cdot)$**:** Same as Definition 2.
- **Key Reveal** $\mathcal{O}_{\mathsf{sk}}(\cdot)$**:** Same as Definition 2.
- **Signature Generation** $\mathcal{O}_{\mathsf{Sign}}(\cdot, \cdot)$**:** On input $\mathsf{ID} \in \mathcal{L}_{\mathsf{sk}}$ and m, the challenger extracts $\mathsf{sk}_{\mathsf{ID}}$ from $\mathcal{L}_{\mathsf{sk}}$, runs $\Sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{m})$, and returns Σ to $\mathcal{A}$. In addition, the challenger updates $\mathcal{L}_\Sigma := \mathcal{L}_\Sigma \cup \{(\mathsf{ID}, \mathsf{m}, \Sigma)\}$.

3 Hierarchial Identity-Based Matchmaking Encryption

In this section, we introduce a new cryptographic primitive called hierarchical identity-based matchmaking encryption (HIB-ME).

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 HIB-ME denote an HIB-ME scheme with a maximum depth k for sender keys and depth l for receiver keys. (k, l)-level HIB-ME consists of the following seven algorithms $(\mathsf{Setup}, \mathsf{SKGen}, \mathsf{SKDel}, \mathsf{RKGen}, \mathsf{RKDel}, \mathsf{Enc}, \mathsf{Dec})$:

$\mathsf{Setup}(1^\lambda, k, l) \to (\mathsf{mpk}, \mathsf{msk})$**:** 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.

$\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma) \to \mathsf{ek}_\sigma$**:** The sender key generation algorithm takes mpk, msk, and a sender identity σ, and outputs a encryption key ek_σ.

$\mathsf{SKDel}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma) \to \mathsf{ek}_{\sigma'|\sigma}$**:** The sender key delegation algorithm takes mpk, $\mathsf{ek}_{\sigma'}$, and a sender identity σ, and outputs a encryption key $\mathsf{ek}_{\sigma'|\sigma}$ for the $d+1$ depth identity $\sigma'|\sigma$.

$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho) \to \mathsf{dk}_\rho$**:** The receiver key generation algorithm takes mpk, msk, and a receiver identity ρ, and outputs a decryption key dk_ρ.

$\mathsf{RKDel}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho) \to \mathsf{dk}_{\rho'|\rho}$**:** The receiver key delegation algorithm takes mpk, $\mathsf{dk}_{\rho'}$, and a receiver identity ρ, and outputs a decryption key $\mathsf{dk}_{\rho'|\rho}$ for the $d+1$ depth identity $\rho'|\rho$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{m}) \to \mathsf{ct}$**:** The encryption algorithm takes mpk, ek_σ, a target receiver identity rcv, and a plaintext m, and outputs a ciphertext m.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{ct}) \to \mathsf{m}/\perp$**:** The decryption algorithm takes mpk, dk_ρ, a target sender identity snd, and ct, and outputs m or $\perp$.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, k, l)$, $\sigma, \mathsf{snd}, \rho, \mathsf{rcv} \in \mathcal{ID}$ such that $\sigma = \mathsf{snd} \wedge \rho = \mathsf{rcv}$, $\mathsf{ek}_\sigma \leftarrow \mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma)$, $\mathsf{dk}_\rho \leftarrow \mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho)$, and $\mathsf{m} \in \mathcal{M}$,

$$\Pr[\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{m})) = \mathsf{m}] = 1.$$

Moreover, we also require the distribution of $\mathsf{ek}_{\sigma'|\sigma} \leftarrow \mathsf{SKDel}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$ (resp., $\mathsf{dk}_{\rho'|\rho} \leftarrow \mathsf{RKDel}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$) is identical to the one from $\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma'|\sigma)$ (resp., $\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho'|\rho)$).

Security. Next, we define security requirements (hib-cca-priv security, hib-tcca-priv security, hib-cpa-priv security, and hib-auth security) for HIB-ME. hib-cca-priv security is based on [CHHS23,LLC24], While hib-cpa-priv security is based on [AFNV21], and hib-auth security is based on [FGRV21], hib-tcca-priv security is original meaningful privacy definition. While CCA security is the most desirable security, we can consider a weaker version that $\mathcal{A}$ cannot get the (secret) encryption key of the target sender σ^*. In the tweaked CCA security experiment, $\mathcal{A}$ can no longer compute encryption using the challenge (secret) encryption key. Hence $\mathcal{A}$ is given access to encryption oracle $\mathcal{O}_{\mathsf{Enc}}$.

Definition 4 (Privacy). *Let* HIB-ME *be an (k, l)-level HIB-ME scheme. We define the advantage of each privacy game as*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow\!\$\ \{0,1\}, \mathcal{L}_{\mathsf{ek}}, \mathcal{L}_{\mathsf{dk}}, \mathcal{L}_{\mathsf{ct}} := \emptyset; \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, k, l); \\ (\sigma_0^*, \sigma_1^*, \mathsf{rcv}_0^*, \mathsf{rcv}_1^*, \mathsf{m}_0^*, \mathsf{m}_1^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{mpk}); \\ \mathsf{ek}_{\sigma_b^*} \leftarrow \mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma_b^*); \\ \mathsf{ct}_b^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_{\sigma_b^*}, \mathsf{rcv}_b^*, \mathsf{m}_b^*); \\ b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{ct}_b^*); \end{array} \right] - \frac{1}{2} \right|.$$

In addition, we define eight type of oracles as follows:

- **Sender Key Generate** $\mathcal{O}_{\mathsf{SKGen}}(\mathsf{mpk}, \mathsf{msk}, \cdot)$**:** On input σ, the challenger runs $\mathsf{ek}_\sigma \leftarrow \mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma)$ and updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \mathsf{ek}_\sigma)\}$.
- **Receiver Key Generate** $\mathcal{O}_{\mathsf{RKGen}}(\mathsf{mpk}, \mathsf{msk}, \cdot)$**:** On input ρ, the challenger runs $\mathsf{dk}_\rho \leftarrow \mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho)$ and updates $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \mathsf{dk}_\rho)\}$.
- **Sender Key Delegate** $\mathcal{O}_{\mathsf{SKDel}}(\mathsf{mpk}, \cdot, \cdot)$**:** On input σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$ and σ, the challenger extracts $\mathsf{ek}_{\sigma'}$ from $\mathcal{L}_{\mathsf{ek}}$, runs $\mathsf{ek}_{\sigma'|\sigma} \leftarrow \mathsf{SKDel}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$, and updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma, \mathsf{ek}_{\sigma'|\sigma})\}$.
- **Receiver Key Delegate** $\mathcal{O}_{\mathsf{RKDel}}(\mathsf{mpk}, \cdot, \cdot)$**:** On input ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$ and ρ, the challenger extracts $\mathsf{dk}_{\rho'}$ from $\mathcal{L}_{\mathsf{dk}}$, runs $\mathsf{dk}_{\rho'|\rho} \leftarrow \mathsf{RKDel}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$, and updates $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho, \mathsf{dk}_{\rho'|\rho})\}$.
- **Sender Key Reveal** $\mathcal{O}_{\mathsf{ek}}(\cdot)$**:** On input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$, the challenger extracts ek_σ from $\mathcal{L}_{\mathsf{ek}}$ and returns ek_σ to $\mathcal{A}$. In addition, the challenger updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \mathsf{ek}_\sigma)\}$.
- **Receiver Key Reveal** $\mathcal{O}_{\mathsf{dk}}(\cdot)$**:** On input ρ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$, the challenger extracts dk_ρ from $\mathcal{L}_{\mathsf{dk}}$ and returns dk_ρ to $\mathcal{A}$. In addition, the challenger updates $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \mathsf{dk}_\rho)\}$.
- **Encrypt** $\mathcal{O}_{\mathsf{Enc}}(\mathsf{mpk}, \cdot, \cdot, \cdot)$**:** On input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$, rcv, and m, the challenger extracts ek_σ from $\mathcal{L}_{\mathsf{ek}}$, runs $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{m})$, and returns ct to $\mathcal{A}$. In addition, the challenger updates $\mathcal{L}_{\mathsf{ct}} := \mathcal{L}_{\mathsf{ct}} \cup \{(\sigma, \mathsf{rcv}, \mathsf{m}, \mathsf{ct})\}$.
- **Decrypt** $\mathcal{O}_{\mathsf{Dec}}(\mathsf{mpk}, \cdot, \cdot, \cdot)$**:** On input ρ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$, snd, and ct, the challenger returns $\perp$ if $\mathsf{snd} = \sigma^* \wedge \rho = \mathsf{rcv}^* \wedge \mathsf{ct} = \mathsf{ct}_b^*$. Otherwise, it extracts dk_ρ from $\mathcal{L}_{\mathsf{dk}}$, runs $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{ct})$, and returns m to $\mathcal{A}$.

Now, we say that HIB-ME *satisfies:*

1. hib-cpa-priv *if for all* PPT *adversary $\mathcal{A}$ that can access to $\mathcal{O} := \{\mathcal{O}_{\mathsf{SKGen}}, \mathcal{O}_{\mathsf{RKGen}}, \mathcal{O}_{\mathsf{SKDel}}, \mathcal{O}_{\mathsf{RKDel}}, \mathcal{O}_{\mathsf{ek}}, \mathcal{O}_{\mathsf{dk}}\}$, it holds that $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{negl}(\lambda)$ with restriction $(\mathsf{rcv}^*, \cdot) \notin \mathcal{L}_{\mathsf{dk}} \wedge (\mathtt{prefix}(\mathsf{rcv}^*), \cdot) \in \mathcal{L}_{\mathsf{dk}}$.*
2. hib-cca-priv *if for all* PPT *adversary $\mathcal{A}$ that can access to $\mathcal{O} := \{\mathcal{O}_{\mathsf{SKGen}}, \mathcal{O}_{\mathsf{RKGen}}, \mathcal{O}_{\mathsf{SKDel}}, \mathcal{O}_{\mathsf{RKDel}}, \mathcal{O}_{\mathsf{ek}}, \mathcal{O}_{\mathsf{dk}}, \mathcal{O}_{\mathsf{Dec}}\}$, it holds that $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{negl}(\lambda)$ with restriction $(\mathsf{rcv}^*, \cdot) \in \mathcal{L}_{\mathsf{dk}} \wedge (\mathtt{prefix}(\mathsf{rcv}^*), \cdot) \in \mathcal{L}_{\mathsf{dk}}$.*

3. $\mathsf{hib\text{-}tcca\text{-}priv}$ *if for all* PPT *adversary* $\mathcal{A}$ *that can access to* $\mathcal{O} := \{\mathcal{O}_{\mathsf{SKGen}}, \mathcal{O}_{\mathsf{RKGen}}, \mathcal{O}_{\mathsf{SKDel}}, \mathcal{O}_{\mathsf{RKDel}}, \mathcal{O}_{\mathsf{ek}}, \mathcal{O}_{\mathsf{dk}}, \mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{Dec}}\}$*, it holds that* $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{negl}(\lambda)$ *with restriction* $(\sigma^*, \cdot) \in \mathcal{L}_{\mathsf{ek}} \wedge (\mathtt{prefix}(\sigma^*), \cdot) \in \mathcal{L}_{\mathsf{ek}} \wedge (\mathsf{rcv}^*, \cdot) \in \mathcal{L}_{\mathsf{dk}} \wedge (\mathtt{prefix}(\mathsf{rcv}^*), \cdot) \in \mathcal{L}_{\mathsf{dk}}$.

Definition 5 (Authenticity). *Let* HIB-ME *be an* (k, l)*-level HIB-ME scheme. We say that* HIB-ME *satisfies* $\mathsf{hib\text{-}auth}$ *security if for all* PPT *adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) := \Pr\left[\begin{array}{c|l} (\mathsf{snd}^*, \cdot) \in \mathcal{L}_{\mathsf{ek}} \wedge & \mathcal{L}_{\mathsf{ek}}, \mathcal{L}_{\mathsf{dk}}, \mathcal{L}_{\mathsf{ct}} := \emptyset; \\ (\mathsf{snd}^*, \mathsf{m}^*) \notin \mathcal{L}_{\mathsf{ct}} \wedge & (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, k, l); \\ \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{\rho^*}, \mathsf{snd}^*, \mathsf{ct}^*) \neq \bot & (\mathsf{snd}^*, \rho^*, \mathsf{ct}^*, \mathsf{m}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{mpk}); \end{array}\right] = \mathsf{negl}(\lambda),$$

with restriction $(\mathsf{snd}^*, \cdot) \in \mathcal{L}_{\mathsf{ek}} \wedge (\mathtt{prefix}(\mathsf{snd}^*), \cdot) \in \mathcal{L}_{\mathsf{ek}} \wedge (\mathsf{snd}^*, \rho^*, \cdot, \mathsf{ct}^*) \notin \mathcal{L}_{\mathsf{ct}}$*. Where* $\mathcal{A}$ *can access* $\mathcal{O} := \{\mathcal{O}_{\mathsf{SKGen}}, \mathcal{O}_{\mathsf{RKGen}}, \mathcal{O}_{\mathsf{SKDel}}, \mathcal{O}_{\mathsf{RKDel}}, \mathcal{O}_{\mathsf{ek}}, \mathcal{O}_{\mathsf{dk}}, \mathcal{O}_{\mathsf{Enc}}\}$ *defined in Definition 4.*

3.2 Construction from Anonymous HIBE and HIBS

In this section, we show a construction of CPA secure HIB-ME scheme. We can obtain $\mathsf{hib\text{-}cpa\text{-}priv}$ and $\mathsf{hib\text{-}auth}$ secure (k, l)-level HIB-ME scheme HIB-ME from k-level HIBS and $l+1$-level HIBE. The main idea of the construction is extension of Wang et al.'s sign-then-encrypt IB-ME scheme [WWLZ25].

Construction. Fix integers $k \geq 1$, $l \geq 1$. Let $\mathsf{HIBE} = (\mathsf{Setup}_{\mathsf{HIBE}}, \mathsf{KGen}_{\mathsf{HIBE}}, \mathsf{KDel}_{\mathsf{HIBE}}, \mathsf{Enc}_{\mathsf{HIBE}}, \mathsf{Dec}_{\mathsf{HIBE}})$ be an $l+1$-level HIBE scheme, and $\mathsf{HIBS} = (\mathsf{Setup}_{\mathsf{HIBS}}, \mathsf{KGen}_{\mathsf{HIBS}}, \mathsf{KDel}_{\mathsf{HIBS}}, \mathsf{Sign}_{\mathsf{HIBS}}, \mathsf{Ver}_{\mathsf{HIBS}})$ be a k-level HIBS scheme. Then, our (k, l)-level HIB-ME scheme HIB-ME $= (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{SKDel}, \mathsf{RKGen}, \mathsf{RKDel}, \mathsf{Enc}, \mathsf{Dec})$ is described as follows:

$\mathsf{Setup}(1^\lambda, k, l)$**:** It runs $(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{msk}_{\mathsf{HIBS}}) \leftarrow \mathsf{Setup}_{\mathsf{HIBS}}(1^\lambda, k)$ and $(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{msk}_{\mathsf{HIBE}}) \leftarrow \mathsf{Setup}_{\mathsf{HIBE}}(1^\lambda, l+1)$. Then, it outputs $(\mathsf{mpk}, \mathsf{msk})$ where $\mathsf{mpk} := (\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{mpk}_{\mathsf{HIBE}})$ and $\mathsf{msk} := (\mathsf{msk}_{\mathsf{HIBS}}, \mathsf{msk}_{\mathsf{HIBE}})$.

$\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma)$**:** It runs $\mathsf{ek}_\sigma \leftarrow \mathsf{KGen}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{msk}_{\mathsf{HIBS}}, \sigma)$ and outputs ek_σ.

$\mathsf{SKDel}(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$**:** It runs $\mathsf{ek}_{\sigma'|\sigma} \leftarrow \mathsf{KDel}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{ek}_{\sigma'}, \sigma)$ and outputs $\mathsf{ek}_{\sigma|\sigma'}$.

$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho)$**:** It runs $\mathsf{dk}_\rho \leftarrow \mathsf{KGen}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{msk}_{\mathsf{HIBE}}, \rho)$ and outputs dk_ρ.

$\mathsf{RKDel}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$**:** It runs $\mathsf{dk}_{\rho'|\rho} \leftarrow \mathsf{KDel}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{dk}_{\rho'}, \rho)$ and outputs $\mathsf{dk}_{\rho|\rho'}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}|\sigma, \mathsf{m})$**:** It runs $\Sigma \leftarrow \mathsf{Sign}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{ek}_\sigma, \mathsf{m}||\mathsf{rcv})$ and $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{rcv}|\sigma, \mathsf{m}||\Sigma)$, and outputs ct.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{ct})$**:** It runs $\mathsf{dk}_{\rho|\mathsf{snd}} \leftarrow \mathsf{KDel}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{dk}_\rho, \mathsf{snd})$ $\mathsf{m}||\Sigma \leftarrow \mathsf{Dec}_{\mathsf{HIBE}}(\mathsf{mpk}, \mathsf{dk}_{\rho|\sigma}, \mathsf{ct})$. If $\mathsf{Ver}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{snd}, \Sigma, \mathsf{m}||\rho) = 1$, it outputs m. Otherwise, it outputs $\bot$.

Correctness. If the ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{rcv}|\mathsf{snd}, \mathsf{m}||\Sigma)$ generated correctly, $\mathsf{Ver}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{snd}, \sigma, \mathsf{m}||\rho)$ always outputs 1 due to the correctness of HIBS. Hence, if HIBE satisfies correctness, HIB-ME also satisfies correctness.

Security. Here, we show that our scheme satisfies security requirements.

Theorem 1. *Suppose that the HIBE scheme* HIBE *is* IND-hID-CPA *secure. If there exists an adversary* $\mathcal{A}$ *that breaks* $\mathsf{hib\text{-}cpa\text{-}priv}$ *security if* HIB-ME*, there exists an adversary* $\mathcal{B}$ *that breaks* IND-hID-CPA *security of* HIBE *such that*

$$\mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{ind\text{-}hid\text{-}cpa}}_{\mathsf{HIBE},\mathcal{B}}(\lambda, l+1)$$

where k *(resp.,* l*) is the maximum depth of senders (resp., receivers). The running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Proof. Let $\mathcal{A}$ be an adversary that breaks the $\mathsf{hib\text{-}cpa\text{-}priv}$ security of HIB-ME. We show an adversary $\mathcal{B}$ that breaks the IND-hID-CPA security of HIBE by using $\mathcal{A}$. The description of $\mathcal{B}$ is as follows.

1. Upon receiving $\mathsf{mpk}_{\mathsf{HIBE}}$, $\mathcal{B}$ generates $(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{msk}_{\mathsf{HIBS}}) \leftarrow \mathsf{Setup}_{\mathsf{HIBS}}(1^\lambda, k)$, picks coin $\hat{b} \leftarrow\!\!\$\ \{0, 1\}$, and prepares $\mathcal{L}_{\mathsf{ek}}, \mathcal{L}_{\mathsf{dk}} := \emptyset$. Then, $\mathcal{B}$ executes $\mathcal{A}$ on input $\mathsf{mpk} := (\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{mpk}_{\mathsf{HIBS}})$.

2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}$ answers as follows:
   (a) When $\mathcal{A}$ makes a sender key generation (resp., delegation) query on input σ (resp., σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$ and σ), $\mathcal{B}$ runs $\mathsf{ek}_\sigma \leftarrow \mathsf{KGen}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{msk}_{\mathsf{HIBS}}, \sigma)$ (resp., $\mathsf{ek}_{\sigma'|\sigma} \leftarrow \mathsf{KDel}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{ek}_{\sigma'}, \sigma$ where $\mathsf{ek}_{\sigma'}$ extracted from $\mathcal{L}_{\mathsf{ek}}$) and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \mathsf{ek}_\sigma)\}$ (resp., $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma, \mathsf{ek}_{\sigma'|\sigma})\}$).
   (b) When $\mathcal{A}$ makes a receiver key generation (resp., delegation) query on input ρ (resp., ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$) and ρ), $\mathcal{B}$ queries ρ (resp., ρ' and ρ) to its challegner and updates $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \perp)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho, \perp)\}$).
   (c) When $\mathcal{A}$ makes a sender key reveal query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$, $\mathcal{B}$ extracts ek_σ from $\mathcal{L}_{\mathsf{ek}}$, and returns ek_σ to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \mathsf{ek}_\sigma)\}$.
   (d) When $\mathcal{A}$ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$, $\mathcal{B}$ queries to its challenger to obtain dk_ρ and returns it to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \perp)\}$.
   (e) When $\mathcal{A}$ makes challenge ciphertexct query on input (σ_0^*, σ_1^*), $(\mathsf{rcv}_0^*, \mathsf{rcv}_1^*)$, and $(\mathsf{m}_0^*, \mathsf{m}_1^*)$, $\mathcal{B}$ computes $\Sigma \leftarrow \mathsf{Sign}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{ek}_{\sigma_{\hat{b}}^*}, \mathsf{m}_{\hat{b}}^* || \mathsf{rcv}^*)$ and makes a challenge ciphertext query on input $(\mathsf{rcv}_0^*|\sigma_{\hat{b}}^*, \mathsf{rcv}_1^*|\sigma_{\hat{b}}^*)$ and
   $(\mathsf{m}_0^*||\Sigma, \mathsf{m}_1^*||\Sigma)$ to obtain ct^*, and returns it to $\mathcal{A}$.
3. Finally, when $\mathcal{A}$ outputs b', $\mathcal{B}$ outputs the same.

From above construction, $\mathcal{B}$ perfectly simulates the hib-cpa-priv game against $\mathcal{A}$. Moreover, since $(\mathsf{rcv}_0^*, \cdot) \notin \mathcal{L}_{\mathsf{dk}} \wedge (\mathsf{rcv}_1^*, \cdot) \notin \mathcal{L}_{\mathsf{dk}}$ implies $(\mathsf{rcv}^*, \cdot) \notin \mathcal{L}_{\mathsf{sk}} \wedge (\mathsf{rcv}_1^*, \cdot) \notin \mathcal{L}_{\mathsf{sk}}$, $\mathcal{B}$ is admissible. Therefore, if $\mathcal{A}$ breaks the hib-cpa-priv security, $\mathcal{B}$ also breaks the IND-hID-CPA security, that is,

$$\mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{ind\text{-}hid\text{-}cpa}}_{\mathsf{HIBE},\mathcal{B}}(\lambda, l).$$

□

Theorem 2. *Suppose that the HIBS scheme* HIBS *is* EUF-hID-CMA *secure. If there exists an adversary* $\mathcal{A}$ *that breaks* hib-auth *security of* HIB-ME*, there exists an adversary* $\mathcal{B}$ *that breaks* EUF-hID-CMA *security of* HIBS *such that*

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{euf\text{-}hid\text{-}cma}}_{\mathsf{HIBS},\mathcal{B}}(\lambda, k)$$

where k *(resp.,* l*) is the maximum depth of senders (resp., receivers). The running time of* $\mathcal{B}$ *is about* $\mathcal{A}$*.*

Proof. Let $\mathcal{A}$ be an adversary that breaks the hib-cpa-priv security of HIB-ME. We show an adversary $\mathcal{B}$ that breaks the EUF-hID-CMA security of HIBS by using $\mathcal{A}$. The description of $\mathcal{B}$ is as follows.

1. Upon receiving $\mathsf{mpk}_{\mathsf{HIBS}}$, $\mathcal{B}$ generates $(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{msk}_{\mathsf{HIBE}}) \leftarrow \mathsf{Setup}_{\mathsf{HIBE}}(1^\lambda, l+1)$ and prepares $\mathcal{L}_{\mathsf{ek}}, \mathcal{L}_{\mathsf{dk}}, \mathcal{L}_{\mathsf{ct}} \coloneqq \emptyset$. Then, $\mathcal{B}$ executes $\mathcal{A}$ on input $\mathsf{mpk} \coloneqq (\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{mpk}_{\mathsf{HIBS}})$.
2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}$ answers as follows:
   (a) When $\mathcal{A}$ makes a sender key generation (resp., delegation) query on input σ (resp., σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$) and σ), $\mathcal{B}$ queries σ (resp., σ' and σ) to its challenger and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \perp)\}$ (resp., $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma, \perp)\}$).
   (b) When $\mathcal{A}$ makes a receiver key generation (resp., delegation) query on input ρ (resp., ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$ and ρ), $\mathcal{B}$ runs $\mathsf{dk}_\rho \leftarrow \mathsf{KGen}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{msk}_{\mathsf{HIBE}}, \rho)$ (resp., $\mathsf{dk}_{\rho'|\rho} \leftarrow \mathsf{KDel}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{dk}_{\rho'}, \rho)$ where $\mathsf{dk}_{\rho'}$ extracted from $\mathcal{L}_{\mathsf{ek}}$) and updates $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \mathsf{dk}_\rho)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho, \mathsf{dk}_{\rho'|\rho})\}$).
   (c) When $\mathcal{A}$ makes a sender key reveal query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$, $\mathcal{B}$ extracts ek_σ from $\mathcal{L}_{\mathsf{ek}}$, and returns ek_σ to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \perp)\}$.
   (d) When $\mathcal{A}$ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$, $\mathcal{B}$ queries to its challenger to obtain dk_ρ and returns it to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \mathsf{dk}_\rho)\}$.
   (e) When $\mathcal{A}$ makes encryption query on input σ, rcv, and m, $\mathcal{B}$ queries σ, rcv, and m|rcv and obtains Σ. Then, $\mathcal{B}$ computes $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{rcv}|\sigma, \mathsf{m}|\Sigma)$ and returns ct to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ct}} \coloneqq \mathcal{L}_{\mathsf{ct}} \cup \{(\sigma, \mathsf{rcv}, \mathsf{m}, \mathsf{ct})\}$.

3. Finally, when $\mathcal{A}$ outputs snd^*, ρ^*, and ct, $\mathcal{B}$ extracts dk_{ρ^*} from $\mathcal{L}_{\mathsf{dk}}$, computes $\mathsf{m}^*|\Sigma^* \leftarrow \mathsf{Dec}_{\mathsf{HIBE}}(\mathsf{mpk}_{\mathsf{HIBE}}, \mathsf{dk}_{\rho^*}, \mathsf{ct})$. If $\mathsf{Ver}_{\mathsf{HIBS}}(\mathsf{mpk}_{\mathsf{HIBS}}, \mathsf{snd}^*, \mathsf{m}^*|\mathsf{rcv}^*, \Sigma^*) = 1$, $\mathcal{B}$ submits snd^*, $\mathsf{m}^*|\mathsf{rcv}^*$, Σ^* as forgery to its challenger. Otherwise, $\mathcal{B}$ halts.

From above construction, $\mathcal{B}$ perfectly simulates the hib-auth game against $\mathcal{A}$. Moreover, since $(\mathsf{snd}^*, \cdot) \in \mathcal{L}_{\mathsf{ek}}$ implies $(\mathsf{snd}^*, \cdot) \neq \mathcal{L}_{\mathsf{sk}}$ and $(\mathsf{snd}^*, \rho^*, \mathsf{m}^*, \cdot) \notin \mathcal{L}_{\mathsf{ct}}$ implies $(\mathsf{snd}^*, \mathsf{m}^*|\rho^*, \Sigma^*) \notin \mathcal{L}_{\Sigma}$, $\mathcal{B}$ is admissible. Therefore, if $\mathcal{A}$ breaks the hib-auth security, $\mathcal{B}$ also breaks the EUF-hID-CMA security, that is,

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{euf\text{-}hid\text{-}cma}}_{\mathsf{HIBS},\mathcal{B}}(\lambda, k).$$

□

Remark 1. We can obtain mismatch case privacy using computational reusable extractor using the same techniques of [FGRV21, BL23, CHHS23].

4 Applications of HIB-ME

In this section, we show some applications of HIB-ME. In Section 4.1, we show how to achieve a full-fledged CCA secure (H)IB-ME scheme by employing CHK transformation [CHK04]. In Section 4.2, we remove the ciphertext overhead of the one-time signature and prove that our tweaked CHK transformation uplifts CPA secure IB-ME to tweaked CCA secure IB-ME using authenticity.

4.1 Construction with Native CHK paradigm

In this section, we give the formal description of our CCA secure (k, l)-level HIB-ME scheme from an $(k, l+1)$-level HIB-ME scheme and a strong one-time signature scheme. Roughly, toward CCA security, we extend the technique by Canetti et al. [CHK04].

Construction. Fix integers $k \geq 1$ and $l \geq 2$. Let $\mathsf{HIB\text{-}ME}' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{SKDel}', \mathsf{RKGen}', \mathsf{RKDel}', \mathsf{Enc}', \mathsf{Dec}')$ 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 $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ be a strong one-time signature scheme with a verification key space $\mathcal{ID}$. Then, our (k, l)-level HIB-ME scheme $\mathsf{HIB\text{-}ME} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{SKDel}, \mathsf{RKGen}, \mathsf{RKDel}, \mathsf{Enc}, \mathsf{Dec})$ is described as follows:

$\mathsf{Setup}(1^\lambda, k, l)$**:** It runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}'(1^\lambda, k, l+1)$ and outputs $(\mathsf{mpk}, \mathsf{msk})$.
$\mathsf{SKGen}(\mathsf{mpk}, \mathsf{msk}, \sigma)$**:** It runs $\mathsf{ek}_\sigma \leftarrow \mathsf{SKGen}'(\mathsf{mpk}, \mathsf{msk}, \sigma)$ and outputs ek_σ.
$\mathsf{SKDel}(\mathsf{mpk}, \mathsf{sk}_{\sigma'}, \sigma)$**:** It runs $\mathsf{ek}_{\sigma'|\sigma} \leftarrow \mathsf{SKDel}'(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$ and outputs $\mathsf{ek}_{\sigma'|\sigma}$.
$\mathsf{RKGen}(\mathsf{mpk}, \mathsf{msk}, \rho)$**:** It runs $\mathsf{dk}_\rho \leftarrow \mathsf{RKGen}'(\mathsf{mpk}, \mathsf{msk}, 0.\rho)$ and outputs dk_ρ.
$\mathsf{RKDel}(\mathsf{mpk}, \mathsf{dk}_{\rho'}, \rho)$**:** It runs $\mathsf{dk}_{\rho'|\rho} \leftarrow \mathsf{RKDel}'(\mathsf{mpk}, \mathsf{dk}_{\rho'}, 0.\rho)$ and outputs $\mathsf{dk}_{\rho'|\rho}$.
$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{m})$**:** It runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$ and sets $\mathsf{rcv}' := 1.\mathsf{vk}|0.\mathsf{rcv}$. Then, it runs $\mathsf{ctxt} \leftarrow \mathsf{Enc}'(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}', \mathsf{m})$ and $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{ctxt})$, and outputs $\mathsf{ct} := (\mathsf{ctxt}, \mathsf{vk}, \Sigma)$.
$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{m})$**:** It checks whether $0 = \mathsf{Ver}(\mathsf{vk}, \mathsf{ctxt}, \Sigma)$ holds. If this is the case, then it returns $\perp$. Otherwise, it generates $\mathsf{dk}_{0.\rho|1.\mathsf{vk}} \leftarrow \mathsf{RKDel}'(\mathsf{mpk}, \mathsf{dk}_\rho, 1.\mathsf{vk})$. Finally, it runs $\mathsf{m} \leftarrow \mathsf{Dec}'(\mathsf{mpk}, \mathsf{dk}_{0.\rho|1.\mathsf{vk}}, \mathsf{snd}, \mathsf{ctxt})$ and outputs the plaintext m.

Correctness. If the ciphertexct $\mathsf{ct} = (\mathsf{ctxt}, \mathsf{vk}, \Sigma)$ is generated correctly, $\mathsf{Ver}(\mathsf{vk}, \mathsf{ctxt}, \Sigma)$ always outputs 1 due to the correctness of Sig. Hence, if $\mathsf{HIB\text{-}ME}'$ satisfies correctness, HIB-ME also satisfies correctness.

Security. Here, we show that our scheme satisfies security requirements.

Theorem 3. *Suppose that the HIBME scheme* $\mathsf{HIB\text{-}ME}'$ *is* hib-cpa-priv *secure and the Signature scheme* Sig *is* sEUF-CMA *secure. If there exists an adversary* $\mathcal{A}$ *that breaks the* hib-cca-priv *security of* HIB-ME*, there exists an adversary* $\mathcal{B}_1$ *that breaks the* sEUF-CMA *security of* Sig*,* $\mathcal{B}_2$ *that breaks* hib-cpa-priv *security of* $\mathsf{HIB\text{-}ME}'$ *such that*

$$\mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) \leq \mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_2}(\lambda, k, l+1),$$

where k (resp, l) is the maximum depth of senders (resp., receivers). The running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Proof. To prove the theorem, we consider the following sequence of games Game_i for $i \in \{0, \ldots, 3\}$ and define

$$\epsilon_i := \Pr\left[b = b' \;\middle|\; \mathsf{Game}_i^{\mathcal{A}}(\lambda, k, l)\right].$$

Game_0 : This is the original hib-cca-priv security game. By definition, we have

$$\left|\epsilon_0 - \frac{1}{2}\right| := \mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l).$$

Game_1: In previous game, the challenger picks $(\mathsf{vk}^*, \mathsf{sk}^*)$ in the challenge phase. But in this game, the challenger runs $(\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KGen}(1^\lambda)$ in the setup phase. Since there is no difference from $\mathcal{A}$'s viewpoint, we have

$$\epsilon_1 = \epsilon_0.$$

Game_2: In this game, we change the behavior of the pre-challenge decryption oracle. Specifically, when $\mathcal{A}$ makes a decryption query on input $(\mathsf{snd}, \rho, \mathsf{ct} = (\mathsf{ctxt}, \mathsf{vk}, \Sigma))$, the challenger always returns $\perp$ to $\mathcal{A}$ if $\mathsf{vk} = \mathsf{vk}^*$.

Let Forge be $\mathcal{A}$ sends valid signature Σ such that $\mathsf{Ver}(\mathsf{vk}^*, \Sigma^*, \mathsf{m})$ to decryption oracle. The two games become different if Forge occurs. Thus, $|\epsilon_2 - \epsilon_1| \leq \Pr[\mathsf{Forge}]$.

To estimate $\Pr[\mathsf{Forge}]$, we show that if $\mathcal{A}$ triggers the event Forge, we can construct an adversary $\mathcal{B}_1$ that breaks the sEUF-CMA security of Sig. The construction of $\mathcal{B}_1$ is as follows.

1. Upon receiving vk^*, $\mathcal{B}_1$ runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}'(1^\lambda, .k, l+1)$. Next, $\mathcal{B}_1$ sets $\mathcal{L}_{\mathsf{ek}}, \mathcal{L}_{\mathsf{dk}} := \emptyset$ and picks $\hat{b} \leftarrow \{0,1\}$. Then, $\mathcal{B}_1$ executes $\mathcal{A}$ on input mpk.
2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}_1$ answers as follows:
   (a) When $\mathcal{A}$ makes sender (resp., receiver) key generation query on input σ (resp., ρ), $\mathcal{B}_1$ runs $\mathsf{ek}_\sigma \leftarrow \mathsf{SKGen}'(\mathsf{mpk}, \mathsf{msk}, \sigma)$ (resp., $\mathsf{dk}_\rho \leftarrow \mathsf{RKGen}'(\mathsf{mpk}, \mathsf{msk}, 0.\rho)$) and updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \mathsf{ek}_\sigma)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \mathsf{dk}_\rho)\}$).
   (b) When $\mathcal{A}$ makes sender (resp., receiver) key delegation query on input σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$ (resp., ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$) and σ (resp., ρ), $\mathcal{B}_1$ extracts $\mathsf{ek}_{\sigma'}$ from $\mathcal{L}_{\mathsf{ek}}$ (resp., $\mathsf{dk}_{\rho'}$ from $\mathcal{L}_{\mathsf{dk}}$), runs $\mathsf{ek}_{\sigma'|\sigma} \leftarrow \mathsf{SKDel}'(\mathsf{mpk}, \mathsf{ek}_{\sigma'}, \sigma)$ (resp., $\mathsf{dk}_{\rho'|\rho} \leftarrow \mathsf{RKDel}'(\mathsf{mpk}, \mathsf{dk}_{\rho'}, 0.\rho)$, and updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \cup \left\{(\sigma'|\sigma, \mathsf{ek}_{\sigma'|\sigma})\right\}$ (resp., $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \cup \left\{(\rho'|\rho), \mathsf{dk}_{\rho'|\rho}\right\}$).
   (c) When $\mathcal{A}$ makes sender (resp., receiver) key reveal query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$ (resp., ρ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$), $\mathcal{B}_1$ extracts ek_σ from $\mathcal{L}_{\mathsf{ek}}$ (resp., dk_ρ from $\mathcal{L}_{\mathsf{dk}}$), returns ek_σ (resp., dk_ρ to $\mathcal{A}$, and updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \mathsf{ek}_\sigma)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \mathsf{dk}_\rho)\}$).
   (d) When $\mathcal{A}$ makes decryption query on input $(\mathsf{snd}, \rho, \mathsf{ct} := (\mathsf{ctxt}, \mathsf{vk}, \Sigma))$, $\mathcal{B}_1$ works as follows:
      – if $\mathsf{vk} = \mathsf{vk}^*$: $\mathcal{B}_1$ checks whether $\mathsf{Ver}(\mathsf{vk}^*, \Sigma, \mathsf{ctxt}) = 1$ or not. If so, $\mathcal{B}_1$ outputs (Σ, ctxt) as forgely and halts. Otherwise, $\mathcal{B}_1$ returns $\perp$.
      – if $\mathsf{vk} \neq \mathsf{vk}^*$: $\mathcal{B}_1$ checks whether $\mathsf{Ver}(\mathsf{vk}^*, \Sigma, \mathsf{ctxt}) = 1$ or not. If so, $\mathcal{B}_1$ extracts dk_ρ from $\mathcal{L}_{\mathsf{dk}}$, runs $\mathsf{dk}_{0.\rho|1.\mathsf{vk}} \leftarrow \mathsf{RKDel}(\mathsf{mpk}, \mathsf{dk}_\rho, 1.\mathsf{vk})$, $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{0.\rho|1.\mathsf{vk}}, \mathsf{snd}, \mathsf{ctxt})$, and returns m to $\mathcal{A}$. Otherwise, $\mathcal{B}_1$ returns $\perp$.
   (e) When $\mathcal{A}$ makes challenge query on input (σ_0^*, σ_1^*), $(\mathsf{rcv}_0^*, \mathsf{rcv}_1^*)$, $\mathsf{m}_0^*, \mathsf{m}_1^*$, $\mathcal{B}_1$ extracts ek_{σ^*} from $\mathcal{L}_{\mathsf{ek}}$, and runs $\mathsf{ctxt}_{\hat{b}}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_{\sigma_{\hat{b}}^*}, 0.\mathsf{rcv}_{\hat{b}}^*|1.\mathsf{vk}^*, \mathsf{m}^*)$. Next, $\mathcal{B}_1$ sends $\mathsf{ctxt}_{\hat{b}}$ to its challenger and obtains Σ^*. Then, $\mathcal{B}_1$ returns $\mathsf{ct}^* = (\mathsf{ctxt}_{\hat{b}}^*, \mathsf{vk}^*, \Sigma^*)$ to $\mathcal{A}$.
3. Finally, when $\mathcal{A}$ outputs b', $\mathcal{B}_1$ halts.

From the above construction, $\mathcal{B}_1$ perfectly simulates hib-cca-priv game for $\mathcal{A}$. Since $(\mathsf{ctxt}_{\hat{b}}^*, \Sigma' *) \neq (\mathsf{ctxt}, \Sigma)$ holds from the requirements for decryption queries by $\mathcal{A}$, the tuple of a message and a signature (ctxt, Σ) output by $\mathcal{B}_1$ satisfies the winning conditions of the experiment sEUF-CMA game. Therefore, we have $\Pr[\mathsf{Forge}] = \mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda)$, i.e.,

$$|\epsilon_2 - \epsilon_1| = \mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda).$$

Finally, to estimate $\epsilon_2 = \frac{1}{2}$, we construct an adversary $\mathcal{B}_2$ that breaks the hib-cpa-priv of HIB-ME$'$ using an adversary $\mathcal{A}$ that breaks Game_2. The construction of $\mathcal{B}_2$ is as follows.

1. Upon receiving mpk, $\mathcal{B}_2$ runs $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{KGen}(1^\lambda)$. Next, $\mathcal{B}_2$ sets $\mathcal{L}_{\mathsf{ek}}, \mathcal{L}_{\mathsf{dk}} := \emptyset$ and picks $\hat{b} \leftarrow \{0,1\}$. Then, $\mathcal{B}$ executes $\mathcal{A}$ on input mpk.
2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}_2$ answers as follows:
   (a) When $\mathcal{A}$ makes a sender (resp., receiver) key generation query on input σ (resp., ρ), $\mathcal{B}_2$ makes a sender (resp., receiver) key generation query to its challenger on input σ (resp., ρ), and updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \perp)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \perp)\}$).
   (b) When $\mathcal{A}$ makes a sender (resp., receiver) key delegation query on input σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$ and σ (resp., ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$ and ρ), $\mathcal{B}_2$ makes a sender (resp., receiver) key delegation query to its challenger on input σ' and σ (resp., $0.\rho'$ and $0.\rho$), and updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma, \perp)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho, \perp)\}$).
   (c) When $\mathcal{A}$ makes a sender (resp., receiver) key reveal query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$ (resp., ρ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$), $\mathcal{B}_2$ makes a sender (resp., receiver) key reveal query to its challenger on input σ (resp., ρ), receives ek_σ (resp., dk_ρ), and returns as it is to $\mathcal{A}$. In addition, $\mathcal{B}_2$ updates $\mathcal{L}_{\mathsf{ek}} := \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \perp)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} := \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \perp)\}$).
   (d) When $\mathcal{A}$ makes decryption query on input snd, ρ and $\mathsf{ct} = (\mathsf{ctxt}, \mathsf{vk}, \Sigma)$, $\mathcal{B}_2$ returns $\perp$ if $\mathsf{vk} = \mathsf{vk}^*$ or $\mathsf{Ver}(\mathsf{vk}, \Sigma, \mathsf{ctxt}) = 0$. Otherwise, $\mathcal{B}_2$ makes a receiver key delegation query on input $0.\rho$ and $1.\mathsf{vk}$, and a receiver key reveal query on input $0.\rho|1.\mathsf{vk}$ to receive $\mathsf{dk}_{0.\rho|1.\mathsf{vk}}$. Then, $\mathcal{B}$ runs $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{snd}, \mathsf{dk}_{0.\rho|1.\mathsf{vk}}, \mathsf{ctxt})$ and returns m to $\mathcal{A}$.
3. When $\mathcal{A}$ makes the challenge ciphertext query on input (σ_0^*, σ_1^*), $(\mathsf{rcv}_0^*, \mathsf{rcv}_1^*)$ and $(\mathsf{m}_0^*, \mathsf{m}_1^*)$, $\mathcal{B}_2$ sets $\mathsf{rcv}_0' := 0.\mathsf{rcv}_0^*|1.\mathsf{vk}^*$ and $\mathsf{rcv}_1' := 0.\mathsf{rcv}_1^*|1.\mathsf{vk}^*$, and makes the challenge ciphertext query on input (σ_0^*, σ_1^*), (ρ_0', ρ_1') and $(\mathsf{m}_0^*, \mathsf{m}_1^*)$ to receives ctxt^*. Then, $\mathcal{B}_2$ runs $\Sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}^*, \mathsf{ctxt}^*)$ and returns $\mathsf{ct}^* := (\mathsf{ctxt}^*, \mathsf{vk}^*, \Sigma^*)$ to $\mathcal{A}$.
4. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}_2$ answers as follows. (Except decryption oracle query, $\mathcal{B}_2$ behaves the same way as in the pre-challenge query.)
   (a) When $\mathcal{A}$ makes a decryption query on input snd, ρ and ct, $\mathcal{B}_2$ returns $\perp$ if $\mathsf{vk} = \mathsf{vk}^*$. Otherwise, $\mathcal{B}_2$ makes a receiver key delegation query on input $0.\rho$ and $1.\mathsf{vk}$, and a receiver key reveal query on input $0.\rho|1.\mathsf{vk}$, and receives $\mathsf{dk}_{0.\rho|1.\mathsf{vk}}$ [4]. Then, $\mathcal{B}_2$ runs $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{0.\rho|1.\mathsf{vk}}, \mathsf{snd}, \mathsf{ctxt})$ and returns m to $\mathcal{A}$.
5. Finally, when $\mathcal{A}$ outputs b', $\mathcal{B}_2$ outputs the same way.

From above construction, $\mathcal{B}_2$ perfectly simulates Game_2 for $\mathcal{A}$ and the challenge bits of $\mathcal{B}_2$ and $\mathcal{A}$ correspond. Therefore, we have

$$\left|\epsilon_2 - \frac{1}{2}\right| = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_2}(\lambda, k, l+1).$$

Putting everything together, we have

$$\mathsf{Adv}^{\mathsf{hib\text{-}cca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{Sig},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_2}(\lambda, k, l+1).$$

□

Theorem 4. *Suppose that the HIB-ME scheme* HIB-ME$'$ *is* hib-auth *secure. If there exists an adversary* $\mathcal{A}$ *that breaks the* hib-auth *security of* HIB-ME*, there exists an adversary* $\mathcal{B}$ *that breaks the* hib-auth *security of* HIB-ME$'$ *such that*

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}}(\lambda, k, l+1).$$

where k *(resp.,* l*) is the maximum depth of senders (resp., receivers). The running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

[4] If $\mathsf{vk} \neq \mathsf{vk}^*$, since $0.\mathsf{rcv}_0^*|1.\mathsf{vk}$ (resp., $0.\mathsf{rcv}_1^*|1.\mathsf{vk}$) is different from $0.\mathsf{rcv}_0^*|1.\mathsf{vk}^*$ (resp., $0.\mathsf{rcv}_1^*|1.\mathsf{vk}$) and it do not leaks about $0.\mathsf{rcv}_0^*|1.\mathsf{vk}^*$ (resp., $0.\mathsf{rcv}_1^*|1.\mathsf{vk}$), $\mathcal{B}_2$ can query $0.\mathsf{rcv}_0^*|1.\mathsf{vk}$ (resp., $0.\mathsf{rcv}_1^*|1.\mathsf{vk}$) to receiver key reveal query.

Proof. To prove the theorem, we construct an adversary $\mathcal{B}$ that breaks hib-auth of HIB-ME$'$ using $\mathcal{A}$ that breaks hib-auth of HIB-ME. The construction of $\mathcal{B}$ is as follows.

1. Upon receiving mpk, $\mathcal{B}$ executes $\mathcal{A}$ on input mpk.
2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}$ answers as follows.
   (a) When $\mathcal{A}$ makes a sender (resp., receiver) key generation query on input σ (resp., ρ), $\mathcal{B}$ makes a sender (resp., receiver) key generation query to its challenger on input σ (resp., ρ), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \bot)\}$).
   (b) When $\mathcal{A}$ makes a sender (resp., receiver) key delegation query on input σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$ and σ (resp., ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$ and ρ), $\mathcal{B}$ makes a sender (resp., receiver) key delegation query to its challenger on input σ' and σ (resp., $0.\rho'$ and $0.\rho$), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho, \bot)\}$).
   (c) When $\mathcal{A}$ makes a sender (resp., receiver) key reveal query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$ (resp., ρ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$), $\mathcal{B}$ makes a sender (resp., receiver) key reveal query to its challenger on input σ (resp., ρ), receives ek_σ (resp., dk_ρ), and returns as it is to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \bot)\}$).
   (d) When $\mathcal{A}$ makes a encryption query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$, rcv such that $(\mathsf{rcv}, \cdot) \in \mathcal{L}_{\mathsf{dk}}$, and message m, $\mathcal{B}$ runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$ and sets $\mathsf{rcv}' = 0.\mathsf{rcv}|1.\mathsf{vk}$. Next, $\mathcal{B}$ sends a encryption query on input σ, rcv', and m to obtain ctxt. Then, $\mathcal{B}$ computes $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{ctxt})$ and returns $\mathsf{ct} = (\mathsf{ctxt}, \mathsf{vk}, \Sigma)$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ct}} \coloneqq \mathcal{L}_{\mathsf{ct}} \cup \{(\sigma, \mathsf{rcv}, \mathsf{m}, \mathsf{ct})\}$.
3. Finally, when $\mathcal{A}$ submits $(\mathsf{snd}^*, \rho^*, \mathsf{ct}^* = (\mathsf{ctxt}^*, \mathsf{vk}^*, \Sigma^*))$ as a forgery, $\mathcal{B}$ submits the same $(\mathsf{snd}^*, 0.\rho^*|1.\mathsf{vk}, \mathsf{ctxt}^*)$ if $\mathsf{Ver}(\mathsf{vk}, \Sigma^*, \mathsf{ctxt}^*) = 1$.

We can verify that $\mathcal{B}$ perfectly simulates the hib-auth game. Therefore, if $\mathcal{A}$ breaks the hib-auth security of HIB-ME, $\mathcal{B}$ also breaks the hib-auth security of HIB-ME$'$. Thus, we have

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}}(\lambda, k, l+1).$$

□

4.2 Construction with Tweaked CHK Paradigm

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 $k \geq 1$ and $l \geq 2$. Let HIB-ME$' = (\mathsf{Setup}', \mathsf{SKGen}', \mathsf{SKDel}', \mathsf{RKGen}', \mathsf{RKDel}', \mathsf{Enc}', \mathsf{Dec}')$ 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 HIB-ME $= (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{SKDel}, \mathsf{RKGen}, \mathsf{RKDel}, \mathsf{Enc}, \mathsf{Dec})$. Setup algorithm Setup, sender key generation algorithm SKGen, sender key delegation algorithm SKDel, receiver key generation algorithm RKGen, and receiver key delegation algorithm RKDel is the same as construction in section 4.1. Now, we show encryption algorithm Enc and decryption algorithm Dec as follows:

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}, \mathsf{m})$**:** It sets $\mathsf{rcv}' \coloneqq 0.\mathsf{rcv}|1.\mathsf{snd}$, runs $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}', \mathsf{m})$, and outputs ct.
$\mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_\rho, \mathsf{snd}, \mathsf{ct})$**:** It runs $\mathsf{dk}_{0.\rho|1.\mathsf{snd}} \leftarrow \mathsf{RKDel}(\mathsf{mpk}, \mathsf{dk}_\rho, 1.\mathsf{snd})$ and $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{0.\rho|1.\mathsf{snd}}, \mathsf{snd}, \mathsf{m})$, and outputs m.

Correctness. Obviously, if HIB-ME$'$ satisfies correctness, HIB-ME satisfies correctness.

Security. Here, we show that our scheme satisfies seurity requirements.

Theorem 5. *Suppose that the HIB-ME scheme* $\mathsf{HIB\text{-}ME}'$ *satisfies* $\mathsf{hib\text{-}cpa\text{-}priv}$ *security and* $\mathsf{hib\text{-}auth}$ *security. If there exists an adversary* $\mathcal{A}$ *that breaks the* $\mathsf{hib\text{-}tcca\text{-}priv}$ *security of* $\mathsf{HIB\text{-}ME}$*, there exists an adversary* $\mathcal{B}_1$ *that breaks the* $\mathsf{hib\text{-}cpa\text{-}priv}$ *security of* $\mathsf{HIB\text{-}ME}'$ *and* $\mathcal{B}_2$ *that breaks the* $\mathsf{hib\text{-}auth}$ *security of* $\mathsf{HIB\text{-}ME}'$ *such that*

$$\mathsf{Adv}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda,k,l) \leq \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_1}(\lambda,k,l+1) + \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_2}(\lambda,k,l+1)$$

where k *(resp.,* l*) is the maximum depth of senders (resp., receivers). The running time of* $\mathcal{B}$ *is about of* $\mathcal{A}$*.*

Proof. To prove the theorem, we consider the following sequence of games Game_i for $i \in \{0,1\}$ and define

$$\epsilon_i \coloneqq \Pr\left[b = b' \;\middle|\; \mathsf{Game}_i^{\mathcal{A}}(\lambda,k,l)\right].$$

Game_0: This is the originial $\mathsf{hib\text{-}tcca\text{-}priv}$ secuirty game. By definition, we have

$$\left|\epsilon_0 - \frac{1}{2}\right| \coloneqq \mathsf{Adv}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda,k,l).$$

Game_1: Same as Game_0, except that, when $\mathcal{A}$ makes a decryption query $(\rho,\mathsf{snd},\mathsf{ct})$, the challenger returns as

$$\begin{cases} \mathsf{m} & \text{if } (\mathsf{snd},\rho,\mathsf{m},\mathsf{ct}) \in \mathcal{L}_{\mathsf{ct}} \\ \mathsf{Dec}(\mathsf{mpk},\mathsf{dk}_{0.\rho|1.\mathsf{snd}},\mathsf{snd}.\mathsf{ct}) & \text{if } \mathsf{snd} \neq \sigma^* \vee \rho \neq \mathsf{rcv}^* \\ \bot & \text{othweise} \end{cases}$$

to $\mathcal{A}$.

Let Forge be $\mathcal{A}$ sends valid ciphertext tuple snd, such that $\mathsf{snd} \in \mathcal{L}_{\mathsf{ek}}$, ρ such that $\rho \in \mathcal{L}_{\mathsf{dk}}$, ct such that $\mathsf{Dec}(\mathsf{mpk},\mathsf{dk}_\rho,\mathsf{snd},\mathsf{ct}) \neq \bot \wedge (\mathsf{snd},\mathsf{rcv},\cdot,\mathsf{ct}) \notin \mathcal{L}_{\mathsf{ct}}$ to decryption query. The two games become different if Forge occurs. Thus $|\epsilon_1 - \epsilon_0| = \Pr[\mathsf{Forge}]$.

To estimate $\Pr[\mathsf{Forge}]$, we show that if $\mathcal{A}$ triggers the event Forge, we can construct an adversary $\mathcal{B}_2$ that breaks $\mathsf{hib\text{-}auth}$ security of $\mathsf{HIB\text{-}ME}'$. The construction of $\mathcal{B}_2$ is follows.

1. Upon receiving mpk, $\mathcal{B}_2$ sets $\mathcal{L}_{\mathsf{ek}},\mathcal{L}_{\mathsf{dk}},\mathcal{L}_{\mathsf{ct}} \coloneqq \emptyset$ and picks $\hat{b} \leftarrow\!\!\$\ \{0,1\}$. Then, $\mathcal{B}$ executes $\mathcal{A}$ on inputs mpk.
2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}_2$ answers as follows.
   (a) When $\mathcal{A}$ makes a sender (resp., receiver) key generation query on input σ (resp., ρ), $\mathcal{B}_2$ makes a sender (resp., receiver) key generation query to its challenger on input σ (resp., ρ), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma,\bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho,\bot)\}$).
   (b) When $\mathcal{A}$ makes a sender (resp., receiver) key delegation query on input σ' such that $(\sigma',\cdot) \in \mathcal{L}_{\mathsf{ek}}$ and σ (resp., ρ' such that $(\rho',\cdot) \in \mathcal{L}_{\mathsf{dk}}$ and ρ), $\mathcal{B}_2$ makes a sender (resp., receiver) key delegation query to its challenger on input σ' and σ (resp., $0.\rho'$ and $0.\rho$), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma,\bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho,\bot)\}$).
   (c) When $\mathcal{A}$ makes a sender (resp., receiver) key reveal query on input σ such that $(\sigma,\cdot) \in \mathcal{L}_{\mathsf{ek}}$ (resp., ρ such that $(\rho,\cdot) \in \mathcal{L}_{\mathsf{dk}}$), $\mathcal{B}_2$ makes a sender (resp., receiver) key reveal query to its challenger on input σ (resp., ρ), receives ek_σ (resp., dk_ρ), and returns as it is to $\mathcal{A}$. In addition, $\mathcal{B}_2$ updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma,\bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho,\bot)\}$).
   (d) When $\mathcal{A}$ makes a encryption query on input σ such that $(\sigma,\cdot) \in \mathcal{L}_{\mathsf{ek}}$, rcv such that $(\mathsf{rcv},\cdot) \in \mathcal{L}_{\mathsf{dk}}$, and message m, $\mathcal{B}$ sets $\mathsf{rcv}' = 0.\mathsf{rcv}|1.\sigma$. Next, $\mathcal{B}$ sends a encryption query on input σ, rcv' and m to obtain ct, and returns ct. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ct}} \coloneqq \mathcal{L}_{\mathsf{ct}} \cup \{(\sigma,\mathsf{rcv},\mathsf{m},\mathsf{ct})\}$.
   (e) When $\mathcal{A}$ makes a decryption query on input snd such that $(\mathsf{snd},\cdot) \in \mathcal{L}_{\mathsf{ek}}$, ρ such that $(\rho,\cdot) \in \mathcal{L}_{\mathsf{dk}}$, ct, if $(\mathsf{snd},\rho,\cdot,\mathsf{ct}) \in \mathcal{L}_{\mathsf{ct}}$, $\mathcal{B}$ extracts m from $\mathcal{L}_{\mathsf{ct}}$ and returns m. Otherwise, $\mathcal{B}$ obtains $\mathsf{dk}_{0.\rho|1.\mathsf{snd}}$ by making a receiver key reveal query on input $0.\rho|1.\mathsf{snd}$. If $\mathsf{Dec}(\mathsf{mpk},\mathsf{dk}_{0.\rho|1.\mathsf{snd}},\mathsf{snd},\mathsf{ct}) \neq \bot$, $\mathcal{B}$ outputs $(\mathsf{snd},0.\rho|1.\mathsf{snd},\mathsf{ct})$ as forgery and terminates. Otherwise, $\mathcal{B}$ returns $\bot$ to $\mathcal{A}$.
3. When $\mathcal{A}$ makes the challenge ciphertext query on input (σ_0^*,σ_1^*), $(\mathsf{rcv}_0^*,\mathsf{rcv}_1^*)$, and $(\mathsf{m}_0^*,\mathsf{m}_1^*)$, $\mathcal{B}_2$ sets $\mathsf{rcv}_0' = 0.\mathsf{rcv}_0^*|1.\sigma_0^*$ and $\mathsf{rcv}_1' = 0.\mathsf{rcv}_1^*|1.\sigma_1^*$, makes a encryption query on input (σ_0^*,σ_1^*), $(\mathsf{rcv}_0',\mathsf{rcv}_1')$, and $(\mathsf{m}_0^*,\mathsf{m}_1^*)$ and obtains ct^*. Then, $\mathcal{B}_2$ returns ct^* to $\mathcal{A}$ and updates $\mathcal{L}_{\mathsf{ct}} \coloneqq \mathcal{L}_{\mathsf{ct}} \cup \{(\sigma^*,\mathsf{rcv}^*,\mathsf{m}^*,\mathsf{ct}^*)\}$.

4. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$, $\mathcal{B}_2$ halts.

From the above construction, $\mathcal{B}_2$ perfectly simulates hib-tcca-priv game for $\mathcal{A}$. Since $\mathcal{B}$ can reveal the target receiver's decryption key in hib-auth game, $\mathcal{B}_2$ satisfies the winning conditions of hib-auth game. Therefore, we have $\Pr[\mathsf{Forge}] = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_2}(\lambda, k, l+1)$, i.e.,

$$|\epsilon_1 - \epsilon_0| = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_2}(\lambda, k, l+1).$$

Finally, to estimate $\left|\epsilon_1 - \frac{1}{2}\right|$, we construct an adversary $\mathcal{B}_1$ that breaks the hib-cpa-priv using an adversary $\mathcal{A}$ that breaks Game_1. The construction of $\mathcal{B}_1$ is as follows.

1. Upon receiving mpk, $\mathcal{B}_1$ sets $\mathcal{L}_{\mathsf{ek}}, \mathcal{L}_{\mathsf{dk}}, \mathcal{L}_{\mathsf{ct}} \coloneqq \emptyset$ and executes $\mathcal{A}$ on input mpk.
2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}_1$ answers as follows.
   (a) When $\mathcal{A}$ makes a sender (resp., receiver) key generation query on input σ (resp., ρ), $\mathcal{B}_1$ makes a sender (resp., receiver) key generation query to its challenger on input σ (resp., ρ), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \bot)\}$).
   (b) When $\mathcal{A}$ makes a sender (resp., receiver) key delegation query on input σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$ and σ (resp., ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$ and ρ), $\mathcal{B}_1$ makes a sender (resp., receiver) key delegation query to its challenger on input σ' and σ (resp., $0.\rho'$ and $0.\rho$), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho, \bot)\}$).
   (c) When $\mathcal{A}$ makes a sender (resp., receiver) key reveal query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$ (resp., ρ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$), $\mathcal{B}_1$ makes a sender (resp., receiver) key reveal query to its challenger on input σ (resp., ρ), receives ek_σ (resp., dk_ρ), and returns as it is to $\mathcal{A}$. In addition, $\mathcal{B}_1$ updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \bot)\}$).
   (d) When $\mathcal{A}$ makes a encryption query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$, rcv such that $(\mathsf{rcv}, \cdot) \in \mathcal{L}_{\mathsf{dk}}$, and message m, $\mathcal{B}$ sets $\mathsf{rcv}' = 0.\mathsf{rcv}|1.\sigma$. Next, $\mathcal{B}$ sends a sender key reveal query on input σ to obtain ek_σ. Then, $\mathcal{B}$ computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ek}_\sigma, \mathsf{rcv}', \mathsf{m})$ and returns ct to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ct}} \coloneqq \mathcal{L}_{\mathsf{ct}} \cup \{(\sigma, \mathsf{rcv}, \mathsf{m}, \mathsf{ct})\}$.
   (e) When $\mathcal{A}$ makes a decryption query on input snd, ρ and ct, if $(\mathsf{snd}, \rho, \cdot, \mathsf{ct}) \in \mathcal{L}_{\mathsf{ct}}$, $\mathcal{B}$ extracts m from $\mathcal{L}_{\mathsf{ct}}$ and returns m to $\mathcal{A}$. Otherwise, if $\rho = \mathsf{rcv}^* \wedge \mathsf{snd} \neq \sigma^*$, $\mathcal{B}_1$ returns $\bot$ to $\mathcal{A}$. Otherwise, $\mathcal{B}_1$ makes a receiver key reveal query on $0.\rho|1.\mathsf{snd}$ to obtain $\mathsf{dk}_{0.\rho|1.\mathsf{snd}}$. Then, $\mathcal{B}_1$ computes $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{dk}_{0.\rho|1.\mathsf{snd}}, \mathsf{snd}, \mathsf{ct})$ and returns m to $\mathcal{A}$.
3. When $\mathcal{A}$ makes the challenge ciphertext query on input σ^*, rcv^* and m^*, $\mathcal{B}_1$ sets $\mathsf{rcv}' \coloneqq 0.\mathsf{rcv}^*|1.\mathsf{snd}^*$, makes the challenge query on input σ^*, rcv', m^* and receives ct^*. Then, $\mathcal{B}_1$ returns ct^* to $\mathcal{A}$.
4. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$, $\mathcal{B}_1$ outputs the same guess.

From the above construction, $\mathcal{B}_1$ perfectly simulates Game_1 fro $\mathcal{A}$ and the challenge bits of $\mathcal{B}_1$ and $\mathcal{A}$ correspond. Therefore, we have

$$\left|\epsilon_1 - \frac{1}{2}\right| = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathcal{A}}(\lambda, k, l+1)$$

Putting everything together, we have

$$\mathsf{Adv}^{\mathsf{hib\text{-}tcca\text{-}priv}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{hib\text{-}cpa\text{-}priv}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_1}(\lambda, k, l+1) + \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}_2}(\lambda, k, l+1).$$

□

Theorem 6. *Suppose that the HIB-ME scheme* $\mathsf{HIB\text{-}ME}'$ *satisfies* hib-auth *security. If there exists an adversary* $\mathcal{A}$ *that breaks the* hib-auth *security of* HIB-ME*, there exists an adversary* $\mathcal{B}$ *that breaks the* hib-auth *security of* $\mathsf{HIB\text{-}ME}'$ *such that*

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}}(\lambda, k, l+1)$$

where k *(resp.,* l*) is the maximum depth of senders (resp., receivers). The running time of* $\mathcal{B}$ *is about of* $\mathcal{A}$*.*

Proof. To prove the theorem, we construct an adversary $\mathcal{B}$ that breaks hib-auth of HIB-ME′ using $\mathcal{A}$ that breaks hib-auth of HIB-ME. The construction of $\mathcal{B}$ is as follows.

1. Upon receiving mpk, $\mathcal{B}$ executes $\mathcal{A}$ on input mpk.
2. When $\mathcal{A}$ makes some oracle queries, $\mathcal{B}$ answers as follows.
   (a) When $\mathcal{A}$ makes a sender (resp., receiver) key generation query on input σ (resp., ρ), $\mathcal{B}$ makes a sender (resp., receiver) key generation query to its challenger on input σ (resp., ρ), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho, \bot)\}$).
   (b) When $\mathcal{A}$ makes a sender (resp., receiver) key delegation query on input σ' such that $(\sigma', \cdot) \in \mathcal{L}_{\mathsf{ek}}$ and σ (resp., ρ' such that $(\rho', \cdot) \in \mathcal{L}_{\mathsf{dk}}$ and ρ), $\mathcal{B}$ makes a sender (resp., receiver) key delegation query to its challenger on input σ' and σ (resp., $0.\rho'$ and $0.\rho$), and updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \cup \{(\sigma'|\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \cup \{(\rho'|\rho, \bot)\}$).
   (c) When $\mathcal{A}$ makes a sender (resp., receiver) key reveal query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$ (resp., ρ such that $(\rho, \cdot) \in \mathcal{L}_{\mathsf{dk}}$), $\mathcal{B}$ makes a sender (resp., receiver) key reveal query to its challenger on input σ (resp., ρ), receives ek_σ (resp., dk_ρ), and returns as it is to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ek}} \coloneqq \mathcal{L}_{\mathsf{ek}} \setminus \{(\sigma, \bot)\}$ (resp., $\mathcal{L}_{\mathsf{dk}} \coloneqq \mathcal{L}_{\mathsf{dk}} \setminus \{(\rho, \bot)\}$).
   (d) When $\mathcal{A}$ makes a encryption query on input σ such that $(\sigma, \cdot) \in \mathcal{L}_{\mathsf{ek}}$, rcv such that $(\mathsf{rcv}, \cdot) \in \mathcal{L}_{\mathsf{dk}}$, and message m, $\mathcal{B}$ runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$ and sets $\mathsf{rcv}' = 0.\mathsf{rcv}|1.\mathsf{vk}$. Next, $\mathcal{B}$ sends a encryption query on input σ, rcv', and m to obtain ct. Then, $\mathcal{B}$ computes $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{ct})$ and returns $\mathsf{ct} = (\mathsf{ct}, \mathsf{vk}, \Sigma)$. In addition, $\mathcal{B}$ updates $\mathcal{L}_{\mathsf{ct}} \coloneqq \mathcal{L}_{\mathsf{ct}} \cup \{(\sigma, \mathsf{rcv}, \mathsf{m}, \mathsf{ct})\}$.
3. Finally, when $\mathcal{A}$ submits $(\mathsf{snd}^*, \rho^*, \mathsf{ct}^* = (\mathsf{ct}^*, \mathsf{vk}^*, \Sigma^*))$ as a forgery, $\mathcal{B}$ submits the same $(\mathsf{snd}^*, 0.\rho^*|1.\mathsf{vk}, \mathsf{ct}^*)$ if $\mathsf{Ver}(\mathsf{vk}, \Sigma^*, \mathsf{ct}^*) = 1$.

We can verify that $\mathcal{B}$ perfectly simulates the hib-auth game. Therefore, if $\mathcal{A}$ breaks the hib-auth security of HIB-ME, $\mathcal{B}$ also breaks the hib-auth security of HIB-ME′. Thus, we have

$$\mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME},\mathcal{A}}(\lambda, k, l) = \mathsf{Adv}^{\mathsf{hib\text{-}auth}}_{\mathsf{HIB\text{-}ME}',\mathcal{B}}(\lambda, k, l+1).$$

□

Acknowledgments. The authors would like to thank anonymous reviewers for their constructive and valuable comments. This work was supported by JSPS KAKENHI Grant Number JP23K24846, Japan. Sohto Chiku was partially supported by JSPS KAKENHI Grant Number JP25KJ1319, Japan. Keisuke Hara was partially supported by JSPS KAKENHI Grant Number JP24K20776 and JST-CREST JPMJCR22M1, Japan.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

AFNV19. Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 701–731, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

AFNV21. Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. *Journal of Cryptology*, 34(3):16, July 2021.

BCF24. Roberta Cimorelli Belfiore, Andrea De Cosmo, and Anna Lisa Ferrara. Identity-based matchmaking encryption from standard lattice assumptions. In Christina Pöpper and Lejla Batina, editors, *ACNS 24: 22nd International Conference on Applied Cryptography and Network Security, Part II*, volume 14584 of *Lecture Notes in Computer Science*, pages 163–188, Abu Dhabi, UAE, March 5–8, 2024. Springer, Cham, Switzerland.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

BKP14. Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 408–425, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

BL23. Xavier Boyen and Qinyi Li. Identity-based matchmaking encryption with enhanced privacy — a generic construction with practical instantiations. In *ESORICS 2023*, Heidelberg, September 2023. Springer.

BW06. Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

CHHS23. Sohto Chiku, Keitaro Hashimoto, Keisuke Hara, and Junji Shikata. 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.

CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

CHKP12. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012.

CLWW22. Jie Chen, Yu Li, Jinming Wen, and Jian Weng. Identity-based matchmaking encryption from standard assumptions. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part III*, volume 13793 of *Lecture Notes in Computer Science*, pages 394–422, Taipei, Taiwan, December 5–9, 2022. Springer, Heidelberg, Germany.

DLP14. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 22–41, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

FGRV21. Danilo Francati, Alessio Guidi, Luigi Russo, and Daniele Venturi. Identity-based matchmaking encryption without random oracles. In Avishek Adhikari, Ralf Küsters, and Bart Preneel, editors, *INDOCRYPT 2021*, volume 13143 of *LNCS*, pages 415–435, Heidelberg, December 2021. Springer.

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68, Kamakura, Japan, March 1–3, 1999. Springer, Heidelberg, Germany.

GS02. Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany.

HL02. Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

KN08. Eike Kiltz and G. Neven. *Identity-Based Signatures*, pages 31–44. Cryptology and Information Security Series on Identity-Based Cryptography. I, January 2008.

LLC24. Shen Lin, Yu Li, and Jie Chen. Cca-secure identity-based matchmaking encryption from standard assumptions. In Chunpeng Ge and Moti Yung, editors, *Information Security and Cryptology*, pages 253–273, Singapore, 2024. Springer Nature Singapore.

PFH+22. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

SKOS09. Jae Hong Seo, Tetsutaro Kobayashi, Miyako Ohkubo, and Koutarou Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 215–234, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany.

SOK00. Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, January 2000.

SW08. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 560–578, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

WWLZ25. Yuejun Wang, Baocang Wang, Qiqi Lai, and Yu Zhan. Identity-based matchmaking encryption with stronger security and instantiation on lattices. *Theoretical Computer Science*, 1029:115048, 2025.

Table of Contents