Simulation-Based Bi-Selective Opening Security for Public Key Encryption

Junzuo Lai[1], Rupeng Yang[2], Zhengan Huang[3], and Jian Weng[1]

[1] College of Information Science and Technology, Jinan University, Guangzhou, China
laijunzuo@gmail.com, cryptjweng@gmail.com
[2] Department of Computer Science, The University of Hong Kong, Hong Kong, China
orbbyrp@gmail.com
[3] Peng Cheng Laboratory, Shenzhen, China
zhahuang.sjtu@gmail.com

Abstract. Selective opening attacks (SOA) (for public-key encryption, PKE) concern such a multi-user scenario, where an adversary adaptively corrupts some fraction of the users to break into a subset of honestly created ciphertexts, and tries to learn the information on the messages of some unopened (but potentially related) ciphertexts. Until now, the notion of selective opening attacks is only considered in two settings: sender selective opening (SSO), where part of senders are corrupted and messages together with randomness for encryption are revealed; and receiver selective opening (RSO), where part of receivers are corrupted and messages together with secret keys for decryption are revealed.

In this paper, we consider a more natural and general setting for selective opening security. In the setting, the adversary may adaptively corrupt part of senders and receivers *simultaneously*, and get the plaintext messages together with internal randomness for encryption and secret keys for decryption, while it is hoped that messages of uncorrupted parties remain protected. We denote it as Bi-SO security since it is reminiscent of Bi-Deniability for PKE.

We first formalize the requirement of Bi-SO security by the simulation-based (SIM) style, and prove that some practical PKE schemes achieve SIM-Bi-SO-CCA security in the random oracle model. Then, we suggest a weak model of Bi-SO security, denoted as SIM-wBi-SO-CCA security, and argue that it is still meaningful and useful. We propose a generic construction of PKE schemes that achieve SIM-wBi-SO-CCA security in the standard model and instantiate them from various standard assumptions. Our generic construction is built on a newly presented primitive, namely, universal$_\kappa$ hash proof system with key equivocability, which may be of independent interest.

Keywords: Public Key Encryption, Multi-User Security, Selective Opening Security, Simulation-Based Security, Chosen-Ciphertext Security

1 Introduction

Public key encryption (PKE) is a fundamental tool to protect messages sent over a public channel. Usually, a PKE scheme is used in an open system with multi-users. The system contains multiple, say n, users, each with a public key/secret key pair, i.e., there are n public keys in the system. Anyone (even not registered in the system) can send messages over the public channel to a user securely via encrypting the message under the user's public key. Thus, each public key will be used for multiple, say k, times during the lifetime of the system.

Selective Opening Attacks. Currently, the standard security for PKE schemes is the so-called "Chosen-ciphertext attack (CCA) security", which allows the attacker to learn the decryption of its selected ciphertexts. Generally, PKE schemes are designed to guarantee security of all messages in the system against a CCA attacker under the assumption that internal status of all users are properly protected. This assumption, however, will be challenged in some real-world scenarios:

- The attacker may corrupt the senders and learn their messages and the encryption randomness.
- The attacker may corrupt the receivers and learn their secret keys. With the receivers' secret keys, the attacker is able to decrypt all ciphertexts sent to the receivers and obtain the messages.

While it is hopeless to protect those opened messages, one natural question is whether the unopened messages are still well protected. The above attacks are called selective opening attacks. Surprisingly, it is proved that standard security notion (i.e., CCA security) is *not* able to guarantee security against selective opening attacks (SO security) [2,19,18].

The notion of SO security for PKE was firstly formalized by Bellare et al. [3] at EUROCRYPT 2009. To date, two settings have been considered for SO security: sender corruption [3] and receiver corruption [2]. In the sender corruption setting, part of senders are corrupted, with the corruption exposing their coins and messages. In the receiver corruption setting, part of receivers are corrupted, with corruption exposing their secret keys and messages. We denote SO security in the sender-corruption setting and in the receiver-corruption setting by SSO security and RSO security, respectively.

Furthermore, for each setting, there are two types of definitions for SO security: indistinguishability-based (IND) SO security and simulation-based (SIM) SO security. IND-SO security requires that no efficient ad-

versary can distinguish the uncorrupted users' ciphertexts from the encryption of fresh messages, which are sampled according to a conditional probability distribution (conditioned on the opened ciphertexts, which means the ciphertexts of the corrupted parties). In other words, IND-SO security requires that the considered message distributions should be efficiently conditionally re-samplable [3]. SIM-SO security requires that anything, which can be computed efficiently from the ciphertexts, the opened messages as well as the corrupted information, can also be computed efficiently only with the opened messages. SIM-SO security imposes no limitation on the message distributions.

Motivations. Previous works on SIM-SO-CCA secure PKE schemes only provide *either* sender selective opening security [3,10,17,21,27,15,26,28,22], *or* receiver selective opening security [2,13,24,20,12,33]. However, it is rarely possible to predict whether the attacker will corrupt the senders or the receivers beforehand in practice. Moreover, most of the previous works about RSO security only focused on the single-challenge setting, i.e., each public key can only be used *once* to produce a *single ciphertext*. This is very unrealistic in practice.[4]

Based on the above facts, the following question is raised naturally: *How to define security models to capture the practical requirements of selective opening security in the multi-user scenario, and provide secure PKE schemes in the new models?*

Our Contributions. In this paper, for a multi-user system with multiple public keys where each public key will be used multiple times, we give a new security definition of SO security, denoted as SIM-Bi-SO-CCA security. In the security model, the adversary may adaptively corrupt some fraction of senders and receivers *simultaneously*, and get the plaintext messages together with internal randomness for encryption and secret keys for decryption, while it is hoped that messages of uncorrupted parties remain protected. (The definition is reminiscent of Bi-Deniability [30] for PKE.) We prove that some practical PKE schemes achieve SIM-Bi-SO-CCA security in the random oracle model.

Then, we suggest a weak model of SIM-Bi-SO-CCA security, denoted as SIM-wBi-SO$_k$-CCA security ($k \in \mathbb{N}$), where (i) the adversary has to specify whether it is going to corrupt the senders or the receivers after receiving the public keys and before seeing the challenge ciphertexts, and

[4] Very recently, Yang et al. [33] formalized the notion of RSO security in the multi-challenge setting. But their work only considers the receiver corruption setting.

(ii) if the adversary chooses to corrupt some fraction of the receivers, it is just allowed to corrupt the receivers whose public keys are employed for encryption *at most k times*. We stress that the weak model is still meaningful and useful because it provides the original SIM-SSO-CCA security and SIM-RSO-CCA security *simultaneously*. Furthermore, we show that SIM-wBi-SO$_k$-CCA security is strictly stronger than SIM-SSO-CCA security and SIM-RSO-CCA security. We also stress that the recently proposed SIM-RSO$_k$-CCA security notion [33] is a special case of our SIM-wBi-SO$_k$-CCA security.

Finally, we propose a generic construction of PKE that achieves SIM-wBi-SO$_k$-CCA security in the standard model and instantiate it from various standard assumptions. Our generic construction is built on a new variant of hash proof system (HPS), which should additionally satisfy the universal$_{k+1}$ property and key equivocability. The technical overview of the generic construction is given in Sec. 4.1. We also explore the existence of universal$_{k+1}$ HPS with key equivocability and provide instantiations from either the DDH assumption or the DCR assumption.

Related works. Since proposed by Bellare et al. in [3], selective opening secure PKE has been extensively studied.

For SSO security, Bellare et al. in [3] firstly showed that any lossy encryption is IND-SSO-CPA secure. IND-SSO-CCA secure PKE schemes were constructed from All-But-N lossy trapdoor functions [14] or All-But-Many lossy trapdoor functions [17,26,5,22]. If this lossy encryption has an efficient opener, then the resulting PKE scheme can be proven to be SIM-SSO-CCA secure as shown in [3]. Fehr et al. [10] showed an approach, employing extended hash proof system and cross-authentication code (XAC), to build SIM-SSO-CCA secure PKE schemes. As pointed out in [21], a stronger property of XAC is needed to make the proof rigorous. Following this line of research, a generic construction of SIM-SSO-CCA seucre PKE, from a special kind of key encapsulation mechanism (KEM) and a strengthened XAC, was proposed in [27] and then extended to achieve tight security in [28]. As showed in [15,16], some practical PKE constructions also enjoy SIM-SSO-CCA security.

For RSO security, Hazay et al. [13] showed that SIM-RSO-CPA secure PKE can be built from non-committing encryption for receiver (NCER) [6], and IND-RSO-CPA secure PKE can be built from a tweaked variant of NCER. IND-RSO-CCA secure PKE schemes were proposed in [24]. SIM-RSO-CCA secure PKE was constructed using indistinguishability obfuscation (iO) in [23], and constructed based on standard computation-

al assumptions in [12,20]. Recently, Yang et al. [33] formalized the notion of multi-challenge RSO security (RSO_k security), proved that SIM-RSO security is not enough to guarantee SIM-RSO_k security ($k > 1$), and showed SIM-RSO_k-CPA/CCA secure PKE constructions.

Roadmap. In the rest part of this work, we give some preliminaries in Sec. 2. We introduce the formal definitions for SIM-Bi-SO-CCA security and SIM-wBi-SO_k-CCA security ($k \in \mathbb{N}$), and show that SIM-wBi-SO_k-CCA security is strictly stronger than SIM-SSO-CCA and SIM-RSO-CCA security in Sec. 3. Next, we introduce the main building block, namely, universal$_\kappa$ HPS with key equivocability, and present a generic construction of PKE scheme that achieves SIM-wBi-SO_k-CCA security in the standard model in Sec. 4. Finally, we show that some practical PKE schemes achieve SIM-Bi-SO-CCA security in the random oracle model, in Sec. 5.

2 Preliminaries

Notations. Throughout this paper, let $\lambda \in \mathbb{N}$ denote the security parameter. For $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. For a finite set $\mathcal{S}$, we use $|\mathcal{S}|$ to denote the size of $\mathcal{S}$; we use $s \leftarrow \mathcal{S}$ to denote the process of sampling s uniformly from $\mathcal{S}$. For a distribution Dist, $x \leftarrow \mathsf{Dist}$ denotes the process of sampling x from Dist.

We use boldface to denote vectors, e.g., $\mathbf{x}$. We use $\mathbf{x}[i]$ to denote the i-th component of $\mathbf{x}$.

For a probabilistic algorithm $\mathcal{A}$, let $\mathcal{R}_\mathcal{A}$ denote the randomness space of $\mathcal{A}$. We let $y \leftarrow \mathcal{A}(x; r)$ denote the process of running $\mathcal{A}$ on input x and inner randomness $r \in \mathcal{R}_\mathcal{A}$ and outputting y. We write $y \leftarrow \mathcal{A}(x)$ for $y \leftarrow \mathcal{A}(x; r)$ with uniformly chosen $r \in \mathcal{R}_\mathcal{A}$. We write PPT for probabilistic polynomial-time. For a function $f(\lambda)$, we write that $f(\lambda) \leq \mathsf{negl}(\lambda)$ if it is negligible.

For two distributions Dist_1 and Dist_2, the statistical distance between Dist_1 and Dist_2 is defined as

$$\Delta(\mathsf{Dist}_1, \mathsf{Dist}_2) := \frac{1}{2} \sum_x \left| \Pr_{X_1 \leftarrow \mathsf{Dist}_1}[X_1 = x] - \Pr_{X_2 \leftarrow \mathsf{Dist}_2}[X_2 = x] \right|.$$

We say that Dist_1 and Dist_2 are statistically indistinguishable (denoted by $\mathsf{Dist}_1 \overset{s}{\approx} \mathsf{Dist}_2$), if $\Delta(\mathsf{Dist}_1, \mathsf{Dist}_2)$ is negligible.

Collision-resistant hash. We recall the definition of collision-resistant hash function here.

Definition 1. (Collision-resistant hash function). A family of *collision-resistant hash function* $\mathcal{H}$, with domain Dom and range Rge, is a family of functions having the following property: for any PPT algorithm $\mathcal{A}$, its advantage $\mathtt{Adv}^{\mathrm{CR}}_{\mathcal{H},\mathcal{A}}(\lambda) := \Pr[\mathsf{H} \leftarrow \mathcal{H}; (x, x') \leftarrow \mathcal{A}(\mathsf{H}) : x \neq x' \bigwedge \mathsf{H}(x) = \mathsf{H}(x')]$ is negligible.

Efficiently samplable and explainable domain. In this paper, some of the domains are required to be efficiently samplable and explainable [10]. We recall its definition as follows.

Definition 2. (Efficiently samplable and explainable domain). *We say that a domain* Dom *is* efficiently samplable and explainable*, if there are two PPT algorithms* $(\mathsf{Sample}, \mathsf{Explain})$*:*

- $\mathsf{Sample}(\mathsf{Dom}; r)$: *On input a domain* Dom *with uniformly sampled* $r \leftarrow \mathcal{R}_{\mathsf{Sample}}$*,* Sample *outputs an element which is uniformly distributed over* Dom*.*
- $\mathsf{Explain}(\mathsf{Dom}, x)$: *On input* Dom *and* $x \in \mathsf{Dom}$*,* $\mathsf{Explain}$ *outputs* r *which is uniformly distributed over the set* $\{r \in \mathcal{R}_{\mathsf{Sample}} \mid \mathsf{Sample}(\mathsf{Dom}; r) = x\}$*.*

This notion can be relaxed by allowing a negligibly small error probability (which includes that sampling algorithms may produce near-uniform output).

Cross-authentication code. The notion of ℓ-cross-authentication code (XAC) was proposed by Fehr et al. [10], and later adapted to strong and semi-unique XAC in [25].

Definition 3. (ℓ-Cross-authentication code). For $\ell \in \mathbb{N}$, an ℓ-cross-authentication code (ℓ-XAC) XAC, associated with a key space $\mathcal{XK}$ and a tag space $\mathcal{XT}$, consists of three PPT algorithms (XGen, XAuth, XVer). Algorithm $\mathsf{XGen}(1^\lambda)$ generates a uniformly random key $K \in \mathcal{XK}$, deterministic algorithm $\mathsf{XAuth}(K_1, \cdots, K_\ell)$ produces a tag $T \in \mathcal{XT}$, and deterministic algorithm $\mathsf{XVer}(K, T)$ outputs $b \in \{0, 1\}$. The following properties are required:

- **Correctness:** For all $i \in [\ell]$, $\mathsf{fail}_{\mathsf{XAC}}(\lambda) := \Pr[\mathsf{XVer}(K_i, \mathsf{XAuth}(K_1, \cdots, K_\ell)) \neq 1]$ is negligible, where $K_1, \cdots, K_\ell \leftarrow \mathsf{XGen}(1^\lambda)$ in the probability.

- **Security against impersonation and substitution attacks:** $\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda)$ and $\mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)$ as defined below are both negligible: $\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda) := \max_{i,T'} \Pr[K \leftarrow \mathsf{XGen}(1^\lambda) : \mathsf{XVer}(K, T') = 1]$, where the max is over all $i \in [\ell]$ and $T' \in \mathcal{XT}$, and

$$\mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda) := \max_{i,K_{\neq i},F} \Pr\left[\begin{array}{l} K_i \leftarrow \mathsf{XGen}(1^k) \\ T = \mathsf{XAuth}((K_j)_{j\in[\ell]}) \\ T' \leftarrow F(T) \end{array} : \begin{array}{c} T' \neq T \bigwedge \\ \mathsf{XVer}(K_i, T') = 1 \end{array}\right],$$

  where the max is over all $i \in [\ell]$, all $K_{\neq i} := (K_j)_{j\neq i} \in \mathcal{XK}^{\ell-1}$ and all possibly randomized functions $F : \mathcal{XT} \to \mathcal{XT}$.

Definition 4. (Strong and semi-unique ℓ-XAC). For $\ell \in \mathbb{N}$, we say that an ℓ-XAC XAC is strong and semi-unique, if it has the following two properties:

- **Strongness:** There is a PPT algorithm ReSamp, which takes $i \in [\ell]$, $K_{\neq i}$ and T as input (where $K_1, \cdots, K_\ell \leftarrow \mathsf{XGen}(1^\lambda)$ and $T = \mathsf{XAuth}((K_j)_{j\in[\ell]})$) and outputs K'_i, such that K'_i and K_i are statistically indistinguishable, i.e.,

$$\begin{aligned} \mathtt{StD}^{\mathrm{STRN}}_{\mathsf{XAC}}(\lambda) &:= \Delta(K'_i, K_i) \\ &= \frac{1}{2} \sum_{K\in\mathcal{XK}} \left|\Pr[K'_i = K|(K_{\neq i}, T)] - \Pr[K_i = K|(K_{\neq i}, T)]\right| \end{aligned}$$

  is negligible, where the probabilities are taken over $K_i \leftarrow \mathsf{XGen}(1^\lambda)$, conditioned on $(K_{\neq i}, T)$, and the randomness of ReSamp.
- **Semi-uniqueness:** The key space $\mathcal{XK}$ can be written as $\mathcal{K}_a \times \mathcal{K}_b$. Given a tag $T \in \mathcal{XT}$ and $K_a \in \mathcal{K}_a$, there is at most one $K_b \in \mathcal{K}_b$ such that $\mathsf{XVer}((K_a, K_b), T) = 1$.

3 Bi-SO Security for PKE

Previous security notions of SOA for PKE only consider *either* sender corruption setting *or* receiver corruption setting. We consider a more natural and general setting for selective opening security. In the setting, the adversary may adaptively corrupt part of senders and receivers *simultaneously*. We denote it as Bi-SO security since it is reminiscent of Bi-Deniability [30] for PKE.

For a multi-user system with multiple public keys where each public key will be used many times, we firstly give the most natural security

notion of Bi-SO security, denoted as SIM-Bi-SO-CCA security. Then, we suggest a weak model of SIM-Bi-SO-CCA security, denoted as SIM-wBi-SO$_k$-CCA security ($k \in \mathbb{N}$). The weak model is still meaningful and useful because it provides the original SIM-SSO-CCA security and SIM-RSO-CCA security *simultaneously*. Finally, for completeness, we show that SIM-wBi-SO$_k$-CCA security is strictly stronger than SIM-SSO-CCA and SIM-RSO-CCA security.

3.1 Security Definitions

Simulation-based Bi-SO security. In the Bi-SO setting, some of the senders and some of the receivers may be corrupted *simultaneously*, and each public key may be used to encrypt multiple messages. The formal definition is as follows.

Definition 5. (SIM-Bi-SO-CCA). *We say that a PKE scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$[5] *is* SIM-Bi-SO-CCA *secure, if for any PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S}$*, such that for any PPT distinguisher* $\mathcal{D}$*,*

$$\begin{aligned}\mathtt{Adv}^{\text{SIM-Bi-SO-CCA}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda) := |\Pr[\mathcal{D}(\mathtt{Exp}^{\text{Bi-SO-real}}_{\mathsf{PKE},\mathcal{A}}(\lambda)) = 1] \\ - \Pr[\mathcal{D}(\mathtt{Exp}^{\text{Bi-SO-ideal}}_{\mathsf{PKE},\mathcal{S}}(\lambda)) = 1]|\end{aligned}$$

is negligible, where both $\mathtt{Exp}^{\text{Bi-SO-real}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *and* $\mathtt{Exp}^{\text{Bi-SO-ideal}}_{\mathsf{PKE},\mathcal{S}}(\lambda)$ *are defined in Fig. 1.*

Note that in the real experiment, the total number of public keys and the times that each public key is used for encryption are completely determined by the adversary.

Remark 1 *One can generalize both SIM-Bi-SO-CCA and SIM-wBi-SO$_k$-CCA security to a new version that the adversary is allowed to make multiple selective opening queries* adaptively*. We stress that all the PKE constructions presented in this paper also achieve the generalized security.*

[5] Note that both SIM-Bi-SO-CCA and SIM-wBi-SO$_k$-CCA security capture the security requirements in a multi-user scenario, where multiple public/secret key pairs are involved. In this setting, some global information is needed to be generated by a global algorithm Setup, as done in previous works about multi-user security, such as [1].

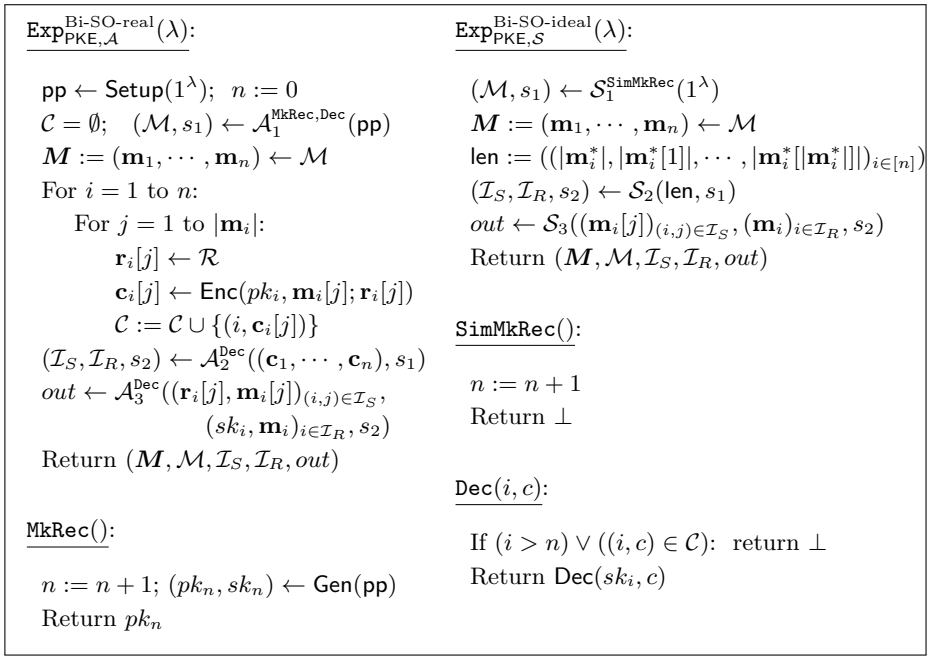

Fig. 1 Experiments for defining SIM-Bi-SO-CCA security of PKE. In these two experiments, we require that $\mathcal{I}_S \subset \{(i,j) \mid i \in [n], j \in [|\mathbf{m}_i|]\}$ and $\mathcal{I}_R \subset [n]$.

Simulation-based weak Bi-SO security. Now we introduce a weak model of SIM-Bi-SO-CCA security, which we denote as SIM-wBi-SO$_k$-CCA security ($k \in \mathbb{N}$). The differences between these two security models are that in the real experiment of SIM-wBi-SO$_k$-CCA security: (i) the adversary has to specify whether it is going to corrupt some fraction of the senders *or* the receivers, *before seeing the challenge ciphertexts*; (ii) if the adversary chooses to corrupt some fraction of the receivers, it is just allowed to corrupt the receivers whose public keys are used for encryption *at most k times*. The formal definition is as follows.

Definition 6. (SIM-wBi-SO$_k$-CCA). *For any $k \in \mathbb{N}$, we say that a PKE scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* SIM-wBi-SO$_k$-CCA *secure, if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$, such that for any PPT distinguisher $\mathcal{D}$,*

$$\mathtt{Adv}^{\text{SIM-wBi-SO}_k\text{-CCA}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda) := |\Pr[\mathcal{D}(\mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda)) = 1]|$$

is negligible, where both $\mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)$ *and* $\mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda)$ *are defined in Fig. 2.*

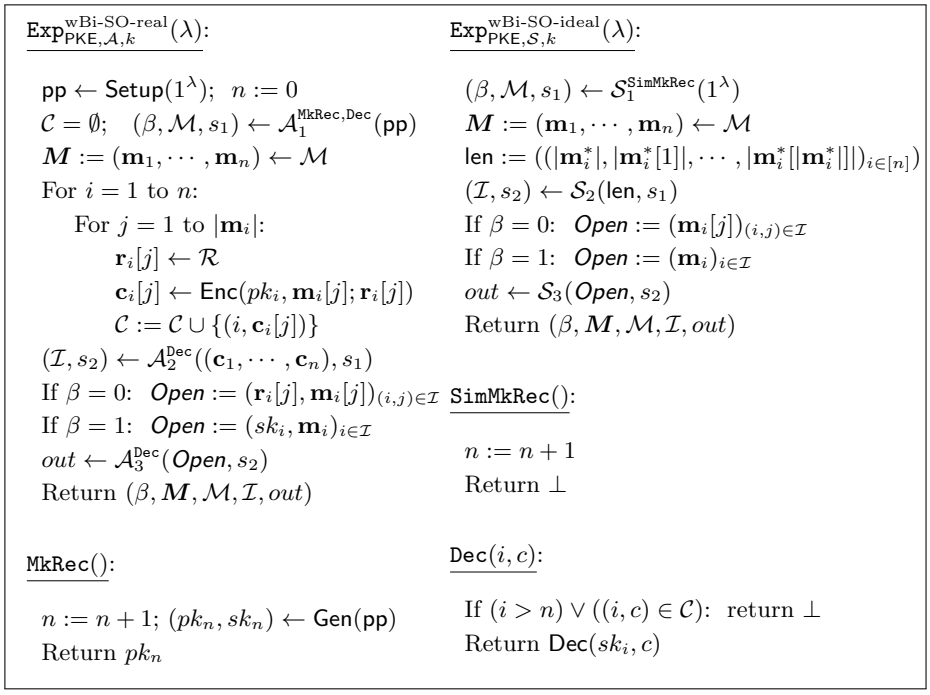

Fig. 2 Experiments for defining SIM-wBi-SO$_k$-CCA security. Here in both $\mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)$ and $\mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda)$, we require that (i) $\beta \in \{0, 1\}$, and (ii) when $\beta = 0$, $\mathcal{I} \subset \{(i, j) \mid i \in [n], j \in [|\mathbf{m}_i|]\}$, and when $\beta = 1$, $\mathcal{I} \subset \{i \in [n] \mid |\mathbf{m}_i| \leq k\}$.

In both $\mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)$ and $\mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda)$, we use $\beta = 0$ (resp. $\beta = 1$) to represent that adversary $\mathcal{A}$/simulator $\mathcal{S}$ chooses to corrupt some of the senders (resp. receivers). We stress that in $\mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)$, when $\mathcal{A}_1$ outputs $\beta = 0$, the parameter k puts no restrictions on sender corruptions $\mathcal{I}$; and when $\mathcal{A}_1$ outputs $\beta = 1$, $\mathcal{A}_2$ is allowed to corrupt the receivers whose public keys are used for encryption at most k times (i.e., $\mathcal{I} \subset \{i \in [n] \mid |\mathbf{m}_i| \leq k\}$).

Note that the original SIM-SSO-CCA security [14,10] and SIM-RSO-CCA security [12,20] are both special cases of SIM-wBi-SO$_k$-CCA security. Specifically, the original SIM-SSO-CCA security is SIM-wBi-SO$_k$-CCA security when $\mathcal{A}_1$ always outputs $\beta = 0$ and queries the $\mathtt{MkRec}$ oracle on-

ly once[6], and the original SIM-RSO-CCA security is SIM-wBi-SO$_k$-CCA security when $\mathcal{A}_1$ always outputs $\beta = 1$ and $|\mathbf{m}_1| = \cdots = |\mathbf{m}_n| = 1$ (note that the latter implicitly suggests $k = 1$). Hence, for a SIM-wBi-SO$_k$-CCA secure PKE scheme, it achieves the original SIM-SSO-CCA and SIM-RSO-CCA (and even SIM-RSO$_k$-CCA) security *simultaneously*.

Very recently, Yang et al. [33] introduced an enhanced security notion of RSO, *SIM-RSO$_k$-CCA security* ($k \in \mathbb{N}$), for PKE. We notice that their SIM-RSO$_k$-CCA security is a special case of SIM-wBi-SO$_k$-CCA security as well. Specifically, SIM-RSO$_k$-CCA security is SIM-wBi-SO$_k$-CCA security when $\mathcal{A}_1$ always outputs $\beta = 1$.

3.2 Separation of SIM-wBi-SO$_k$-CCA and SIM-SSO-CCA & SIM-RSO-CCA

Now we show that SIM-wBi-SO$_k$-CCA security is *strictly* stronger than SIM-SSO-CCA security and SIM-RSO-CCA security. Our conclusion is derived from the fact that SIM-wBi-SO$_k$-CCA security implies SIM-SSO-CCA and SIM-RSO-CCA security simultaneously, and SIM-SSO-CCA and SIM-RSO-CCA security do not imply each other. Actually, we have stronger conclusions:

(1) Supposing that the κ-Linear assumption holds ($\kappa \in \mathbb{N}$), SIM-SSO-CCA security does not imply SIM-RSO-*CPA* security;
(2) Supposing that the DDH or DCR assumption holds, SIM-RSO-CCA security does not imply SIM-SSO-*CPA* security.

SIM-SSO-CCA*$\not\Rightarrow$*SIM-RSO-CPA. Bellare et al. [2] introduced the notion of *decryption verifiability* for PKE, and showed that assuming the existence of a family of collision-resistant hash functions, which can be constructed under the discrete-logarithm assumption [11], any decryption-verifiable PKE scheme is not SIM-RSO-CPA secure [2, Theorem 5.1][7].

Informally, a PKE scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is called *decryption-verifiable*, if it is infeasible to generate $(pk, sk_0, sk_1, c, m_0, m_1)$ such that (i) $m_0 \neq m_1$, (ii) both sk_0 and sk_1 are valid secret keys corresponding to pk, and (iii) $\mathsf{Dec}(sk_0, c) = m_0$ and $\mathsf{Dec}(sk_1, c) = m_1$. We

[6] The SIM-SSO-CPA security notion presented in [4] allows the adversary to query the `MkRec` oracle multiple times.

[7] Both [2, Theorem 5.1] and [2, Theorem 4.1] only hold in the the auxiliary input model (i.e., in the experiments defining SIM-RSO-CPA and SIM-SSO-CPA security, both the adversary and the simulator get an auxiliary input). So do our counterexamples in this section. These counterexamples may be modified with the technique proposed in [2, Sec. 6] to drop the auxiliary inputs.

note that (i) and (iii) implicitly suggest that $sk_0 \neq sk_1$. In other words, for any PKE scheme, if each of its public key uniquely determines its corresponding secret key, then it must be decryption-verifiable.

We notice that the κ-Linear-based SIM-SSO-CCA secure PKE scheme proposed by Liu and Paterson [27] is such a decryption-verifiable PKE scheme. Generally, a public key of the κ-Linear-based Liu-Paterson scheme is of the form $(g^y, (g^{x_\theta}, g^{x_\theta \alpha_\theta}, g^{x_\theta \beta_\theta})_{\theta \in [\kappa]})$, where g is a generator of a cyclic group $\mathbb{G}$ of prime order q and $(y, (x_\theta, \alpha_\theta, \beta_\theta)_{\theta \in [\kappa]}) \in (\mathbb{Z}_q)^{3\kappa+1}$, and the corresponding secret key is $(\alpha_\theta, \beta_\theta, x_\theta^{-1} y)_{\theta \in [\kappa]}$. It's obvious that the public key uniquely determines its corresponding secret key. So the κ-Linear-based Liu-Paterson scheme is decryption-verifiable. According to [2, Theorem 5.1], we conclude that assuming the existence of a family of collision-resistant hash functions, the κ-Linear-based Liu-Paterson scheme is not SIM-RSO-CPA secure.

For completeness, we recall the formal definition of decryption verifiability [2] and the κ-Linear-based Liu-Paterson scheme [27] in Appendix B and C respectively.

SIM-RSO-CCA$\not\Rightarrow$SIM-SSO-CPA. As pointed out in [2, Theorem 4.1], the DDH-based Cramer-Shoup scheme [7] is not SIM-SSO-CPA secure. On the other hand, Huang et al. [20] and Hara et al. [12] showed that this PKE scheme (for single-bit message) achieves SIM-RSO-CCA security. This fact suggests that when the DDH assumption holds, SIM-RSO-CCA security does not imply SIM-SSO-CPA security. With similar analysis, this conclusion can be extended to the case that the DCR assumption holds.

4 PKE with SIM-wBi-SO$_k$-CCA Security

In this section, we propose a PKE scheme achieving SIM-wBi-SO$_k$-CCA security. We firstly introduce a new primitive, universal$_\kappa$ HPS with key equivocability for any polynomially bounded function κ, and provide concrete constructions for it from the DDH assumption and the DCR assumption respectively. Then, with this new primitive as a building block, we show our PKE construction and prove that it meets SIM-wBi-SO$_k$-CCA security in the standard model.

In order to make our idea more understandable, we firstly provide a technique overview before going into the details.

4.1 Technique Overview

In the real experiment of SIM-wBi-SO$_k$-CCA security, the bit β is used to indicate whether the adversary wants to corrupt some fraction of the senders ($\beta = 0$) or the receivers ($\beta = 1$), and the adversary does not specify the value of β until it sees public keys $(pk_i)_{i\in[n]}$ via querying the oracle $\texttt{MkRec}$. Hence, to prove SIM-wBi-SO$_k$-CCA security, when $\beta = 0$, we need to somehow generate malformed ciphertexts for $(pk_i)_{i\in[n]}$, such that they can be opened in the sense of SSO (i.e., exposing the messages and the corresponding randomness to the adversary); and when $\beta = 1$, we need to somehow generate malformed ciphertexts for $(pk_i)_{i\in[n]}$, such that they can be opened in the sense of RSO (i.e., exposing the messages and the corresponding secret keys to the adversary).

Our scheme, encrypting ℓ-bit messages, is inspired by the works of [10,21,25]. The public/secret key pair is ℓ pairs of public and secret keys (i.e., $(hpk_\gamma, hsk_\gamma)_{\gamma\in[\ell]}$) of a hash proof system (HPS) HPS [8]. Informally, to encrypt a message $m = (m_1, \cdots, m_\ell) \in \{0,1\}^\ell$, the encryption algorithm sets that for each $\gamma \in [\ell]$,

$$\begin{cases} \text{If } m_\gamma = 0: \ x_\gamma \leftarrow \mathcal{X};\ K_\gamma \leftarrow \mathcal{K}_{sp} \\ \text{If } m_\gamma = 1: \ x_\gamma \leftarrow \mathcal{L};\ K_\gamma = \mathsf{PubEv}(hpk_\gamma, x_\gamma, w_\gamma) \end{cases}$$

where $\mathcal{L} \subset \mathcal{X}$ and $\mathcal{X}$ are both finite sets generated with a hard subset membership problem, PubEv is the public evaluation algorithm of HPS, w_γ is a witness for $x_\gamma \in \mathcal{L}$, and $\mathcal{K}_{sp}$ is the range of PubEv. Then, we use a strengthened cross-authentication code (XAC) to "glue" $x_1, \cdots, x_\ell$ together, obtaining a XAC tag T. So the generated ciphertext corresponding to m is $c = (x_1, \cdots, x_\ell, T)$. To decrypt a ciphertext $c = (x_1, \cdots, x_\ell, T)$, the decryption algorithm firstly computes that $(\overline{K}_\gamma = \mathsf{SecEv}(hsk_\gamma, x_\gamma))_{\gamma\in[\ell]}$, where SecEv is the secret evaluation algorithm of HPS, and then for each $\gamma \in [\ell]$, sets $\overline{m}_\gamma = 1$ if and only if T is verified correctly by $\overline{K}_\gamma$ (via the verification algorithm of XAC).

Now we turn to the security proof. In order to prove SIM-wBi-SO$_k$-CCA security, we need to construct a PPT simulator $\mathcal{S}$, such that the ideal experiment and the real experiment are indistinguishable. In particular, we need to generate some malformed ciphertexts (before seeing the real messages), such that they are computationally indistinguishable from the real challenge ciphertexts, and meanwhile can be efficiently opened according to the value of β.

If $\beta = 0$, we need to generate malformed ciphertexts $c = (x_1, \cdots, x_\ell, T)$, and then open them according to the real messages $m = (m_1, \cdots, m_\ell)$,

by providing random coins which can be used to encrypt the real messages to recover the malformed ciphertexts. We generate the malformed ciphertexts with encryptions of ℓ ones, i.e., for each $\gamma \in [\ell]$, $x_\gamma \leftarrow \mathcal{L} \subset \mathcal{X}$ and $K_\gamma = \mathsf{PubEv}(hpk_\gamma, x_\gamma, w_\gamma) \subset \mathcal{K}_{sp}$. Hence, after generating these malformed ciphertexts, to open a ciphertext, for each $\gamma \in [\ell]$, if the real message bit $m_\gamma = 1$, the random coin (i.e., w_γ) employed to generate (x_γ, K_γ) can be returned directly; if $m_\gamma = 0$, return the random coin which is generated by explaining x_γ as a random element sampled from $\mathcal{X}$, and explaining K_γ as a random key sampled from $\mathcal{K}_{sp}$.

Now, we show that a real challenge ciphertext can be substituted with the malformed ciphertext without changing the adversary's view significantly. For $\gamma = 1$ to ℓ,

1) We modify the decryption procedure of the decryption oracle, such that it does not make use of hsk_γ. More specifically, for a decryption query $c' = (x'_1, \cdots, x'_\ell, T')$, if $x'_\gamma \notin \mathcal{L}$, the decryption oracle directly sets $\overline{m}_\gamma = 0$. The statistical properties of HPS and strengthened XAC guarantee that this modification does not change the adversary's view significantly.
2) If $m_\gamma = 0$, the randomly sampled K_γ is replaced with $K_\gamma = \mathsf{SecEv}(hsk_\gamma, x_\gamma)$. The perfect universality of HPS guarantees that this change is imperceptible to the adversary.
3) If $m_\gamma = 0$, K_γ is updated again via the resampling algorithm of strengthened XAC. The statistical property of strengthened XAC guarantees that this modification does not change the adversary's view significantly.
4) The decryption procedure of the decryption oracle is changed to work with the original decryption rules. The statistical properties of HPS and strengthened XAC guarantee that this modification is imperceptible to the adversary.
5) If $m_\gamma = 0$, $x_\gamma \leftarrow \mathcal{L}$ instead of uniformly sampling from $\mathcal{X}$. The underlying subset membership problem of HPS guarantees that this change is also imperceptible to the adversary.

Note that these substitutions only consider the situation that a single public key is used to encrypt a single message. Fortunately, we can extend it to the situation that there are n public keys (for any $n \in \mathbb{N}$), and each public key is employed to encrypt multiple messages.

If $\beta = 1$, we need to generate malformed ciphertexts, and then open them according to the real messages, by providing valid secret keys which can be used to decrypt the malformed ciphertexts to obtain the messages.

Note that a public key of this scheme is of the form $pk = (hpk_1, \cdots, hpk_\ell)$, and the corresponding secret key is $sk = (hsk_1, \cdots, hsk_\ell)$. Hence, informally, what we need is to generate a malformed ciphertext without seeing the message, such that for any message $m = (m_1, \cdots, m_\ell) \in \{0, 1\}^\ell$, we can generate some secret key $sk' = (hsk'_1, \cdots, hsk'_\ell)$ satisfying that (i) sk' is a valid secret key corresponding to pk (i.e., for all $\gamma \in [\ell]$, hsk'_γ is a valid HPS secret key corresponding to hpk_γ); (ii) decrypting the malformed ciphertext with sk' will lead to m.

We try to generate such a malformed ciphertext $c = (x_1, \cdots, x_\ell, T)$. For each $\gamma \in [\ell]$, if $x_\gamma \in \mathcal{L}$ (with a witness w_γ), all the HPS secret keys corresponding to hpk_γ will lead to the same $\widetilde{K}_\gamma = \mathsf{PubEv}(hpk_\gamma, x_\gamma, w_\gamma) = K_\gamma$. In other words, for any fixed ciphertext $(\cdots, x_\gamma, \cdots, T)$, no matter what the secret key is, the decryption of this ciphertext will lead to the same $\overline{m}_\gamma$. So it's impossible to open the malformed ciphertext successfully when $m_\gamma = 1 - \overline{m}_\gamma$. Hence, our malformed ciphertexts focus on the case $c = (x_1, \cdots, x_\ell, T)$ that $x_1, \cdots, x_\ell \in \mathcal{X} \backslash \mathcal{L}$. On the other hand, if K_γ is uniformly sampled, it seems unlikely to decrypt the ciphertext to recover the original message when $m_\gamma = 1$ due to the property of XAC. So our malformed ciphertexts further focus on the case $c = (x_1, \cdots, x_\ell, T)$ that for all $\gamma \in [\ell]$, $x_\gamma \in \mathcal{X} \setminus \mathcal{L}$ and $K_\gamma = \mathsf{SecEv}(hsk_\gamma, x_\gamma)$.

We stress that in the real experiment of SIM-wBi-SO$_k$-CCA security, the adversary is just allowed to corrupt the receivers whose public keys are used for encryption at most k times. So for simplicity, here we only consider the case that $pk = (hpk_1, \cdots, hpk_\ell)$ is used to encrypt *exactly* k messages (i.e., $m_j = (m_{j,1}, \cdots, m_{j,\ell}) \in \{0,1\}^\ell$ $(j \in [k])$). More specifically, for each $\gamma \in [\ell]$, hsk_γ is used k times (note that we use sk to generate the malformed ciphertexts), generating k ciphertext parts (i.e., $K_{1,\gamma} = \mathsf{SecEv}(hsk_\gamma, x_{1,\gamma}), \cdots, K_{k,\gamma} = \mathsf{SecEv}(hsk_\gamma, x_{k,\gamma})$). In other words, to generate the k malformed ciphertexts, for each $\gamma \in [\ell]$, we need to

(i) compute $\mathsf{SecEv}(hsk_\gamma, x_{1,\gamma}), \cdots, \mathsf{SecEv}(hsk_\gamma, x_{k,\gamma})$ for some $x_{1,\gamma}, \cdots, x_{k,\gamma} \in \mathcal{X} \setminus \mathcal{L}$ before seeing the messages;

(ii) generate a HPS secret key hsk'_γ such that $\mathsf{SecEv}(hsk'_\gamma, x_{j,\gamma}) = \mathsf{SecEv}(hsk_\gamma, x_{j,\gamma})$ if $m_{j,\gamma} = 1$, and $\mathsf{SecEv}(hsk'_\gamma, x_{j,\gamma}) \neq \mathsf{SecEv}(hsk_\gamma, x_{j,\gamma})$ if $m_{j,\gamma} = 0$.

However, there is no algorithm for HPS which can generate two HPS secret keys (i.e. hsk_γ and hsk'_γ) meeting the above requirements. Therefore, we introduce the following new property, which we call "key equivocability", of HPS. Informally, we require that there is an efficient algorithm $\mathsf{SampHsk}$ and a trapdoor td, such that for any $x_1, \cdots, x_k \in \mathcal{X} \setminus \mathcal{L}$, the

following two distribution ensembles, Dist_0^k and Dist_1^k, are statistically indistinguishable:

$$\begin{aligned}\mathsf{Dist}_0^k := \{(hsk, K_1, \cdots, K_k, hpk) \big| & hsk \leftarrow \mathcal{SK};\ hpk = \mu(hsk); \\ & \forall j \in [k]: \\ & \quad K_j \leftarrow \mathcal{K}_{sp} \quad \text{if } m_j = 0; \\ & \quad K_j = \mathsf{SecEv}(hsk, x_j) \quad \text{if } m_j = 1\}, \quad (1)\end{aligned}$$

$$\begin{aligned}\mathsf{Dist}_1^k := \{(hsk', K_1, \cdots, K_k, hpk) \big| & hsk \leftarrow \mathcal{SK};\ hpk = \mu(hsk); \\ & (K_j = \mathsf{SecEv}(hsk, x_j))_{j\in[k]}; \\ & hsk' \leftarrow \mathsf{SampHsk}(hsk, \mathsf{td}, \{x_j\}_{j\in[k]})\}. \quad (2)\end{aligned}$$

We stress that this property requires that no information about hsk beyond hpk is leaked. Similar to the proof of case $\beta = 0$, we introduce a modification to the decryption oracle before employing the key equivocability of HPS in order to make sure that nothing about hsk beyond hpk is leaked. For any decryption query $(x'_1, \cdots, x'_\ell, T')$ and any γ, if $x'_\gamma \in \mathcal{X} \setminus \mathcal{L}$, the decryption oracle sets $\overline{m}_\gamma = 0$ directly. However, we note that in the SIM-wBi-SO$_k$-CCA security model, each public key is used to encrypt k messages. As a result, hsk may be employed k times, i.e., to compute $\mathsf{SecEv}(hsk, x_1), \cdots, \mathsf{SecEv}(hsk, x_k)$ for some $x_1, \cdots, x_k$. So the perfect universality$_2$ of HPS [8] is not enough to guarantee that the modification to the decryption oracle is imperceptible to the adversary. To solve this problem, we introduce another property, *perfect universality*$_{k+1}$, for HPS. Roughly speaking, HPS is called perfectly universal$_{k+1}$, if for any $x_1, \cdots, x_{k+1} \in \mathcal{X} \setminus \mathcal{L}$ and any $K' \in \mathcal{K}_{sp}$, even given $(hpk, \mathsf{SecEv}(hsk, x_1), \cdots, \mathsf{SecEv}(hsk, x_k))$, the probability that $\mathsf{SecEv}(hsk, x_{k+1}) = K'$ is $\frac{1}{|\mathcal{K}_{sp}|}$.

With the help of this new variant of HPS, we can use algorithm $\mathsf{SampHsk}$ to open the aforementioned equivocable ciphertexts $c = (x_1, \cdots, x_\ell, T)$ where for each $\gamma \in [\ell]$, $x_\gamma \in \mathcal{X} \setminus \mathcal{L}$ and $K_\gamma = \mathsf{SecEv}(hsk_\gamma, x_\gamma)$, successfully. Now, we show that a real challenge ciphertext can be substituted with the malformed ciphertext without changing the adversary's view significantly. A high-level description of the substitution is presented as follows.

1) We use the secret keys to generate the challenge ciphertexts, instead of the public keys. The statistical property of HPS guarantees that this change is imperceptible to the adversary.
2) All the $x_{j,\gamma}$ $(j \in [k], \gamma \in [\ell])$ are sampled from $\mathcal{X} \setminus \mathcal{L}$, instead of being sampled from $\mathcal{L}$ (when $m_{j,\gamma} = 1$). The underlying subset membership

problem of HPS guarantees that this change is also imperceptible to the adversary.

3) Note that $sk = (hsk_1, \cdots, hsk_\ell)$ is employed to encrypt $m_j = (m_{j,1}, \cdots, m_{j,\ell}) \in \{0,1\}^\ell$ $(j \in [k])$, and specifically, for each $\gamma \in [\ell]$, hsk_γ is used to handle $m_{1,\gamma}, \cdots, m_{k,\gamma}$, as shown in Fig. 3. For each $\gamma \in [\ell]$, employ hsk_γ to compute $K_{j,\gamma}$ when $m_{j,\gamma} = 0$ (for all $j \in [k]$). The key equivocability of HPS guarantees that this modification does not change the adversary's view significantly.

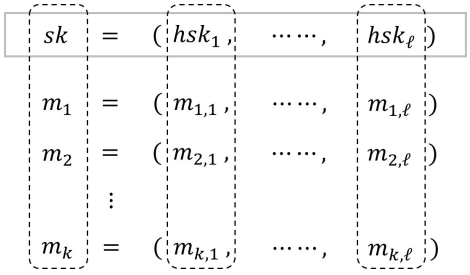

Fig. 3 Relations among sk and $m_1, \cdots, m_k$

4.2 Universal$_\kappa$ Hash Proof System with Key Equivocability

Now we introduce the main building block, namely, universal$_\kappa$ HPS with key equivocability, for any polynomially bounded κ, and show concrete constructions for it.

The definition. For any polynomially bounded function κ, we provide a definition of universal$_\kappa$ HPS with key equivocability, which enhances the standard HPS [8] with key equivocability and universal$_\kappa$ property. It works on a strengthened version of subset membership problem SSMP, which defines some additional languages and provides a trapdoor to recognize elements from these languages.

Definition 7 (Strengthened Subset Membership Problem). *A strengthened subset membership problem (SSMP)* SSMP *consists of five PPT algorithms* $(\mathsf{SSmpG}, \mathsf{SSmpX}, \mathsf{SSmpL}, \mathsf{SSmpLS}, \mathsf{SSmpChk})$*:*

– $\mathsf{SSmpG}(1^\lambda, k)$: *On input* 1^λ *and polynomially bounded* $k > 0$*, algorithm* SSmpG *outputs a system parameter* prm *and a trapdoor* td*. The parameter* prm *defines* $2k+2$ *sets* $(\mathcal{X}, \mathcal{L}, \mathcal{L}_1, \cdots, \mathcal{L}_{2k})$*, where* $\mathcal{X}$ *is an*

efficiently recognizable finite set, $\mathcal{L} \subset \mathcal{X}$, and $\mathcal{L}_1, \cdots, \mathcal{L}_{2k}$ are distinct subsets of $\mathcal{X} \setminus \mathcal{L}$. For simplicity of notation, we write

$$\mathsf{prm} = (\mathcal{X}, \mathcal{L}, \mathcal{L}_1, \ldots, \mathcal{L}_{2k})$$

when employing HPS *for* SSMP *to construct PKE schemes.*

- $\mathsf{SSmpX}(\mathsf{prm})$: *On input* prm, SSmpX *outputs a uniformly chosen $x \leftarrow \mathcal{X}$.*
- $\mathsf{SSmpL}(\mathsf{prm})$: *On input* prm, SSmpL *samples $x \leftarrow \mathcal{L}$ with randomness $w \in \mathcal{R}_{\mathsf{SSmpL}}$, and outputs (x, w). We say that w is a witness for $x \in \mathcal{L}$.*
- $\mathsf{SSmpLS}(\mathsf{prm}, i \in [2k])$: *On input* prm *and $i \in [2k]$,* SSmpLS *outputs a uniformly chosen $x_i \leftarrow \mathcal{L}_i$.*
- $\mathsf{SSmpChk}(\mathsf{prm}, \mathsf{td}, x)$: *On input* prm, td *and x,* SSmpChk *outputs an integer $[0, 2k]$ or an abort symbol $\perp$.*

Also, it satisfies the following properties:

- **Hardness.** *For all $i \in [2k]$, for any PPT distinguisher $\mathcal{D}$, the following advantages are all negligible,*

$$\mathtt{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D},i}(\lambda) := |\Pr[\mathcal{D}(\mathsf{prm}, x_{\mathcal{X}}) = 1] - \Pr[\mathcal{D}(\mathsf{prm}, x_i) = 1]|,$$

$$\mathtt{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D},i}(\lambda) := |\Pr[\mathcal{D}(\mathsf{prm}, x_{\mathcal{L}}) = 1] - \Pr[\mathcal{D}(\mathsf{prm}, x_i) = 1]|,$$

  where the probabilities are over $\mathsf{prm} \leftarrow \mathsf{SSmpG}(1^\lambda, k)$, $x_{\mathcal{X}} \leftarrow \mathsf{SSmpX}(\mathsf{prm})$, $(x_{\mathcal{L}}, w) \leftarrow \mathsf{SSmpL}(\mathsf{prm})$, and $x_i \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, i)$. [8]
- **Sparseness.** *The probability*

$$\mathtt{Spar}_{\mathsf{SSMP}}(\lambda) := \Pr[(\mathsf{prm}, \mathsf{td}) \leftarrow \mathsf{SSmpG}(1^\lambda, k); x_{\mathcal{X}} \leftarrow \mathsf{SampX}(\mathsf{prm}) : x_{\mathcal{X}} \in \mathcal{L}]$$

  is negligible.
- **Explainability.** *The finite set $\mathcal{X}$ is an efficiently samplable and explainable domain (as defined in Definition 2).*
- **Sampling Correctness.** *Let $(\mathsf{prm}, \mathsf{td}) \leftarrow \mathsf{SSmpG}(1^\lambda, k)$. Then the distributions of the outputs of* $\mathsf{SSmpX}(\mathsf{prm})$, $\mathsf{SSmpL}(\mathsf{prm})$, *and* $\mathsf{SSmpLS}(\mathsf{prm}, i)$ *($i \in [2k]$) are statistically indistinguishable from uniform distributions over $\mathcal{X}$, $\mathcal{L}$ and $\mathcal{L}_i$ ($i \in [2k]$) respectively.*
- **Checking Correctness.** *For any $(\mathsf{prm}, \mathsf{td})$ generated by* SSmpG, *if $x \in \mathcal{L}$, then $\mathsf{SSmpChk}(\mathsf{prm}, \mathsf{td}, x) = 0$; if there exists $i \in [2k]$ s.t. $x \in \mathcal{L}_i$, then $\mathsf{SSmpChk}(\mathsf{prm}, \mathsf{td}, x) = i$; otherwise, $\mathsf{SSmpChk}(\mathsf{prm}, \mathsf{td}, x) = \perp$.*

[8] Note that a hard SSMP is also a hard SMP, since a simple hybrid argument shows that for any PPT distinguisher $\mathcal{D}$, $|\Pr[\mathcal{D}(\mathsf{prm}, x_{\mathcal{X}}) = 1] - \Pr[\mathcal{D}(\mathsf{prm}, x_{\mathcal{L}}) = 1]| \leq \mathtt{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D},1}(\lambda) + \mathtt{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D},1}(\lambda)$.

Remark 2 *The additional trapdoor, generated by* SSmpG*, will also be used in the key equivocability property (see Definition 10) of HPS.*

Definition 8 (Hash Proof System [8]). *A* hash proof system HPS *for a SSMP* SSMP *consists of three PPT algorithms* (PrmG, PubEv, SecEv)*:*

- $\mathsf{PrmG}(\mathsf{prm})$: *Given* prm*, which is generated by* $\mathsf{SSmpG}(1^\lambda, k)$ *and defines* $2k+2$ *sets* $(\mathcal{X}, \mathcal{L}, \mathcal{L}_1, \ldots, \mathcal{L}_{2k})$*, algorithm* PrmG *outputs a parameterized instance* $\mathsf{prmins} := (\mathcal{K}_{sp}, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \mu)$*, where* $\mathcal{K}_{sp}$*,* $\mathcal{SK}$*,* $\mathcal{PK}$ *are all finite sets,* $\Lambda_{(\cdot)} : \mathcal{X} \to \mathcal{K}_{sp}$ *is a family of hash functions indexed with secret hash key* $hsk \in \mathcal{SK}$*, and* $\mu : \mathcal{SK} \to \mathcal{PK}$ *is an efficiently computable function.*
- $\mathsf{SecEv}(hsk, x)$: *On input* $hsk \in \mathcal{SK}$ *and* $x \in \mathcal{X}$*, the* deterministic *secret evaluation algorithm* SecEv *outputs a hash value* $K = \Lambda_{hsk}(x) \in \mathcal{K}_{sp}$*.*
- $\mathsf{PubEv}(hpk, x, w)$: *On input* $hpk = \mu(hsk) \in \mathcal{PK}$*,* $x \in \mathcal{L}$ *and a witness* w *for* $x \in \mathcal{L}$*, the* deterministic *public evaluation algorithm* PubEv *outputs a hash value* $K = \Lambda_{hsk}(x) \in \mathcal{K}_{sp}$*.*

Also, it should be

- **Projective.** *For any* $hsk \in \mathcal{SK}$ *and any* $x \in \mathcal{L}$ *with witness* w*, the hash value* $\Lambda_{hsk}(x)$ *is uniquely determined by* $hpk = \mu(hsk)$ *and* x*, concretely, we require that* $\mathsf{SecEv}(hsk, x) = \mathsf{PubEv}(hpk, x, w)$*.*
- **Perfectly Universal.** *For all* prm *generated by* $\mathsf{SSmpG}(1^\lambda)$*, all possible* $\mathsf{prmins} \leftarrow \mathsf{PrmG}(\mathsf{prm})$*, all* $hpk \in \mathcal{PK}$*, all* $x \in \mathcal{X} \setminus \mathcal{L}$*, and all* $K \in \mathcal{K}_{sp}$*, the probability* $\Pr[\Lambda_{hsk}(x) = K \mid \mu(hsk) = hpk] = \frac{1}{\mathcal{K}_{sp}}$*, where the probability is over* $hsk \leftarrow \mathcal{SK}$*.*

Definition 8 is the same as the original definition of HPS in [8]. In our PKE construction, we further require that $\mathcal{K}_{sp}$ is efficiently samplable and explainable. Besides, we require HPS to have the following two properties.

Definition 9 (Perfectly Universal$_\kappa$). *For any polynomial* κ*, we say that* HPS *is* perfectly universal$_\kappa$*, if for all* prm *generated by* $\mathsf{SSmpG}(1^\lambda, k)$*, all possible* $\mathsf{prmins} \leftarrow \mathsf{PrmG}(\mathsf{prm})$*, all* $hpk \in \mathcal{PK}$*, all pairwise different* $x_1, \cdots, x_\kappa \in \mathcal{X} \setminus \mathcal{L}$*, and all* $K_1, \cdots, K_\kappa \in \mathcal{K}_{sp}$*,*

$$\Pr\left[\Lambda_{hsk}(x_\kappa) = K_\kappa \,\middle|\, \begin{array}{c} \mu(hsk) = hpk \\ \Lambda_{hsk}(x_1) = K_1, \cdots, \Lambda_{hsk}(x_{\kappa-1}) = K_{\kappa-1} \end{array}\right] = \frac{1}{|\mathcal{K}_{sp}|},$$

where the probability is over $hsk \leftarrow \mathcal{SK}$*.*

Definition 10 (Key Equivocability). *We say that* HPS *is* key equivocable*, if there is a PPT algorithm* SampHsk*, which takes* $(hsk, \mathsf{td}, x_1, \cdots, x_{2k})$ *as input and outputs another secret key* hsk'*, such that for all possible* $(\mathsf{prm}, \mathsf{td}) \leftarrow \mathsf{SSmpG}(1^\lambda, k)$*, all possible* $\mathsf{prmins} = (\mathcal{K}_{sp}, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \mu) \leftarrow \mathsf{PrmG}(\mathsf{prm})$*, all permutations* $\mathsf{P} : [2k] \to [2k]$*, and all* $(x_1, \cdots, x_{2k}) \in \mathcal{X}^{2k}$ *satisfying that* $x_i \in \mathcal{L}_{\mathsf{P}(i)}$*,* $\Delta(\mathsf{Dist}_0, \mathsf{Dist}_1)$ *is negligible, where* Dist_0 *and* Dist_1 *are defined in Fig. 4.*

Dist_0:	Dist_1:
$hsk \leftarrow \mathcal{SK}$; $hpk = \mu(hsk)$	$hsk \leftarrow \mathcal{SK}$; $hpk = \mu(hsk)$
For $i = 1$ to k :	For $i = 1$ to $2k$:
$K_i = \mathsf{SecEv}(hsk, x_i)$	$K_i = \mathsf{SecEv}(hsk, x_i)$
For $i = k+1$ to $2k$:	$hsk' \leftarrow \mathsf{SampHsk}(hsk, \mathsf{td}, x_1, \cdots, x_{2k})$
$K_i \leftarrow \mathcal{K}_{sp}$	Return $(hsk', hpk, K_1, \cdots, K_{2k})$
Return $(hsk, hpk, K_1, \cdots, K_{2k})$	

Fig. 4 Distributions for defining key equivocability of HPS.

Instantiation from DDH. Now we present our instantiation of universal$_\kappa$ HPS with key equivocability from the DDH assumption. The definition of the DDH assumption will be recalled in Appendix A.

Let λ be the security parameter and let k, κ be positive integers that are polynomial in λ. Let $\mathbb{G}$ be a multiplicative cyclic group of prime order q and let g be a generator of $\mathbb{G}$. Let $\Gamma : \mathbb{G}^{2k+1} \to \mathbb{Z}_q^{2k+1}$ be an injective function, which can be extended from the injective function in the constructions of HPS in [8] directly.

We construct a strengthened subset membership problem $\mathsf{SSMP}_1 = (\mathsf{SSmpG}, \mathsf{SSmpX}, \mathsf{SSmpL}, \mathsf{SSmpLS}, \mathsf{SSmpChk})$ as follows:

– SSmpG. On input a security parameter λ and an integer k, the parameter generation algorithm first samples $a_i \leftarrow \mathbb{Z}_q$ and computes $g_i = g^{a_i}$ for $i \in [2k+1]$. Then it sets:

$$\mathcal{X} = \{u_1, \ldots, u_{2k+1} \mid \forall i \in [2k+1], u_i \in \mathbb{G}\}$$

$$\mathcal{L} = \{g_1^w, \ldots, g_{2k+1}^w \mid w \in \mathbb{Z}_q\}$$

and for $i \in [2k]$, it sets:

$$\mathcal{L}_i = \{g_1^{w_1}, \ldots, g_{2k+1}^{w_{2k+1}} \mid w, w' \in \mathbb{Z}_q, w \neq w',$$

$$w_i = w', \forall j \in [2k+1] \backslash \{i\}, w_j = w\}$$

The public parameter $\mathsf{prm} = (\mathbb{G}, q, g, g_1, \ldots, g_{2k+1})$ and the trapdoor $\mathsf{td} = (a_1, \ldots, a_{2k+1})$

- SSmpX. On input a public parameter $\mathsf{prm} = (\mathbb{G}, q, g, g_1, \ldots, g_{2k+1})$, the algorithm samples $u_i \leftarrow \mathbb{G}$ for $i \in [2k+1]$ and outputs $x = (u_1, \ldots, u_{2k+1})$.
- SSmpL. On input a public parameter $\mathsf{prm} = (\mathbb{G}, q, g, g_1, \ldots, g_{2k+1})$, the algorithm samples $w \leftarrow \mathbb{Z}_q$ and outputs $x = (g_1^w, \ldots, g_{2k+1}^w)$ and the witness w.
- SSmpLS. On input a public parameter $\mathsf{prm} = (\mathbb{G}, q, g, g_1, \ldots, g_{2k+1})$ and an integer $i \in [2k]$, the algorithm samples $w \leftarrow \mathbb{Z}_q$ and $w' \leftarrow \mathbb{Z}_q$ s.t. $w \neq w'$. Then it computes $u_j = g_j^w$ for $j \in [2k+1] \backslash \{i\}$ and $u_i = g_i^{w'}$ and outputs $(u_1, \ldots, u_{2k+1})$.
- $\mathsf{SSmpChk}$. On input a public parameter $\mathsf{prm} = (\mathbb{G}, q, g, g_1, \ldots, g_{2k+1})$, a trapdoor $\mathsf{td} = (a_1, \ldots, a_{2k+1})$, and $x = (u_1, \ldots, u_{2k+1})$, the algorithm first computes $v_j = u_j^{a_j^{-1}}$ for $j \in [2k+1]$. It outputs 0 if $v_1 = v_2 = \ldots = v_{2k+1}$. It outputs j if there exists some $j \in [2k]$ s.t. $v_\jmath = v_{\jmath'}$ for all $\jmath, \jmath' \in [2k] \backslash \{j\}$ and $v_j \neq v_{2k+1}$. Otherwise, it outputs $\perp$.

Also, we construct the HPS $\mathsf{HPS}_1 = (\mathsf{PrmG}, \mathsf{PubEv}, \mathsf{SecEv}, \mathsf{SampHsk})$ for SSMP_1 as follows:

- PrmG. On input a public parameter $\mathsf{prm} = (\mathbb{G}, q, g, g_1, \ldots, g_{2k+1})$, the algorithm defines $\mathcal{K}_{sp} = \mathbb{G}$, $\mathcal{SK} = \mathbb{Z}_q^{(2k+1)\times\kappa\times(2k+1)}$, and $\mathcal{PK} = \mathbb{G}^{(2k+1)\times\kappa}$.

  Then for any $hsk = (s_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} \in \mathcal{SK}$ and any $x = (u_1, \ldots, u_{2k+1}) \in \mathcal{X}$, it defines the map Λ from $\mathcal{SK} \times \mathcal{X}$ to $\mathcal{K}_{sp}$ as

$$\Lambda_{hsk}(x) = \prod_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} u_j^{s_{h,i,j}\cdot\alpha_h^{i-1}}$$

  where $(\alpha_1, \ldots, \alpha_{2k+1}) = \Gamma(x)$. Also, for any $hsk = (s_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} \in \mathcal{SK}$, it defines the map μ from $\mathcal{SK}$ to $\mathcal{PK}$ as

$$\mu(hsk) = (p_{h,i})_{h\in[2k+1],i\in[\kappa]} = (\prod_{j\in[2k+1]} g_j^{s_{h,i,j}})_{h\in[2k+1],i\in[\kappa]}$$

- SecEv. On input a secret key $hsk = (s_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} \in \mathcal{SK}$ and $x = (u_1, \ldots, u_{2k+1}) \in \mathcal{X}$, the secret evaluation algorithm outputs $K = \Lambda_{hsk}(x)$.

– PubEv. On input a public key $hpk = (p_{h,i})_{h\in[2k+1],i\in[\kappa]} \in \mathcal{PK}$, $x = (u_1, \ldots, u_{2k+1}) \in \mathcal{L}$ and a witness w, the public evaluation algorithm computes $(\alpha_1, \ldots, \alpha_{2k+1}) = \Gamma(x)$ and outputs $K = \prod_{h\in[2k+1],i\in[\kappa]} p_{h,i}^{w\cdot\alpha_h^{i-1}}$.
– $\mathsf{SampHsk}$. On input a secret key $hsk = (s_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]}$, a trapdoor $\mathsf{td} = (a_1, \ldots, a_{2k+1})$, and $2k$ inputs $(x_\ell = (u_{\ell,1}, \ldots, u_{\ell,2k+1}))_{\ell\in[2k]}$, the algorithm works as follows:
  1. For $\ell \in [2k]$, it computes $\boldsymbol{p}[\ell] = \mathsf{SSmpChk}(\mathsf{prm}, \mathsf{td}, x_\ell)$.
  2. It outputs $\perp$ if there exists $\ell \in [2k]$ s.t. $\boldsymbol{p}[\ell] \notin [2k]$ or there exist distinct $\ell_1, \ell_2 \in [2k]$ s.t. $\boldsymbol{p}[\ell_1] = \boldsymbol{p}[\ell_2]$.
  3. For $h \in [2k+1], i \in [\kappa], j \in \{\boldsymbol{p}[1], \ldots, \boldsymbol{p}[k]\}$, it sets $s'_{h,i,j} = s_{h,i,j}$.
  4. For $h \in [2k+1], i \in [\kappa], j \in \{\boldsymbol{p}[k+1], \ldots, \boldsymbol{p}[2k]\}$, it samples $s'_{h,i,j} \leftarrow \mathbb{Z}_q$.
  5. For $h \in [2k+1], i \in [\kappa]$, it sets $s'_{h,i,2k+1} = (\sum_{j\in[2k+1]} a_j s_{h,i,j} - \sum_{j\in[2k]} a_j s'_{h,i,j}) \cdot a_{2k+1}^{-1}$.
  6. It outputs $hsk' = (s'_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]}$.

Theorem 1. *Assuming the DDH assumption holds,* SSMP_1 *is a strengthened subset membership problem with hardness, sparseness, explainability, and correctness.*

Theorem 2. HPS_1 *is a perfect universal*$_\kappa$ *HPS with key equivocability.*

Proofs of Theorem 1 and Theorem 2 are provided in Appendix D.

Instantiation from DCR. We present our instantiation of universal$_\kappa$ HPS with key equivocability from the DCR assumption as follows. The definition of the DCR assumption will be recalled in Appendix A.

Let λ be the security parameter and let k, κ be positive integers that are polynomial in λ. We construct a strengthened subset membership problem $\mathsf{SSMP}_2 = (\mathsf{SSmpG}, \mathsf{SSmpX}, \mathsf{SSmpL}, \mathsf{SSmpLS}, \mathsf{SSmpChk})$ as follows:

– SSmpG. On input a security parameter λ and an integer k, the parameter generation algorithm first samples primes p', q', p, q s.t. $p = 2p' + 1$ and $q = 2q' + 1$. Then it computes $N = pq$ and $N' = p'q'$. Let $\mathbb{Z}_{N^2}^* = \mathbb{G}_N \cdot \mathbb{G}_{N'} \cdot \mathbb{G}_2 \cdot \mathbb{T}$, where $\mathbb{G}_N, \mathbb{G}_{N'}, \mathbb{G}_2, \mathbb{T}$ are defined as in Appendix A. Define $\mathbb{X} = \mathbb{G}_N \cdot \mathbb{G}_{N'} \cdot \mathbb{T}$ and $\mathbb{L} = \mathbb{G}_{N'} \cdot \mathbb{T}$. Define $\chi : \mathbb{Z}_{N^2} \to \mathbb{Z}_N$ as $\chi(a) = \lfloor a/N \rfloor$. Let $\Gamma : \mathbb{X}^{2k} \to \mathbb{Z}_{\lfloor N^2/2 \rfloor}^{2k}$ be an injective function, which can be extended from the injective function in the constructions of HPS in [8] directly. Also, let $g \in \mathbb{Z}_{N^2}^*$ be a fixed generator of $\mathbb{L}$.

Then it sets:

$$\mathcal{X} = \{u_1, \ldots, u_{2k} \mid \forall j \in [2k], u_j \in \mathbb{X}\},$$

$$\mathcal{L} = \{g^{r_1}, \ldots, g^{r_{2k}} \mid \forall j \in [2k], r_j \in \mathbb{Z}_{2N'}\},$$

and for $i \in [2k]$, it sets:

$$\mathcal{L}_i = \{u_1, \ldots u_{2k} \mid u_i \in \mathbb{X}\backslash\mathbb{L}, \forall j \in [2k]\backslash\{i\}, r_j \in \mathbb{Z}_{2N'}, u_j = g^{r_j}\}.$$

The public parameter $\mathsf{prm} = (N, g)$ and the trapdoor $\mathsf{td} = N'$.

- SSmpX. On input a public parameter $\mathsf{prm} = (N, g)$, the algorithm samples $u_j \leftarrow \mathbb{X}$ for $j \in [2k]$ and outputs $x = (u_1, \ldots, u_{2k})$.
- SSmpL. On input a public parameter $\mathsf{prm} = (N, g)$, the algorithm samples $r_j \leftarrow \mathbb{Z}_{\lfloor N/2 \rfloor}$ for $j \in [2k]$ and outputs $x = (g^{r_1}, \ldots, g^{r_{2k}})$ and the witness $(r_1, \ldots, r_{2k})$.
- SSmpLS. On input a public parameter $\mathsf{prm} = (N, g)$ and an integer $i \in [2k]$, the algorithm samples $r_j \leftarrow \mathbb{Z}_{\lfloor N/2 \rfloor}$ for $j \in [2k]\backslash\{i\}$ and $u_i \leftarrow \mathbb{X}$. Then it computes $u_j = g^{r_j}$ for $j \in [2k]\backslash\{i\}$ and outputs $x = (u_1, \ldots, u_{2k})$.
- $\mathsf{SSmpChk}$. On input a public parameter $\mathsf{prm} = (N, g)$, a trapdoor $\mathsf{td} = N'$, and $x = (u_1, \ldots, u_{2k})$, the algorithm first computes $v_j = u_j^{2N'}$ for $j \in [2k]$. It outputs 0 if $v_1 = v_2 = \ldots v_{2k} = 1$. It outputs j if there exists $j \in [2k]$ s.t. $v_\jmath = 1$ for all $\jmath \in [2k]\backslash\{j\}$ and $v_j \neq 1$. Otherwise, it outputs $\perp$.

Also, we construct the HPS $\mathsf{HPS}_2 = (\mathsf{PrmG}, \mathsf{PubEv}, \mathsf{SecEv}, \mathsf{SampHsk})$ for SSMP_2 as follows:

- PrmG. On input a public parameter $\mathsf{prm} = (N, g)$, the algorithm defines $\mathcal{K}_{SP} = \mathbb{Z}_N$, $\mathcal{SK} = \mathbb{Z}_{\lfloor N^2/2 \rfloor}^{(2k)\times(\kappa)\times(2k)}$, and $\mathcal{PK} = \mathbb{L}^{(2k)\times(\kappa)\times(2k)}$. Then for any $hsk = (s_{h,i,j})_{h\in[2k],i\in[\kappa],j\in[2k]} \in \mathcal{SK}$ and any $x = (u_1, \ldots, u_{2k}) \in \mathcal{X}$, it defines the map Λ from $\mathcal{SK} \times \mathcal{X}$ to $\mathcal{K}_{sp}$ as

$$\Lambda_{hsk}(x) = \chi\Big(\prod_{h\in[2k],i\in[\kappa],j\in[2k]} u_j^{s_{h,i,j}\cdot\alpha_h^{i-1}}\Big)$$

  where $(\alpha_1, \ldots, \alpha_{2k}) = \Gamma(x)$. Also, for any $hsk = (s_{h,i,j})_{h\in[2k],i\in[\kappa],j\in[2k]} \in \mathcal{SK}$, it defines the map μ from $\mathcal{SK}$ to $\mathcal{PK}$ as

$$\mu(sk) = (p_{h,i,j})_{h\in[2k],i\in[\kappa],j\in[2k]} = (g^{s_{h,i,j}})_{h\in[2k],i\in[\kappa],j\in[2k]}.$$

– SecEv. On input a secret key $hsk = (s_{h,i,j})_{h\in[2k],i\in[\kappa],j\in[2k]} \in \mathcal{SK}$ and $x = (u_1, \ldots, u_{2k}) \in \mathcal{X}$, the secret evaluation algorithm outputs $K = \Lambda_{hsk}(x)$.
– PubEv. On input a public key $hpk = (p_{h,i,j})_{h\in[2k],i\in[\kappa],j\in[2k]} \in \mathcal{PK}$, $x = (u_1, \ldots, u_{2k}) \in \mathcal{L}$ and a witness $(r_1, \ldots, r_{2k})$, the public evaluation algorithm computes $(\alpha_1, \ldots, \alpha_{2k}) = \Gamma(x)$ and outputs $K = \chi(\prod_{h\in[2k],i\in[\kappa],j\in[2k]} p_{h,i,j}^{r_j\cdot\alpha_h^{i-1}})$.
– $\mathsf{SampHsk}$. On input a secret key $hsk = (s_{h,i,j})_{h\in[2k],i\in[\kappa],j\in[2k]}$, a trapdoor $\mathsf{td} = N'$, and $2k$ inputs $(x_\ell = (u_{\ell,1}, \ldots, u_{\ell,2k}))_{\ell\in[2k]}$, the algorithm works as follows:
  1. For $\ell \in [2k]$, it computes $\boldsymbol{p}[\ell] = \mathsf{SSmpChk}(\mathsf{prm}, \mathsf{td}, x_\ell)$.
  2. It outputs $\perp$ if there exists $\ell \in [2k]$ s.t. $\boldsymbol{p}[\ell] \notin [2k]$ or there exist distinct $\ell_1, \ell_2 \in [2k]$ s.t. $\boldsymbol{p}[\ell_1] = \boldsymbol{p}[\ell_2]$.
  3. For $h \in [2k], i \in [\kappa], j \in \{\boldsymbol{p}[1], \ldots, \boldsymbol{p}[k]\}$, it sets $s'_{h,i,j} = s_{h,i,j}$.
  4. For $h \in [2k], i \in [\kappa], j \in \{\boldsymbol{p}[k+1], \ldots, \boldsymbol{p}[2k]\}$, it samples $t \leftarrow \mathbb{Z}_N$ and uses the Chinese remainder theorem to compute $s'_{h,i,j} \in \mathbb{Z}_{2NN'}$ s.t. $s'_{h,i,j} = t \mod N$ and $s'_{h,i,j} = s_{h,i,j} \mod 2N'$.
  5. It outputs $hsk' = (s'_{h,i,j})_{h\in[2k],i\in[\kappa],j\in[2k]}$.

Theorem 3. *Assuming the DCR assumption holds,* SSMP_2 *is a strengthened subset membership problem with hardness, sparseness, explainability, and correctness.*

Theorem 4. HPS_2 *is a perfect universal*$_\kappa$ *HPS with key equivocability.*

Proofs of Theorem 3 and Theorem 4 are similar to proofs of Theorem 1 and Theorem 2. So, we omit the details here. Note that SSMP_2 only achieves a statistical sampling correctness while SSMP_1 achieves a perfect sampling correctness.

4.3 SIM-wBi-SO$_k$-CCA Secure PKE Construction

For any polynomially bounded function $k > 0$, we propose a PKE scheme achieving SIM-wBi-SO$_k$-CCA security. Our construction is built from a perfectly universal$_{k+1}$ HPS with key-equivocability, and a strong and semi-unique XAC. The details are as follows.

Let $\mathsf{SSMP} = (\mathsf{SSmpG}, \mathsf{SSmpX}, \mathsf{SSmpL}, \mathsf{SSmpLS}, \mathsf{SSmpChk})$ be a hard SSMP. Let $\mathsf{HPS} = (\mathsf{PrmG}, \mathsf{PubEv}, \mathsf{SecEv}, \mathsf{SampHsk})$ be a perfectly universal$_{k+1}$ and key equivocable HPS for SSMP, such that all the $\mathcal{K}_{sp}$ generated by PrmG can be written as $\mathcal{K}_a \times \mathcal{K}_b$. For $\ell \in \mathbb{N}$ and any $\mathsf{prmins} = (\mathcal{K}_{sp}, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \mu)$ generated by PrmG, let $\mathsf{XAC}_{\mathsf{prmins}} = (\mathsf{XGen}, \mathsf{XAuth}, \mathsf{XVer}, \mathsf{ReSamp})$ be a strong and semi-unique $(\ell+1)$-XAC with key space $\mathcal{XK} =$

$\mathcal{K}_{sp} = \mathcal{K}_a \times \mathcal{K}_b$ and tag space $\mathcal{XT}$, and $\mathcal{H}_{\mathsf{prmins}} : (\mathcal{X} \times \mathcal{PK})^\ell \to \mathcal{K}_b$ be a family of collision-resistant hash functions. Our PKE scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (for ℓ-bit messages) is defined in Fig. 5.

$\underline{\mathsf{Setup}(1^\lambda)}$:
$(\mathsf{prm} := (\mathcal{X}, \mathcal{L}, \mathcal{L}_1, \cdots, \mathcal{L}_{2k}), \mathsf{td}) \leftarrow \mathsf{SSmpG}(1^\lambda, k)$
$\mathsf{prmins} = (\mathcal{K}_{sp} = \mathcal{K}_a \times \mathcal{K}_b, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \mu) \leftarrow \mathsf{PrmG}(\mathsf{prm})$; $\mathsf{H} \leftarrow \mathcal{H}_{\mathsf{prmins}}$; $K_a \leftarrow \mathcal{K}_a$
Return $\mathsf{pp} := (\mathsf{prm}, \mathsf{prmins}, \mathsf{H}, K_a)$

$\underline{\mathsf{Gen}(\mathsf{pp})}$:
Parse $\mathsf{prmins} = (\mathcal{K}_{sp}, \mathcal{SK}, \mathcal{PK}, \mathcal{T}, \Lambda_{(\cdot)}, \mu)$
$(hsk_\gamma)_{\gamma \in [\ell]} \leftarrow (\mathcal{SK})^\ell$; $(hpk_\gamma = \mu(hsk_\gamma))_{\gamma \in [\ell]}$; $pk := (hpk_\gamma)_{\gamma \in [\ell]}$; $sk := (hsk_\gamma)_{\gamma \in [\ell]}$
Return (pk, sk)

$\underline{\mathsf{Enc}(pk = (hpk_\gamma)_{\gamma \in [\ell]}, m)}$:
Parse $m = (m_1, \cdots, m_\ell) \in \{0,1\}^\ell$
$r := (r_\gamma^{(\mathcal{X})}, r_\gamma^{(\mathcal{K})}, w_\gamma)_{\gamma \in [\ell]} \leftarrow (\mathcal{R}_{\mathsf{SSmpX}} \times \mathcal{R}_{\mathsf{Sample}} \times \mathcal{R}_{\mathsf{SSmpL}})^\ell$
For $\gamma = 1$ to ℓ :
  If $m_\gamma = 0$: $x_\gamma \leftarrow \mathsf{SSmpX}(\mathsf{prm}; r_\gamma^{(\mathcal{X})})$; $K_\gamma \leftarrow \mathsf{Sample}(\mathcal{K}_{sp}; r_\gamma^{(\mathcal{K})})$
  If $m_\gamma = 1$: $x_\gamma \leftarrow \mathsf{SSmpL}(\mathsf{prm}; w_\gamma)$; $K_\gamma = \mathsf{PubEv}(hpk_\gamma, x_\gamma, w_\gamma)$
$K_b = \mathsf{H}(pk, x_1, \cdots, x_\ell)$; $K_{\ell+1} = (K_a, K_b)$; $T = \mathsf{XAuth}(K_1, \cdots, K_{\ell+1})$
Return $c = (x_1, \cdots, x_\ell, T)$

$\underline{\mathsf{Dec}(sk = (hsk_\gamma)_{\gamma \in [\ell]}, c = (x_1, \cdots, x_\ell, T))}$:
$\overline{K}_b = \mathsf{H}(pk, x_1, \cdots, x_l)$
If $\mathsf{XVer}((K_a, \overline{K}_b), T) = 0$: $\overline{m}_1 = \cdots = \overline{m}_\ell = 0$; return $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$
For $\gamma = 1$ to ℓ :
  $\overline{K}_\gamma = \mathsf{SecEv}(hsk_\gamma, x_\gamma)$; $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T)$
Return $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$

Fig. 5 Construction of PKE.

Correctness. For $\gamma \in [\ell]$, if $m_\gamma = 1$, then $\overline{K}_\gamma = K_\gamma$ by completeness of HPS, so $\overline{m}_\gamma = \mathsf{Xver}(\overline{K}_\gamma, \gamma, T) = 1$ except with probability $\mathsf{fail}_{\mathsf{XAC}}(\lambda)$ by correctness of XAC. On the other hand, if $m_\gamma = 0$, subset sparseness of SSMP and perfect universality of HPS guarantee that with overwhelming probability, $\overline{K}_\gamma$ is uniformly random, even given pk, c and m. In this case, $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T) = 0$ except with probability $\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda)$. So, correctness of PKE follows by a union bound over $\gamma \in [\ell]$.

Security. Formally, we have the following theorem, the formal proof of which will be given in Appendix E.

Theorem 5. *For any polynomial function $k > 0$,* PKE *is SIM-wBi-SO$_k$-CCA secure.*

5 PKE with SIM-Bi-SO-CCA Security

In [15], Heuer et al. showed that a generic construction of DHIES [32] meets SIM-SSO-CCA security in the random oracle model. In this section, we show that a variant of the generic construction actually achieves SIM-Bi-SO-CCA security in the random oracle model.

Building blocks. We simply recall the definitions of key encapsulation mechanism (KEM) and message authentication code (MAC) as follows.

Key Encapsulation Mechanism. A KEM scheme, associated with a session key space $\mathcal{K}_{\mathsf{KEM}}$ and a ciphertext space $\mathcal{C}_{\mathsf{KEM}}$, is a tuple of PPT algorithms $\mathsf{KEM} = (\mathsf{KemGen}, \mathsf{Encap}, \mathsf{Decap})$. The key generation algorithm KemGen takes 1^λ as input, and outputs a public/secret key pair (pk, sk). The encapsulation algorithm Encap takes pk as input, outputs $(K, c) \in \mathcal{K}_{\mathsf{KEM}} \times \mathcal{C}_{\mathsf{KEM}}$. The decapsulation algorithm Decap, taking sk and c as input, outputs a value in $\mathcal{K}_{\mathsf{KEM}} \cup \{\perp\}$. Standard correctness is required. Similar to [15], without loss of generality we assume that Encap uniformly samples $K \leftarrow \mathcal{K}_{\mathsf{KEM}}$. We also assume that $|\mathcal{K}_{\mathsf{KEM}}| \geq 2^\lambda$ and $|\mathcal{C}_{\mathsf{KEM}}| \geq 2^\lambda$.

We say that KEM has *unique encapsulations*, if for any (pk, sk) generated by KemGen, and for any ciphertexts c, c' satisfying $\mathsf{Decap}(sk, c) = \mathsf{Decap}(sk, c') \neq \perp$, $c = c'$.

The security notion, one-way security in the presence of a plaintext-checking oracle (OW-PCA security) [29], is recalled in Appendix F.

Message Authentication Code. A MAC scheme, associated with a key space $\mathcal{K}_{\mathsf{MAC}}$, is a tuple of PPT algorithms $\mathsf{MAC} = (\mathsf{MacGen}, \mathsf{Auth}, \mathsf{Verf})$. The key generation algorithm MacGen takes 1^λ as input and outputs a key $K \in \mathcal{K}_{\mathsf{MAC}}$. The authentication algorithm Auth takes K and a message m as input, outputs a tag t. On input (K, m, t), the verification algorithm Verf outputs a bit $b' \in \{0, 1\}$. Standard correctness is also required here.

MAC is called deterministic, if Auth is deterministic. For a deterministic MAC, MAC is called *injective*, if Auth is an injective function (i.e., for any $K \in \mathcal{K}_{\mathsf{MAC}}$ and any $m \neq m'$, $\mathsf{Auth}(K, m) \neq \mathsf{Auth}(K, m')$).

The security notion of strong unforgeability under one-time chosen message attacks (sUF-OT-CMA security) is recalled in Appendix F.

PKE Construction. Let $\mathsf{KEM} = (\mathsf{KemGen}, \mathsf{Encap}, \mathsf{Decap})$ be an OW-PCA secure KEM scheme, having unique encapsulations, associated with

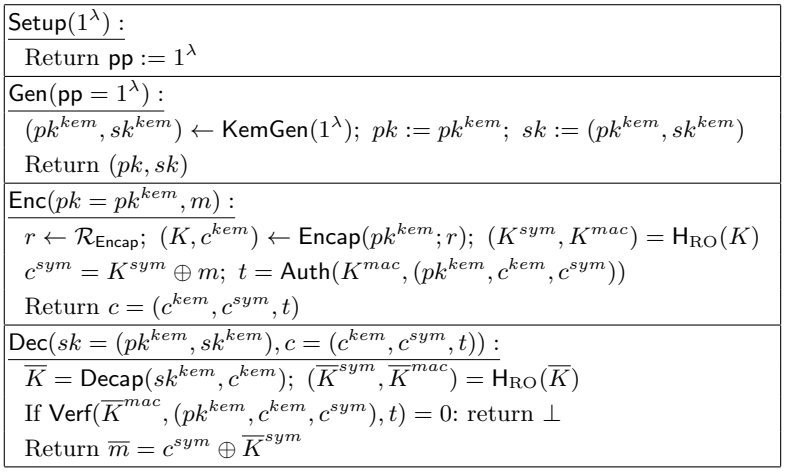

Fig. 6 Construction of $\mathsf{PKE}_{\mathsf{K\text{-}M}}$.

a session key space $\mathcal{K}_{\mathsf{KEM}}$ and a ciphertext space $\mathcal{C}_{\mathsf{KEM}}$, where Encap uniformly samples K, $|\mathcal{K}_{\mathsf{KEM}}| \geq 2^\lambda$ and $|\mathcal{C}_{\mathsf{KEM}}| \geq 2^\lambda$. Let $\mathsf{MAC} = (\mathsf{MacGen}, \mathsf{Auth}, \mathsf{Verf})$ be a deterministic, injective MAC scheme, associated with a key space $\mathcal{K}_{\mathsf{MAC}}$, achieving sUF-OT-CMA security. Let $\mathsf{H}_{\mathrm{RO}} : \mathcal{K}_{\mathsf{KEM}} \to \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$ be a hash function. Our PKE scheme $\mathsf{PKE}_{\mathsf{K\text{-}M}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, associated with a message space $\{0,1\}^\ell$, is defined in Fig. 6.

The correctness analysis of this scheme is trivial. Now we turn to its security analysis. Formally, we have the following theorem. Note that, in our construction, a valid ciphertext contains a tag t generated on $(pk^{kem}, c^{kem}, c^{sym})$, where in [15], the tag t is only generated on c^{sym}. We stress that this crucial modification makes our construction achieve SIM-Bi-SO-CCA security. The intuition for the security proof and details are given in Appendix G.

Theorem 6. *If* KEM *has unique encapsulations and is OW-PCA secure,* MAC *is deterministic, injective and sUF-OT-CMA secure, and* H_{RO} *is modeled as a random oracle, then* $\mathsf{PKE}_{\mathsf{K\text{-}M}}$ *is SIM-Bi-SO-CCA secure in the random oracle model.*

Acknowledgment. We thank Fangguo Zhang for the helpful discussions. We appreciate the anonymous reviewers for their valuable comments.

References

1. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT*, pages

259–274. Springer, 2000.
2. Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In *EUROCRYPT*, pages 645–662. Springer, 2012.
3. Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35. Springer, 2009.
4. Mihir Bellare and Scott Yilek. Encryption schemes secure under selective opening attack. Cryptology ePrint Archive, Report 2009/101, 2009. https://eprint.iacr.org/2009/101.
5. Xavier Boyen and Qinyi Li. All-but-many lossy trapdoor functions from lattices and applications. In *CRYPTO*, pages 298–331. Springer, 2017.
6. Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In *TCC*, pages 150–168. Springer, 2005.
7. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25. Springer, 1998.
8. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64. Springer, 2002.
9. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
10. Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In *EUROCRYPT*, pages 381–402. Springer, 2010.
11. Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
12. Keisuke Hara, Fuyuki Kitagawa, Takahiro Matsuda, Goichiro Hanaoka, and Keisuke Tanaka. Simulation-based receiver selective opening cca secure pke from standard computational assumptions. In *Security and Cryptography for Networks*, pages 140–159. Springer, 2018.
13. Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In *ASIACRYPT*, pages 443–469. Springer, 2015.
14. Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *ASIACRYPT*, pages 70–88. Springer, 2011.
15. Felix Heuer, Tibor Jager, Eike Kiltz, and Sven Schäge. On the selective opening security of practical public-key encryption schemes. In *PKC*, pages 27–51. Springer, 2015.
16. Felix Heuer and Bertram Poettering. Selective opening security from simulatable data encapsulation. In *ASIACRYPT*, pages 248–277. Springer, 2016.
17. Dennis Hofheinz. All-but-many lossy trapdoor functions. In *EUROCRYPT*, pages 209–227. Springer, 2012.
18. Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In *TCC*, pages 121–145. Springer, 2016.
19. Dennis Hofheinz and Andy Rupp. Standard versus selective opening security: Separation and equivalence results. In *TCC*, pages 591–615. Springer, 2014.

20. Zhengan Huang, Junzuo Lai, Wenbin Chen, Man Ho Au, Zhen Peng, and Jin Li. Simulation-based selective opening security for receivers under chosen-ciphertext attacks. *Designs, Codes and Cryptography*, 87(6):1345–1371, 2019.
21. Zhengan Huang, Shengli Liu, and Baodong Qin. Sender-equivocable encryption schemes secure against chosen-ciphertext attacks revisited. In *PKC*, pages 369–385. Springer, 2013.
22. Dingding Jia and Benoît Libert. So-cca secure pke from pairing based all-but-many lossy trapdoor functions. *Designs, Codes and Cryptography*, 89(5):895–923, 2021.
23. Dingding Jia, Xianhui Lu, and Bao Li. Receiver selective opening security from indistinguishability obfuscation. In *INDOCRYPT*, pages 393–410. Springer, 2016.
24. Dingding Jia, Xianhui Lu, and Bao Li. Constructions secure against receiver selective opening and chosen ciphertext attacks. In *CT-RSA*, pages 417–431. Springer, 2017.
25. Junzuo Lai, Robert H. Deng, Shengli Liu, Jian Weng, and Yunlei Zhao. Identity-based encryption secure against selective opening chosen-ciphertext attack. In *EUROCRYPT*, pages 77–92. Springer, 2014.
26. Benoît Libert, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from lwe. In *CRYPTO*, pages 332–364. Springer, 2017.
27. Shengli Liu and Kenneth G. Paterson. Simulation-based selective opening cca security for pke from key encapsulation mechanisms. In *PKC*, pages 3–26. Springer, 2015.
28. Lin Lyu, Shengli Liu, Shuai Han, and Dawu Gu. Tightly sim-so-cca secure public key encryption from standard assumptions. In *PKC*, pages 62–92. Springer, 2018.
29. Tatsuaki Okamoto and David Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA*, pages 159–174. Springer, 2001.
30. Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In *CRYPTO*, pages 525–542. Springer, 2011.
31. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238. Springer, 1999.
32. Ron Steinfeld, Joonsang Baek, and Yuliang Zheng. On the necessity of strong assumptions for the security of a class of asymmetric encryption schemes. In *ACISP*, pages 241–256. Springer, 2002.
33. Rupeng Yang, Junzuo Lai, Zhengan Huang, Man Ho Au, Qiuliang Xu, and Willy Susilo. Possibility and impossibility results for receiver selective opening secure pke in the multi-challenge setting. In *ASIACRYPT*, pages 191–220. Springer, 2020.

A Cryptographic Assumptions

The DDH Assumption. Let $\mathbb{G}$ be a cyclic group of prime order q with a generator g. The DDH assumption requires that it is hard to distinguish (g^a, g^b, g^c) and (g^a, g^b, g^{ab}), where $a, b, c \leftarrow \mathbb{Z}_q$.

The DCR Assumption. Now, we recall the Decision Composite Residuosity (DCR) assumption [31] and some useful facts about it shown in [8].

Let p, q, p', q' be primes such that $p = 2p' + 1$ and $q = 2q' + 1$. Let $N = pq$ and $N' = p'q'$. Then the group $\mathbb{Z}_{N^2}^*$ can be decomposed as the

direct product $\mathbb{G}_N \cdot \mathbb{G}_{N'} \cdot \mathbb{G}_2 \cdot \mathbb{T}$, where $\mathbb{G}_{N'}$ and $\mathbb{G}_2$ are cyclic groups of order N' and order 2 respectively; $\mathbb{G}_N$ is a cyclic group of order N generated by $\xi = (1+N) \mod N^2$; and $\mathbb{T}$ is the order-2 subgroup of $\mathbb{Z}^*_{N^2}$ generated by $(-1 \mod N^2)$. Note that $\xi^a = (1+aN) \mod N^2$ for $a \in \{0, 1, \cdots, N\}$.

The DCR assumption requires that it is hard to distinguish a random element in $\mathbb{Z}^*_{N^2}$ and a random element in $\mathbb{G}_{N'} \cdot \mathbb{G}_2 \cdot \mathbb{T}$.

Next, define $\mathbb{X} = \mathbb{G}_N \cdot \mathbb{G}_{N'} \cdot \mathbb{T}$. The set $\mathbb{X}$ is an efficiently samplable and explainable domain, where the sample algorithm and the explain algorithm work as follows:

- Sample: The sample algorithm proceeds as follows:
  1. For $i \in [1, 160]$:
     (a) $x \leftarrow \mathbb{Z}_{N^2}$
     (b) If the Jacobi symbol $(\frac{x}{N}) = 1$: **output** x.
  2. **Output** $\perp$.
- Explain: on input an element $x \in \mathbb{X}$, the explain algorithm proceeds as follows:
  1. Set r to be an empty string.
  2. For $i \in [1, 160]$:
     (a) Sample $b \leftarrow \{0, 1\}$.
     (b) If $b = 1$, append x to r and **outputs** r.
     (c) Otherwise, sample an element $x' \leftarrow \mathbb{Z}_{N^2}$ s.t. the Jacobi symbol $(\frac{x'}{N}) = -1$ and append x' to r.
  3. **Output** $\perp$.

Note that as $\frac{|\mathbb{X}|}{|\mathbb{Z}^*_{N^2}|} = 1/2$, the expected repetition in the sample algorithm is about 2 and the probability that the sample algorithm outputs $\perp$ is $\frac{1}{2^{160}}$, which is negligible. Also, it is easy to see the probability that the explain algorithm outputs $\perp$ is also $\frac{1}{2^{160}}$, which is negligible.

Also, define $\chi : \mathbb{Z}_{N^2} \to \mathbb{Z}_N$ as $\chi(a) = \lfloor a/N \rfloor$. For any fixed $x \in \mathbb{X}$, $\chi(x\xi^c)$ is uniform in $\mathbb{Z}_N$ if $c \leftarrow \mathbb{Z}_N$.

Finally, define $\mathbb{L} = \mathbb{G}_{N'} \cdot \mathbb{T}$. It is easy to create a generator g for $\mathbb{L}$ by first sampling a random element $\mu \in \mathbb{Z}^*_{N^2}$ and then computing $g = -\mu^{2N}$. Besides, the DCR assumption implies that a random element in $\mathbb{X}$ is computationally indistinguishable from a random element in $\mathbb{L}$.

B The Definition of Decryption Verifiability

We recall the definition of decryption verifiability for PKE, which is proposed in [2].

Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme. A decryption verifier for PKE is a deterministic polynomial-time algorithm $\mathcal{V}$, which takes $(1^\lambda, \mathsf{pp}, pk, sk, c, m)$ as input and returns a single bit $b \in \{0,1\}$. We require that for all $\lambda \in \mathbb{N}$, all pp generated by $\mathsf{Setup}(1^\lambda)$, all (pk, sk) generated by $\mathsf{Gen}(\mathsf{pp})$, all $r \in \mathcal{R}_{\mathsf{Enc}}$ and all valid message $m \in \{0,1\}^*$,

$$\mathcal{V}(1^\lambda, \mathsf{pp}, pk, sk, \mathsf{Enc}(pk, m; r), m) = 1.$$

$\mathcal{V}$ is called *canonical* if $\mathcal{V}(1^\lambda, \mathsf{pp}, pk, sk, \mathsf{Enc}(pk, m; r), m)$ always returns $(\mathsf{Dec}(sk, \mathsf{Enc}(pk, m; r)) = m)$.

Definition 11. (Decryption verifiability)[2]. *We say that a PKE scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* decryption-verifiable *with decryption verifier* $\mathcal{V}$*, if for any PPT adversary* $\mathcal{A}$*,*

$$\mathtt{Adv}^{\text{Dec-Ver}}_{\mathsf{PKE},\mathcal{V},\mathcal{A}}(\lambda) := \Pr[\mathtt{Exp}^{\text{Dec-Ver}}_{\mathsf{PKE},\mathcal{V},\mathcal{A}}(\lambda) = 1]$$

is negligible, where $\mathtt{Exp}^{\text{Dec-Ver}}_{\mathsf{PKE},\mathcal{V},\mathcal{A}}(\lambda)$ *is defined in Fig. 7.*

$\underline{\mathtt{Exp}^{\text{Dec-Ver}}_{\mathsf{PKE},\mathcal{V},\mathcal{A}}(\lambda)}$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(pk, c, m_0, m_1, sk_0, sk_1) \leftarrow \mathcal{A}(\mathsf{pp})$
$b_0 \leftarrow \mathcal{V}(1^\lambda, \mathsf{pp}, pk, sk_0, c, m_0)$; $b_1 \leftarrow \mathcal{V}(1^\lambda, \mathsf{pp}, pk, sk_1, c, m_1)$
Return $((b_0 = 1) \wedge (b_1 = 1) \wedge (m_0 \neq m_1))$

Fig. 7 Experiment for defining decryption verifiability (with decryption verifier $\mathcal{V}$) for PKE.

Bellare et al. [2, Theorem 5.1] pointed out that any decryption-verifiable PKE scheme is not SIM-RSO-CPA secure. We recall their conclusion as follows.

Theorem 7. *[2, Theorem 5.1] Let* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a PKE scheme. Assuming the existence of a family of collision-resistant hash functions* $\mathcal{H}$*, there exists a PPT adversary* $\mathcal{A}'$*, such that for any PPT simulator* $\mathcal{S}$*, there is a PPT distinguisher* $\mathcal{D}'$ *satisfying that* $\mathtt{Adv}^{\text{RSO-CCA}}_{\mathsf{PKE},\mathcal{H},\mathcal{A}',\mathcal{S},\mathcal{D}'}(\lambda) \geq 1 - \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}(\lambda)$*.*[9]

[9] The advantage $\mathtt{Adv}^{\text{RSO-CCA}}_{\mathsf{PKE},\mathcal{H},\mathcal{A}',\mathcal{S},\mathcal{D}'}(\lambda)$, whose subscript includes $\mathcal{H}$, is the advantage of the SIM-RSO-CCA adversary $\mathcal{A}'$ in the auxiliary input model [2]. We refer the readers to [2] for the formal definition and the details.

C The κ-Linear-Based Liu-Paterson PKE Scheme in [27]

In this appendix, we recall the κ-Linear-Based SIM-SSO-CCA secure P-KE scheme, which was proposed by Liu and Paterson in [27].

Let $\mathbb{G}$ be a cyclic group of prime order q and g be a generator of $\mathbb{G}$. Let $F : \mathbb{G}^{\ell} \to (\mathcal{K}_b)^s$ be an injective function, where $s \in \mathbb{N}$ and $\mathcal{K}_b$ is some key space. Let $\mathsf{XAC} = (\mathsf{XGen}, \mathsf{XAuth}, \mathsf{XVer}, \mathsf{ReSamp})$ be a strong and semi-unique $(\ell + s)$-XAC with key space $\mathcal{XK} = \mathcal{K}_{sp} = \mathcal{K}_a \times \mathcal{K}_b$ and tag space $\mathcal{XT}$. Let $\mathcal{H}_{\mathsf{TCR}} : \mathbb{G}^{\kappa} \to \mathbb{Z}_q$ be a family of target collision-resistant hash functions for some $\kappa \in \mathbb{N}$. The κ-Linear-based Liu-Paterson scheme $\mathsf{PKE}_{\mathrm{LP}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (for ℓ-bit messages) [27] is recalled in Fig. 8.

D Deterred Proofs for the Instantiation in Sec. 4.2

We provide the proof for Theorem 1 and Theorem 2 in this section.

Proof. For any $i \in [2k]$, indistinguishability between a uniform element in $\mathcal{L}$ and a uniform element in $\mathcal{L}_i$ comes from the DDH assumption directly; also, indistinguishability between a uniform element in $\mathcal{X}$ and a uniform element in $\mathcal{L}_i$ can be reduced to the DDH assumption by a simple hybrid argument. For a uniform x in $\mathcal{X}$, the probability that it falls in $\mathcal{L}$ is only $1/q^{2k}$, which is negligible. It is easy to see that $\mathcal{X}$ and $\mathcal{K}_{sp}$ are efficiently samplable and explainable. Also, it is easy to check that $\mathsf{SSmpChk}$ and the sampling algorithms are correct. Thus, SSMP_1 is a strengthened subset membership problem with hardness, sparseness, explainability, and correctness.

Next, we prove that HPS_1 is a projective and perfect universal$_{\kappa}$ HPS with key equivocability.

Projective. For any $hsk = (s_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} \in \mathcal{SK}$ and any $x = (g_1^w, g_2^w, \ldots, g_{2k+1}^w) \in \mathcal{L}$, let $hpk = (p_{h,i})_{h\in[2k+1],i\in[\kappa]} = \mu(hsk)$ and

$\underline{\mathsf{Setup}(1^\lambda):}$
$\mathsf{H}_{\mathrm{TCR}} \leftarrow \mathcal{H}_{\mathsf{TCR}}$; $(K_{a_1}, \cdots, K_{a_s}) \leftarrow (\mathcal{K}_a)^s$
Return $\mathsf{pp} := (F, \mathsf{H}_{\mathrm{TCR}}, (K_{a_1}, \cdots, K_{a_s}), (\mathbb{G}, q, g))$

$\underline{\mathsf{Gen}(\mathsf{pp}):}$
$y \leftarrow \mathbb{Z}_q$; $h := g^y$
For $\theta = 1$ to κ :
$(x_\theta, \alpha_\theta, \beta_\theta) \leftarrow (\mathbb{Z}_q)^3$; $g_\theta := g^{x_\theta}$; $w_\theta := x_\theta^{-1} y$; $u_\theta := g_\theta^{\alpha_\theta}$; $v_\theta := g_\theta^{\beta_\theta}$
$pk := (h, (g_\theta, u_\theta, v_\theta)_{\theta \in [\kappa]})$; $sk := ((\alpha_\theta, \beta_\theta, w_\theta)_{\theta \in [\kappa]}, pk)$
Return (pk, sk)

$\underline{\mathsf{Enc}(pk = (h, (g_\theta, u_\theta, v_\theta)_{\theta \in [\kappa]}), m):}$
Parse $m = (m_1, \cdots, m_\ell) \in \{0,1\}^\ell$
For $\gamma = 1$ to ℓ :
If $m_\gamma = 0$:
$K_\gamma \leftarrow \mathcal{K}$; $(r_{\gamma,1}, \cdots, r_{\gamma,\kappa+1}) \leftarrow (\mathbb{Z}_q)^\kappa$; $\psi_\gamma := (g^{r_{\gamma,1}}, \cdots, g^{r_{\gamma,\kappa+1}})$
If $m_\gamma = 1$:
For $\theta = 1$ to κ : $r_{\gamma,\theta} \leftarrow \mathbb{Z}_q$; $c_{\gamma,\theta} := g_\theta^{r_{\gamma,\theta}}$
$t_\gamma := \mathsf{H}_{\mathrm{TCR}}(c_{\gamma,1}, \cdots, c_{\gamma,\kappa})$; $\pi_\gamma := \prod_{\theta=1}^{\kappa} (u_\theta^{t_\gamma} v_\theta)^{r_{\gamma,\theta}}$
$K_\gamma := h^{r_{\gamma,1} + \cdots + r_{\gamma,\kappa}}$; $\psi_\gamma := (c_{\gamma,1}, \cdots, c_{\gamma,\kappa}, \pi_\gamma)$
$(K_{b_1}, \cdots, K_{b_s}) = F(\psi_1, \cdots, \psi_\ell)$; $K_{\ell+1} := (K_{a_1}, K_{b_1})$; $\cdots$; $K_{\ell+s} := (K_{a_s}, K_{b_s})$
$T = \mathsf{XAuth}(K_1, \cdots, K_{\ell+s})$
Return $c = (\psi_1, \cdots, \psi_\ell, T)$

$\underline{\mathsf{Dec}(sk, c = (\psi_1, \cdots, \psi_\ell, T)):}$
For $\gamma = 1$ to ℓ : $\overline{m}_\gamma := 0$
$(\overline{K}_{b_1}, \cdots, \overline{K}_{b_s}) = F(\psi_1, \cdots, \psi_\ell)$; $\overline{K}_{\ell+1} := (K_{a_1}, \overline{K}_{b_1})$; $\cdots$; $\overline{K}_{\ell+s} := (K_{a_s}, \overline{K}_{b_s})$
If $\bigwedge_{\eta=1}^{s} (\mathsf{XVer}(\overline{K}_{\ell+\eta}, T) = 1)$:
For $\gamma = 1$ to ℓ :
Parse $\psi_\gamma = (c_{\gamma,1}, \cdots, c_{\gamma,\kappa}, \pi_\gamma)$
$t_\gamma := \mathsf{H}_{\mathrm{TCR}}(c_{\gamma,1}, \cdots, c_{\gamma,\kappa})$
If $\prod_{\theta=1}^{\kappa} (c_{\gamma,\theta})^{\alpha_\theta t_\gamma + \beta_\theta} \neq \pi_\gamma$: $\overline{m}_\gamma = 0$
Else: $\overline{K}_\gamma = \prod_{\theta=1}^{\kappa} (c_{\gamma,\theta})^{w_\theta}$; $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T)$
Return $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$

Fig. 8 Construction of $\mathsf{PKE}_{\mathrm{LP}}$.

$(\alpha_1, \ldots, \alpha_{2k+1}) = \Gamma(x)$, then

$$
\begin{aligned}
\Lambda_{hsk}(x) &= \prod_{h \in [2k+1], i \in [\kappa], j \in [2k+1]} u_j^{s_{h,i,j} \cdot \alpha_h^{i-1}} \\
&= \prod_{h \in [2k+1], i \in [\kappa], j \in [2k+1]} g_j^{w \cdot s_{h,i,j} \cdot \alpha_h^{i-1}} \\
&= \prod_{h \in [2k+1], i \in [\kappa]} \Big(\prod_{j \in [2k+1]} g_j^{s_{h,i,j}} \Big)^{w \cdot \alpha_h^{i-1}} \\
&= \prod_{h \in [2k+1], i \in [\kappa]} (p_{h,i})^{w \cdot \alpha_h^{i-1}}
\end{aligned}
$$

which is determined by hpk and x, and is exactly the output of PubEv on input hpk, x and w.

Perfect Universal$_\kappa$. For any $hpk = (p_{h,i})_{h\in[2k+1],i\in[\kappa]} \in \mathcal{PK}$, any distinct $(x_\ell = (u_{\ell,1}, \ldots, u_{\ell,2k+1}))_{\ell\in[\kappa]} \in (\mathcal{X}\backslash\mathcal{L})^\kappa$, and any $(K_\ell)_{\ell\in[\kappa]} \in \mathcal{K}_{sp}^\kappa$, we show that

$$\Pr\left[\Lambda_{hsk}(x_\kappa) = K_\kappa : \begin{array}{c}\mu(hsk) = hpk,\\ \Lambda_{hsk}(x_1) = K_1, \cdots, \Lambda_{hsk}(x_{\kappa-1}) = K_{\kappa-1}\end{array}\right] = \frac{1}{|\mathcal{K}_{sp}|}\ .$$

First, as $x_\kappa \in \mathcal{X}\backslash\mathcal{L}$, there exists $j_1, j_2 \in [2k+1]$ that $u_{\kappa,j_1}^{a_{j_1}^{-1}} \neq u_{\kappa,j_2}^{a_{j_2}^{-1}}$. Without loss of generality, we assume $j_1 = 1$ and $j_2 = 2$. Next, we will prove the following stronger argument

$$\Pr\left[\Lambda_{hsk}(x_\kappa) = K_\kappa : \begin{array}{c}\mu(hsk) = hpk,\\ \Lambda_{hsk}(x_1) = K_1, \cdots, \Lambda_{hsk}(x_{\kappa-1}) = K_{\kappa-1}\\ hsk = (*, *, s_{h,i,3}, \ldots, s_{h,i,2k+1})_{h\in[2k+1],i\in[\kappa]}\end{array}\right] = \frac{1}{|\mathcal{K}_{sp}|}\ . \tag{3}$$

Let $p_{h,i} = g^{b'_{h,i}}$ and let $b_{h,i} = b'_{h,i} - (\sum_{j\in[3,2k+1]} a_j s_{h,i,j})$ for $h \in [2k+1]$, $i \in [\kappa]$. Also, for $\ell \in [\kappa]$:

- Let $u_{\ell,1} = g^{w_{\ell,1}}$ and $u_{\ell,2} = g^{w_{\ell,2}}$. Note that $a_1^{-1} w_{\kappa,1} \neq a_2^{-1} w_{\kappa,2}$.
- Let $\boldsymbol{\alpha}_\ell = (\alpha_{\ell,1}, \ldots, \alpha_{\ell,2k+1}) = \Gamma(x_\ell)$. Here, $\boldsymbol{\alpha}_{\ell_1} \neq \boldsymbol{\alpha}_{\ell_2}$ for any $\ell_1 \neq \ell_2$.
- Let $K'_\ell = K_\ell / (\prod_{h\in[2k+1],i\in[\kappa],j\in[3,2k+1]} u_{\ell,j}^{s_{h,i,j}\alpha_{\ell,h}^{i-1}})$ and define $K'_\ell = g^{z_\ell}$.

Then, we can transform Equation (3) into

$$\Pr\left[\begin{array}{l} w_{\kappa,1} \cdot \sum_{h\in[2k+1],i\in[\kappa]}(s_{h,i,1} \cdot \alpha_{\kappa,h}^{i-1})\\ \quad + w_{\kappa,2} \cdot \sum_{h\in[2k+1],i\in[\kappa]}(s_{h,i,2} \cdot \alpha_{\kappa,h}^{i-1}) = z_\kappa :\\ \\ \forall h \in [2k+1], i \in [\kappa], b_{h,i} = a_1 s_{h,i,1} + a_2 s_{h,i,2}\\ \forall \ell \in [\kappa-1], z_\ell = w_{\ell,1} \cdot \sum_{h\in[2k+1],i\in[\kappa]}(s_{h,i,1} \cdot \alpha_{\ell,h}^{i-1})\\ \qquad\qquad + w_{\ell,2} \cdot \sum_{h\in[2k+1],i\in[\kappa]}(s_{h,i,2} \cdot \alpha_{\ell,h}^{i-1})\end{array}\right] = \frac{1}{|\mathcal{K}_{sp}|} \tag{4}$$

where the probability is taken over the random choice of $(s_{h,i,1}, s_{h,i,2})_{h\in[2k+1],i\in[\kappa]}$.

Let $z'_\ell = z_\ell - w_{\ell,1} a_1^{-1} \cdot \sum_{h\in[2k+1],i\in[\kappa]}(b_{h,i} \cdot \alpha_{\ell,h}^{i-1})$ and $\Delta_\ell = w_{\ell,2} - w_{\ell,1} a_1^{-1} a_2$, note that $\Delta_\kappa \neq 0$. Then for all $\ell \in [\kappa]$ that $\Delta_\ell \neq 0$, let $z^*_\ell = z'_\ell / \Delta_\ell$. Then, we can transform Equation (3) into

$$\Pr\left[\begin{array}{l}\sum_{h\in[2k+1],i\in[\kappa]}(s_{h,i,2} \cdot \alpha_{\kappa,h}^{i-1}) = z^*_\kappa :\\ \\ \forall \ell \in [\kappa-1] \text{ s.t. } \Delta_\ell \neq 0, \sum_{h\in[2k+1],i\in[\kappa]}(s_{h,i,2} \cdot \alpha_{\ell,h}^{i-1}) = z^*_\ell\end{array}\right] = \frac{1}{|\mathcal{K}_{sp}|} \tag{5}$$

where the probability is taken over the random choice of $(s_{h,i,2})_{h\in[2k+1],i\in[\kappa]}$.

Define $\boldsymbol{v}_{\ell,h} = (1, \alpha_{\ell,h}, \ldots, \alpha_{\ell,h}^{\kappa-1})$ and define $\boldsymbol{v}_\ell = (\boldsymbol{v}_{\ell,1} \| \ldots \| \boldsymbol{v}_{\ell,2k+1})$. Since $\boldsymbol{\alpha}_\kappa \neq \boldsymbol{\alpha}_\ell$ for $\ell \in [\kappa-1]$, we have $\boldsymbol{v}_\kappa$ linear independent of $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\kappa-1})$, and Equation (5) follows. This completes the proof of perfect universal$_\kappa$.

Key Equivocability. Let $hsk = (s_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]}$ be a random secret key in $\mathcal{SK}$. Let $hpk = (p_{h,i})_{h\in[2k+1],i\in[\kappa]} = \mu(hsk)$. Let $(x_\ell = (u_{\ell,1}, \ldots, u_{\ell,2k+1}))_{\ell\in[2k]}$ be elements from different $\mathcal{L}_i$, and w.l.o.g., we assume that $x_\ell \in \mathcal{L}_\ell$ for $\ell \in [2k]$. Then we can write

$$x_\ell = (g_1^{w_\ell}, \ldots, g_{\ell-1}^{w_\ell}, g_\ell^{w'_\ell}, g_{\ell+1}^{w_\ell}, \ldots, g_{2k+1}^{w_\ell}),$$

where w_ℓ and w'_ℓ are distinct integers in $\mathbb{Z}_q$, and define $\Delta_\ell = w'_\ell - w_\ell$. Let $(\alpha_{\ell,1}, \ldots, \alpha_{\ell,2k+1}) = \Gamma(x_\ell)$ for $\ell \in [2k]$. Let $hsk' \leftarrow \mathsf{SampHsk}(hsk, \mathsf{td}, x_1, \ldots, x_{2k})$ and write it as $hsk' = (s'_{h,i,j})_{h\in[2k+1],i\in[\kappa],j\in[2k+1]}$. Then for $h \in [2k+1], i \in [\kappa]$, we have

- $s'_{h,i,j} = s_{h,i,j}$ for $j \in [k]$;
- $s'_{h,i,j} \leftarrow \mathbb{Z}_q$ for $j \in [k+1, 2k]$;
- $s'_{h,i,2k+1} = (\sum_{j\in[2k+1]} a_j s_{h,i,j} - \sum_{j\in[2k]} a_j s'_{h,i,j}) \cdot a_{2k+1}^{-1}$.

First, note that

$$\begin{aligned}
\mu(hsk') &= (\prod_{j\in[2k+1]} g_j^{s'_{h,i,j}})_{h\in[2k+1],i\in[\kappa]} \\
&= (g^{\sum_{j\in[2k+1]} a_j s'_{h,i,j}})_{h\in[2k+1],i\in[\kappa]} \\
&= (g^{\sum_{j\in[2k+1]} a_j s_{h,i,j}})_{h\in[2k+1],i\in[\kappa]} \\
&= hpk.
\end{aligned}$$

Also, for $\ell \in [k]$,

$$\begin{aligned}
K_\ell &= \prod_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} u_{\ell,j}^{s_{h,i,j}\cdot\alpha_{\ell,h}^{i-1}} \\
&= \prod_{h\in[2k+1],i\in[\kappa]} (g_\ell^{\Delta_\ell \cdot s_{h,i,\ell}\cdot\alpha_{\ell,h}^{i-1}} \cdot \prod_{j\in[2k+1]} g_j^{w_\ell \cdot s_{h,i,j}\cdot\alpha_{\ell,h}^{i-1}}) \\
&= \prod_{h\in[2k+1],i\in[\kappa]} (g_\ell^{\Delta_\ell \cdot s_{h,i,\ell}\cdot\alpha_{\ell,h}^{i-1}} \cdot p_{h,i}^{w_\ell\cdot\alpha_{\ell,h}^{i-1}}) \\
&= \prod_{h\in[2k+1],i\in[\kappa]} (g_\ell^{\Delta_\ell \cdot s'_{h,i,\ell}\cdot\alpha_{\ell,h}^{i-1}} \cdot p_{h,i}^{w_\ell\cdot\alpha_{\ell,h}^{i-1}})
\end{aligned}$$

$$
\begin{aligned}
&= \prod_{h\in[2k+1],i\in[\kappa]} (g_\ell^{\Delta_\ell \cdot s'_{h,i,\ell}\cdot\alpha_{\ell,h}^{i-1}} \cdot \prod_{j\in[2k+1]} g_j^{w_\ell \cdot s'_{h,i,j}\cdot\alpha_{\ell,h}^{i-1}}) \\
&= \prod_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} u_{\ell,j}^{s'_{h,i,j}\cdot\alpha_{\ell,h}^{i-1}} \\
&= \Lambda_{hsk'}(x_\ell).
\end{aligned}
$$

Thus, $(hsk, hpk, K_1, \ldots, K_k)$ and $(hsk', hpk, K_1, \ldots, K_k)$ are identically distributed.

Next, we will show that $(K_{k+1}, \ldots, K_{2k})$ is uniform in $\mathcal{K}_{sp}^k$ given $(hsk', hpk, K_1, \ldots, K_k)$. For all $\ell \in [k+1, 2k]$,

$$
\begin{aligned}
K_\ell &= \prod_{h\in[2k+1],i\in[\kappa],j\in[2k+1]} u_{\ell,j}^{s_{h,i,j}\cdot\alpha_{\ell,h}^{i-1}} \\
&= \prod_{h\in[2k+1],i\in[\kappa]} (g_\ell^{\Delta_\ell \cdot s_{h,i,\ell}\cdot\alpha_{\ell,h}^{i-1}} \cdot \prod_{j\in[2k+1]} g_j^{w_\ell \cdot s_{h,i,j}\cdot\alpha_{\ell,h}^{i-1}}) \\
&= \prod_{h\in[2k+1],i\in[\kappa]} (g_\ell^{\Delta_\ell \cdot s_{h,i,\ell}\cdot\alpha_{\ell,h}^{i-1}} \cdot p_{h,i}^{w_\ell\cdot\alpha_{\ell,h}^{i-1}}),
\end{aligned}
$$

which are uniformly distributed in $\mathcal{K}_{sp}$ since each $(s_{h,i,k+1}, \ldots, s_{h,i,2k})$ is uniform in $\mathbb{Z}_q^k$ and is independent of $(hsk', hpk, K_1, \ldots, K_k)$. This completes the proof of key equivocability. □

E Deterred Proof of Theorem 5

Proof (of Theorem 5). For any PPT adversary $\mathcal{A}$, in the real experiment $\mathsf{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)$, we denote the challenge ciphertexts and their corresponding messages by $(\mathbf{c}_i^*)_{i\in[n]}$ and $(\mathbf{m}_i^*)_{i\in[n]}$, respectively. More specifically, for each $i \in [n]$ and $j \in [|\mathbf{m}_i^*|]$, we write $\mathbf{m}_i^*[j] = (m^*_{i,j,1}, \cdots, m^*_{i,j,\ell}) \in \{0,1\}^\ell$, $\mathbf{r}_i^*[j] = (r^{*(\mathcal{X})}_{i,j,\gamma}, r^{*(\mathcal{K})}_{i,j,\gamma}, w^*_{i,j,\gamma})_{\gamma\in[\ell]}$, and $\mathbf{c}_i^*[j] = (x^*_{i,j,1}, \cdots, x^*_{i,j,\ell}, T^*_{i,j})$. For each $i \in [n]$, $j \in [|\mathbf{m}_i^*|]$ and $\gamma \in [\ell]$, we write $K^*_{i,j,\gamma}$, $K^*_{b,i,j} = \mathsf{H}(pk_i, x^*_{i,j,1}, \cdots, x^*_{i,j,\ell})$ and $K^*_{i,j,\ell+1} = (K_a, K^*_{b,i,j})$ similarly. For each $i \in [n]$, we write $pk_i = (hpk_{i,\gamma})_{\gamma\in[\ell]}$, and $sk_i = (hsk_{i,\gamma})_{\gamma\in[\ell]}$. In $\mathsf{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)$, we additionally define a finite set $\mathcal{I}_{\mathsf{op}}$ as follows. At the beginning of this game, let $\mathcal{I}_{\mathsf{op}} := \emptyset$, and when $\mathcal{A}$ submits its selective opening query $\mathcal{I}$, update that $\mathcal{I}_{\mathsf{op}} = \mathcal{I}$.

Without loss of generality, we assume that $\mathcal{A}$ always makes q_d decryption queries and n `MkRec` queries, where q_d and n are both some polynomial functions.

We proceed in a series of games.

Game $\mathbf{G}_0$: $\mathbf{G}_0$ is the real experiment $\mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda)$, i.e.,

$$\mathbf{G}_0 = \mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda). \tag{6}$$

Game $\mathbf{G}_1$: $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except that we abort this game (with output $\perp$) as soon as there exists some $(i,\gamma) \neq (i',\gamma')$ such that $hsk_{i,\gamma} = hsk_{i',\gamma'}$. By a union bound, we obtain that for any PPT distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(\mathbf{G}_1)=1] - \Pr[\mathcal{D}(\mathbf{G}_0)=1]| \leq \frac{n\ell(n\ell-1)}{2|\mathcal{SK}|}. \tag{7}$$

Game $\mathbf{G}_2$: $\mathbf{G}_2$ is the same as $\mathbf{G}_1$, except for the decryption oracle. Specifically, if $\mathcal{A}$ submits a decryption query $(i, c=(x_1,\cdots,x_\ell,T))$, such that there exists some $j \in [|\mathbf{m}_i|]$ satisfying that $(x^*_{i,j,1},\cdots,x^*_{i,j,\ell}) \neq (x_1, \cdots, x_\ell)$ and $\mathsf{H}(pk_i, x^*_{i,j,1},\cdots,x^*_{i,j,\ell}) = \mathsf{H}(pk_i, x_1,\cdots,x_\ell)$, then we abort this game (with output $\perp$). Since H is collision-resistant, we derive that for any PPT distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(\mathbf{G}_2)=1] - \Pr[\mathcal{D}(\mathbf{G}_1)=1]| \leq \mathtt{Adv}^{\text{CR}}_{\mathcal{H}_{\mathsf{prmins}},\mathcal{A}_{\mathsf{H}}}(\lambda) \tag{8}$$

for a suitable PPT adversary $\mathcal{A}_{\mathsf{H}}$.

From now on, for each game we consider two cases: $\beta=0$ and $\beta=1$.

We firstly look at the case of $\beta=0$.

Game $\mathbf{G}_{3|\beta=0}$: $\mathbf{G}_{3|\beta=0}$ is the same as $\mathbf{G}_2$ when $\beta=0$, except that we abort this game (with output $\perp$) as soon as there exist some $(i',j',\gamma') \neq (i'',j'',\gamma'')$ such that $x^*_{i',j',\gamma'} = x^*_{i'',j'',\gamma''}$. By a union bound, we obtain that for any PPT distinguisher $\mathcal{D}$,

$$\left|\Pr[\mathcal{D}(\mathbf{G}_{3|\beta=0})=1] - \Pr[\mathcal{D}(\mathbf{G}_2)=1 \mid \beta=0]\right| \leq \frac{n\widetilde{k}\ell(n\widetilde{k}\ell-1)}{2|\mathcal{L}|}, \tag{9}$$

where $\widetilde{k} := \max_{i=1}^{n} |\mathbf{m}^*_i|$.

Game $\mathbf{G}^s_{(1,1,0)}$: For convenience, we write that

$$\mathbf{G}^s_{(1,1,0)} := \mathbf{G}_{3|\beta=0}. \tag{10}$$

Now, for any $i \in [n]$ and $j \in [|\mathbf{m}^*_i|]$, let

$$\mathbf{G}^s_{(i,j+1,0)} := \mathbf{G}^s_{(i,j,\ell)}, \tag{11}$$

$$\mathbf{G}^s_{(i+1,1,0)} := \mathbf{G}^s_{(i,|\mathbf{m}^*_i|,\ell)}, \tag{12}$$

and for any $\gamma \in [\ell]$, we consider game $\mathbf{G}^s_{(i,j,\gamma)}$ as follows.

Game $\mathbf{G}^s_{(i,j,\gamma)}$ ($i \in [n], j \in [|\mathbf{m}^*_i|], \gamma \in [\ell]$): $\mathbf{G}^s_{(i,j,\gamma)}$ is the same as $\mathbf{G}^s_{(i,j,\gamma-1)}$, except for the generation and the related selective opening procedure of $(x^*_{i,j,\gamma}, K^*_{i,j,\gamma})$, which corresponds to $\mathbf{c}^*_i[j] = (x^*_{i,j,1}, \cdots, x^*_{i,j,\ell}, T^*_{i,j})$. Specifically, in $\mathbf{G}^s_{(i,j,\gamma)}$,

- *During the generation of* $(x^*_{i,j,\gamma}, K^*_{i,j,\gamma})$*:* The challenger samples $w^*_{i,j,\gamma} \leftarrow \mathcal{R}_{\mathsf{SSmpL}}$, and computes $x^*_{i,j,\gamma} \leftarrow \mathsf{SSmpL}(\mathsf{prm}; w^*_{i,j,\gamma})$ and $K^*_{i,j,\gamma} = \mathsf{PubEv}(hpk_{i,\gamma}, x^*_{i,j,\gamma}, w^*_{i,j,\gamma})$, no matter whether $m^*_{i,j,\gamma}$ is 0 or 1.
- *To answer the selective opening query* $\mathcal{I}$*:* If $(i,j) \notin \mathcal{I}$, the selective opening procedure in $\mathbf{G}^s_{(i,j,\gamma)}$ is the same as that in $\mathbf{G}^s_{(i,j,\gamma-1)}$. So we only need to consider the case that $(i,j) \in \mathcal{I}$. We assume that $m^*_{i,j,\gamma} = 0$ (otherwise $\mathbf{G}^s_{(i,j,\gamma)}$ and $\mathbf{G}^s_{(i,j,\gamma-1)}$ are identical). For any $(i',j') \in \mathcal{I}$ satisfying that $(i',j') \neq (i,j)$, the challenger extracts and generates the corresponding randomness as that in $\mathbf{G}^s_{(i,j,\gamma-1)}$. Then, the challenger samples $\widehat{w}^*_{i,j,\gamma} \leftarrow \mathcal{R}_{\mathsf{SSmpL}}$, computes $\widehat{r}^{*(\mathcal{X})}_{i,j,\gamma} \leftarrow \mathsf{Explain}(\mathcal{X}, x^*_{i,j,\gamma})$, updates $K^*_{i,j,\gamma} \leftarrow \mathsf{ReSamp}(\gamma, K^*_{i,j,\neq\gamma}, T^*_{i,j})$, and computes $\widehat{r}^{*(\mathcal{K})}_{i,j,\gamma} \leftarrow \mathsf{Explain}(\mathcal{XK}, K^*_{i,j,\gamma})$. $(\widehat{r}^{*(\mathcal{X})}_{i,j,\gamma}, \widehat{r}^{*(\mathcal{K})}_{i,j,\gamma}, \widehat{w}^*_{i,j,\gamma})$ is the randomness (for (i,j)) returned to the adversary.

Now, we present the following lemma with a postponed proof.

Lemma 1. *For any $i \in [n]$, any $j \in [|\boldsymbol{m}^*_i|]$, any $\gamma \in [\ell]$, and any PPT distinguisher $\mathcal{D}$,*

$$\left|\Pr[\mathcal{D}(\mathbf{G}^s_{(i,j,\gamma)}) = 1 \mid \beta = 0] - \Pr[\mathcal{D}(\mathbf{G}^s_{(i,j,\gamma-1)}) = 1 \mid \beta = 0]\right| \Pr[\beta = 0] \leq \mathsf{negl}(\lambda).$$

With the above lemma and Eq. (10)-(12), we obtain that

$$\begin{aligned}\mathbf{G}_{3|\beta=0} &= \mathbf{G}^s_{(1,1,0)} \approx \cdots \approx \mathbf{G}^s_{(1,1,\ell)} \\ &= \mathbf{G}^s_{(1,2,0)} \approx \cdots \approx \mathbf{G}^s_{(1,|\mathbf{m}^*_1|,\ell)} = \mathbf{G}^s_{(2,1,0)} \approx \cdots \approx \mathbf{G}^s_{(n,|\mathbf{m}^*_n|,\ell)}. \quad (13)\end{aligned}$$

Game $\mathbf{G}_{4|\beta=0}$: Let $\mathbf{G}_{4|\beta=0} := \mathbf{G}^s_{(n,|\mathbf{m}^*_n|,\ell)}$. So we have that for any PPT distinguisher $\mathcal{D}$,

$$\left|\Pr[\mathcal{D}(\mathbf{G}_{4|\beta=0}) = 1] - \Pr[\mathcal{D}(\mathbf{G}_{3|\beta=0}) = 1]\right| \Pr[\beta = 0] \leq \mathsf{negl}(\lambda). \quad (14)$$

Next, we turn to the case of $\beta = 1$.

Game $\mathbf{G}_{3|\beta=1}$: $\mathbf{G}_{3|\beta=1}$ is the same as $\mathbf{G}_2$ when $\beta = 1$, except that $(\mathbf{c}^*_i)_{i \in \{i' \in [n] || \mathbf{m}^*_{i'}| \leq k\}}$ are generated with $(sk_i)_{i \in \{i' \in [n] || \mathbf{m}^*_{i'}| \leq k\}}$, instead of $(pk_i)_{i \in \{i' \in [n] || \mathbf{m}^*_{i'}| \leq k\}}$. Specifically, during the generation of $(\mathbf{c}^*_i)_{i \in \{i' \in [n] || \mathbf{m}^*_{i'}| \leq k\}}$ in $\mathbf{G}_{3|\beta=1}$, for all $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$, all $j \in [|\mathbf{m}^*_i|]$ and all $\gamma \in [\ell]$, if

$m^*_{i,j,\gamma} = 1$, then the challenger computes $K^*_{i,j,\gamma} = \mathsf{SecEv}(hsk_{i,\gamma}, x^*_{i,j,\gamma})$, instead of $\mathsf{PubEv}(hpk_{i,\gamma}, x^*_{i,j,\gamma}, w^*_{i,j,\gamma})$. The projective property of HPS guarantees that the view of $\mathcal{A}$ in $\mathbf{G}_{3|\beta=1}$ are identical to that in $\mathbf{G}_2$ when $\beta = 1$. So we derive that

$$\mathbf{G}_{3|\beta=1} = \mathbf{G}_2 \text{ (when } \beta = 1). \tag{15}$$

Note that the modification introduced in $\mathbf{G}_1$ ensures that during the generation of $(\mathbf{c}^*_i)_{i\in[n]}$ in $\mathbf{G}_{3|\beta=1}$, for all $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$ and all $\gamma \in [\ell]$, $hsk_{i,\gamma}$ is employed at most k times, and for any $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| > k\}$, $hsk_{i,\gamma}$ has never been used.

Game $\mathbf{G}_{3.1|\beta=1}$: $\mathbf{G}_{3.1|\beta=1}$ is the same as $\mathbf{G}_{3|\beta=1}$, except for the sampling process of $(x^*_{i,j,\gamma})_{i\in\{i'\in[n]||\mathbf{m}^*_{i'}|\leq k\},j\in[|\mathbf{m}^*_i|],\gamma\in[\ell]}$ during the generation of $(\mathbf{c}^*_i)_{i\in\{i'\in[n]||\mathbf{m}^*_{i'}|\leq k\}}$. In $\mathbf{G}_{3.1|\beta=1}$, for all $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$ and $j \in [|\mathbf{m}^*_i|]$, $x^*_{i,j,1}, \cdots, x^*_{i,j,\ell}$ are all sampled from $\mathcal{L}_j$, no matter whether $m^*_{i,j,\gamma}$ (where $\gamma \in [\ell]$) is 1 or 0. Specifically, during the generation of $(\mathbf{c}^*_i)_{i\in\{i'\in[n]||\mathbf{m}^*_{i'}|\leq k\}}$ in $\mathbf{G}_{3.1|\beta=1}$, $(x^*_{i,j,\gamma})_{i\in\{i'\in[n]||\mathbf{m}^*_{i'}|\leq k\},j\in[|\mathbf{m}^*_i|],\gamma\in[\ell]}$ are sampled as shown in Fig. 9.

For $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$,
  For $j = 1$ to $|\mathbf{m}^*_i|$,
    For $\gamma = 1$ to ℓ,
      $x^*_{i,j,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j)$

Fig. 9 The sampling process of $(x^*_{i,j,\gamma})_{i\in\{i'\in[n]||\mathbf{m}^*_{i'}|\leq k\},j\in[|\mathbf{m}^*_i|],\gamma\in[\ell]}$ in $\mathbf{G}_{3.1|\beta=1}$.

A simple hybrid argument shows that for any PPT distinguisher $\mathcal{D}$,

$$\begin{aligned}&\left|\Pr[\mathcal{D}(\mathbf{G}_{3.1|\beta=1}) = 1] - \Pr[\mathcal{D}(\mathbf{G}_{3|\beta=1}) = 1]\right| \Pr[\beta = 1] \\ &\leq nk\ell(\mathtt{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D}}(\lambda) + \mathtt{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D}}(\lambda)).\end{aligned} \tag{16}$$

Game $\mathbf{G}_{3.2|\beta=1}$: $\mathbf{G}_{3.2|\beta=1}$ is the same as $\mathbf{G}_{3.1|\beta=1}$, except for the decryption oracle. In $\mathbf{G}_{3.2|\beta=1}$, for any decryption query $(i, c = (x_1, \cdots, x_\ell, T))$ satisfying $((i, c) \notin \mathcal{C}) \bigwedge (i \notin \mathcal{I}_{\mathsf{op}}) \bigwedge (i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\})$, where $\mathcal{C}$ denotes the set of the challenge ciphertexts and their corresponding public keys (in Fig. 2), the challenger firstly checks whether $\mathsf{XVer}((K_a, \mathsf{H}(pk_i, x_1, \cdots, x_\ell)), T) = 0$. If so, it sets that $\overline{m}_1 = \cdots = \overline{m}_\ell = 0$ and returns $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$. Otherwise, for each $\gamma \in [\ell]$, the challenger sets $\overline{m}_\gamma = 0$ directly if $x_\gamma \notin \mathcal{L}$, and behaves just as in $\mathbf{G}_{3.1|\beta=1}$ if $x_\gamma \in \mathcal{L}$ (i.e., it computes $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x_\gamma)$ and sets $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T)$).

We present the following lemma with a postponed proof.

Lemma 2. *For any (even unbounded) distinguisher $\mathcal{D}_s$,*

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}_{3.2|\beta=1}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}_{3.1|\beta=1}) = 1]\right| \leq q_d \ell \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\}. \tag{17}$$

Note that the decryption oracle in $\mathbf{G}_{3.2|\beta=1}$ is inefficient, and it doesn't leak any information on $hsk_{i,\gamma}$ beyond $hpk_{i,\gamma}$ for any $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$ and any $\gamma \in [\ell]$.

Game $\mathbf{G}^r_{(1,0,0)}$: For convenience, we write that

$$\mathbf{G}^r_{(1,0,0)} := \mathbf{G}_{3.2|\beta=1}. \tag{18}$$

Note that in game $\mathbf{G}^r_{(1,0,0)}$, for any $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$, $sk_i = (hsk_{i,\gamma})_{\gamma\in[\ell]}$ is used to encrypt

$$\begin{aligned}
\mathbf{m}^*_i[1] &= (m^*_{i,1,1}, \cdots, m^*_{i,1,\ell}),\\
&\vdots\\
\mathbf{m}^*_i[|\mathbf{m}^*_i|] &= (m^*_{i,|\mathbf{m}^*_i|,1}, \cdots, m^*_{i,|\mathbf{m}^*_i|,\ell}).
\end{aligned}$$

More specifically, for any $\gamma \in [\ell]$, $hsk_{i,\gamma}$ is employed to handle

$$m^*_{i,1,\gamma}, m^*_{i,2,\gamma}, \cdots, m^*_{i,|\mathbf{m}^*_i|,\gamma}.$$

For all $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| > k\}$, let

$$\mathbf{G}^r_{(1,i-1,0)} = \mathbf{G}^r_{(1,i-1,1)} = \cdots = \mathbf{G}^r_{(1,i-1,\ell)} = \mathbf{G}^r_{(1,i,0)}, \tag{19}$$

and for all $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$, let

$$\mathbf{G}^r_{(1,i,0)} := \mathbf{G}^r_{(1,i-1,\ell)}. \tag{20}$$

Now for any $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$ and any $\gamma \in [\ell]$, we consider game $\mathbf{G}^r_{(1,i-1,\gamma)}$ as follows.

Game $\mathbf{G}^r_{(1,i-1,\gamma)}$ ($i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}, \gamma \in [\ell]$): Note that for each $j \in [|\mathbf{m}^*_i|]$, we write $\mathbf{c}^*_i[j] = (x^*_{i,j,1}, \cdots, x^*_{i,j,\ell}, T^*_{i,j})$, where $T^*_{i,j} = \mathsf{XAuth}(K^*_{i,j,1}, \cdots, K^*_{i,j,\ell+1})$. Game $\mathbf{G}^r_{(1,i-1,\gamma)}$ is the same as $\mathbf{G}^r_{(1,i-1,\gamma-1)}$, except for the generation and the corresponding selective opening procedure of $(K^*_{i,1,\gamma}, \cdots, K^*_{i,|\mathbf{m}^*_i|,\gamma})$. Specifically, in $\mathbf{G}^r_{(1,i-1,\gamma)}$,

- *During the generation of $(K^*_{i,1,\gamma}, \cdots, K^*_{i,|\mathbf{m}^*_i|,\gamma})$:* For all $j \in [|\mathbf{m}^*_i|]$, if $m^*_{i,j,\gamma} = 0$, the challenger computes $K^*_{i,j,\gamma} = \mathsf{SecEv}(hsk_{i,\gamma}, x^*_{i,j,\gamma})$, instead of uniformly sampling $K^*_{i,j,\gamma}$ from $\mathcal{K}_{sp}$.

- *To answer the selective opening query $\mathcal{I}$:* If $i \notin \mathcal{I}$, the selective opening procedure in $\mathbf{G}^r_{(1,i-1,\gamma)}$ is the same as that in $\mathbf{G}^r_{(1,i-1,\gamma-1)}$. So we only need to consider the case that $i \in \mathcal{I}$. The challenger firstly generates $(y_{i,1,\gamma}, \cdots, y_{i,2k,\gamma})$ as follows: for each $j \in [|\mathbf{m}^*_i|]$,
  - if $m^*_{i,j,\gamma} = 1$, then it sets $y_{i,j,\gamma} = x^*_{i,j,\gamma}$, and $y_{i,j+k,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j+k)$;
  - if $m^*_{i,j,\gamma} = 0$, then it sets $y_{i,j+k,\gamma} = x^*_{i,j,\gamma}$, and $y_{i,j,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j+k)$;

  and for each $j \in \{|\mathbf{m}^*_i|+1, \cdots, k\}$ (if $|\mathbf{m}^*_i| \leq k$),
  - it sets that $y_{i,j,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j)$ and $y_{i,j+k,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j+k)$.

  Then, the challenger computes

$$hsk'_{i,\gamma} \leftarrow \mathsf{SampHsk}(hsk_{i,\gamma}, td, y_{i,1,\gamma}, \cdots, y_{i,2k,\gamma}),$$

  and updates $hsk_{i,\gamma} = hsk'_{i,\gamma}$ (which means that *from now on*, the challenger will use the updated $hsk_{i,\gamma}$ to answer the selective opening query *and the decryption queries*).

Key equivocability of HPS guarantees that for any (even unbounded) distinguisher $\mathcal{D}_s$,

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}^r_{(1,i-1,\gamma)}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}^r_{(1,i-1,\gamma-1)}) = 1]\right| \leq \mathsf{negl}(\lambda). \quad (21)$$

Therefore, we obtain that

$$\begin{aligned}\mathbf{G}_{3.2|\beta=1} &= \mathbf{G}^r_{(1,0,0)} \stackrel{s}{\approx} \cdots \stackrel{s}{\approx} \mathbf{G}^r_{(1,0,\ell)} = \mathbf{G}^r_{(1,1,0)} \stackrel{s}{\approx} \cdots \stackrel{s}{\approx} \mathbf{G}^r_{(1,1,\ell)} \\ &= \mathbf{G}^r_{(1,2,0)} \stackrel{s}{\approx} \cdots \stackrel{s}{\approx} \mathbf{G}^r_{(1,k,\ell)}. \end{aligned} \quad (22)$$

Game $\mathbf{G}_{3.3|\beta=1}$: Let $\mathbf{G}_{3.3|\beta=1} = \mathbf{G}^r_{(1,k,\ell)}$. Combining Eq. (20)-(22), we derive that for any (even unbounded) $\mathcal{D}_s$,

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}_{3.3|\beta=1}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}_{3.2|\beta=1}) = 1]\right| \leq \mathsf{negl}(\lambda). \quad (23)$$

We emphasize that in $\mathbf{G}_{3.3|\beta=1}$, for all $i \in \{i' \in [n] \mid |\mathbf{m}^*_{i'}| \leq k\}$, all $j \in [|\mathbf{m}^*_i|]$ and all $\gamma \in [\ell]$, $K^*_{i,j,\gamma}$ is computed with the original $hsk_{i,\gamma}$ when $m^*_{i,j,\gamma} = 0$, and if $i \in \mathcal{I}$, $sk_i = (hsk_{i,1}, \cdots, hsk_{i,\ell})$ will be updated.

Game $\mathbf{G}_{4|\beta=1}$: $\mathbf{G}_{4|\beta=1}$ is the same as $\mathbf{G}_{3.3|\beta=1}$, except that the decryption oracle works with the original decryption rule. In other words, in $\mathbf{G}_{4|\beta=1}$, for any decryption query $(i, c = (x_1, \cdots, x_\ell, T))$, the challenger firstly checks whether $\mathsf{XVer}((K_a, \mathsf{H}(pk_i, x_1, \cdots, x_\ell)), \ell+1, T) = 0$. If so, it sets that $\overline{m}_1 = \cdots = \overline{m}_\ell = 0$ and returns $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$. Otherwise,

for each $\gamma \in [\ell]$, the challenger computes $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x_\gamma)$, sets $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, \gamma, T)$, and finally returns $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$.

We present the following lemma, the proof of which is the same as that of Lemma 2.

Lemma 3. *For any (even unbounded) distinguisher $\mathcal{D}_s$,*

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}_{4|\beta=1}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}_{3.3|\beta=1}) = 1]\right| \leq q_d \ell \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\}. \tag{24}$$

Notice that the decryption oracle in game $\mathbf{G}_{4|\beta=1}$ is efficient again.

Combining Eq. (16), (17), (23) and (24), we derive that for any PPT distinguisher $\mathcal{D}$,

$$\begin{aligned}
&\left|\Pr[\mathcal{D}(\mathbf{G}_{4|\beta=1}) = 1] - \Pr[\mathcal{D}(\mathbf{G}_{3|\beta=1}) = 1]\right| \Pr[\beta = 1] \\
\leq\ &\left|\Pr[\mathcal{D}(\mathbf{G}_{3.1|\beta=1}) = 1] - \Pr[\mathcal{D}(\mathbf{G}_{3|\beta=1}) = 1]\right| \Pr[\beta = 1] \\
&+ \left|\Pr[\mathcal{D}(\mathbf{G}_{3.2|\beta=1}) = 1] - \Pr[\mathcal{D}(\mathbf{G}_{3.1|\beta=1}) = 1]\right| \\
&+ \left|\Pr[\mathcal{D}(\mathbf{G}_{3.3|\beta=1}) = 1] - \Pr[\mathcal{D}(\mathbf{G}_{3.2|\beta=1}) = 1]\right| \\
&+ \left|\Pr[\mathcal{D}(\mathbf{G}_{4|\beta=1}) = 1] - \Pr[\mathcal{D}(\mathbf{G}_{3.3|\beta=1}) = 1]\right| \\
\leq\ &nk\ell(\mathtt{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D}}(\lambda) + \mathtt{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D}}(\lambda)) \\
&+ 2q_d\ell \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\} + \mathsf{negl}(\lambda).
\end{aligned} \tag{25}$$

Now, we can construct a PPT simulator $\mathcal{S}$ as shown in Fig. 10.[10] Obviously,

$$\mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda) \text{ (when } \beta = 0) = \mathbf{G}_{4|\beta=0}, \tag{26}$$

$$\mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda) \text{ (when } \beta = 1) = \mathbf{G}_{4|\beta=1}. \tag{27}$$

So we can write that

$$\mathbf{G}_4 := \mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda). \tag{28}$$

Combining Eq. (9), (14), (15) and (25), we obtain that

$$\begin{aligned}
&|\Pr[\mathcal{D}(\mathbf{G}_2) = 1] - \Pr[\mathcal{D}(\mathbf{G}_4) = 1]| \\
=\ &|(\Pr[\mathcal{D}(\mathbf{G}_2) = 1 \mid \beta = 0] \Pr[\beta = 0] + \Pr[\mathcal{D}(\mathbf{G}_2) = 1 \mid \beta = 1] \Pr[\beta = 1]) \\
&- (\Pr[\mathcal{D}(\mathbf{G}_4) = 1 \mid \beta = 0] \Pr[\beta = 0] + \Pr[\mathcal{D}(\mathbf{G}_4) = 1 \mid \beta = 1] \Pr[\beta = 1])|
\end{aligned} \tag{29}$$

[10] In Fig. 10, for consistency we abuse the notation "n" by using it to mean the current number of receivers in the game simulated by $\mathcal{S}$, analogous to Fig. 2.

$\mathcal{S}_1^{\mathtt{SimMkRec}}(1^\lambda)$:

$(\mathsf{prm} := (\mathcal{X}, \mathcal{L}, \mathcal{L}_1, \cdots, \mathcal{L}_k), td) \leftarrow \mathsf{SmpG}(1^\lambda)$

$\mathsf{prmins} = (\mathcal{K}_{sp} = \mathcal{K}_a \times \mathcal{K}_b, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \mu) \leftarrow \mathsf{PrmG}(\mathsf{prm})$; $\mathsf{H} \leftarrow \mathcal{H}_{\mathsf{prmins}}$; $K_a \leftarrow \mathcal{K}_a$

$\mathsf{pp} := (\mathsf{prm}, \mathsf{prmins}, \mathsf{H}, K_a)$; $n := 0$; $\mathcal{C} := \emptyset$; $(\beta, \mathcal{M}, s_1) \leftarrow \mathcal{A}_1^{\mathtt{MkRec},\mathtt{Dec}}(\mathsf{pp})$

If $\exists\ (i,\gamma) \neq (i',\gamma')$ s.t. $hsk_{i,\gamma} = hsk_{i',\gamma'}$: abort

$\widetilde{s}_1 := (\mathsf{pp}, (pk_i, sk_i)_{i\in[n]}, \mathcal{C}, \beta, \mathcal{M}, s_1)$

Return $(\beta, \mathcal{M}, \widetilde{s}_1)$

$\mathcal{S}_2(\mathsf{len} = ((|\mathbf{m}_i^*|, \ell)_{i\in[n]}), \widetilde{s}_1 = (\mathsf{pp}, (pk_i, sk_i)_{i\in[n]}, \mathcal{C}, \beta, \mathcal{M}, s_1))$:

For $i = 1$ to n:

  For $j = 1$ to $|\mathbf{m}_i^*|$:

    For $\gamma = 1$ to ℓ:

      If $\beta = 0$:

        $w^*_{i,j,\gamma} \leftarrow \mathcal{R}_{\mathsf{SSmpL}}$; $x^*_{i,j,\gamma} \leftarrow \mathsf{SSmpL}(\mathsf{prm}; w^*_{i,j,\gamma})$; $K^*_{i,j,\gamma} = \mathsf{PubEv}(hpk_{i,\gamma}, x^*_{i,j,\gamma}, w^*_{i,j,\gamma})$

      If $\beta = 1$:

        If $|\mathbf{m}_i^*| \le k$:

          $x^*_{i,j,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j)$; $K^*_{i,j,\gamma} = \mathsf{SecEv}(hsk_{i,\gamma}, x^*_{i,j,\gamma})$

        If $|\mathbf{m}_i^*| > k$:

          If $m^*_{i,j,\gamma} = 0$: $(r^{*(\mathcal{X})}_{i,j,\gamma}, r^{*(\mathcal{K})}_{i,j,\gamma}) \leftarrow \mathcal{R}_{\mathsf{SSmpX}} \times \mathcal{R}_{\mathsf{Sample}}$; $x^*_{i,j,\gamma} \leftarrow \mathsf{SSmpX}(\mathsf{prm}; r^{*(\mathcal{X})}_{i,j,\gamma})$; $K^*_{i,j,\gamma} \leftarrow \mathsf{Sample}(\mathcal{K}_{sp}; r^{*(\mathcal{K})}_{i,j,\gamma})$

          If $m^*_{i,j,\gamma} = 1$: $w^*_{i,j,\gamma} \leftarrow \mathcal{R}_{\mathsf{SSmpL}}$; $x^*_{i,j,\gamma} \leftarrow \mathsf{SSmpL}(\mathsf{prm}; w^*_{i,j,\gamma})$; $K^*_{i,j,\gamma} = \mathsf{PubEv}(hpk_{i,\gamma}, x^*_{i,j,\gamma}, w^*_{i,j,\gamma})$

    $K^*_{b,i,j} = \mathsf{H}(pk_i, x^*_{i,j,1}, \cdots, x^*_{i,j,\ell})$; $K^*_{i,j,\ell+1} = (K_a, K^*_{b,i,j})$

    $T^*_{i,j} = \mathsf{XAuth}(K^*_{i,j,1}, \cdots, K^*_{i,j,\ell+1})$; $\mathbf{c}_i^*[j] = (x^*_{i,j,1}, \cdots, x^*_{i,j,\ell}, T^*_{i,j})$; $\mathcal{C} = \mathcal{C} \bigcup \{(i, \mathbf{c}_i^*[j])\}$

If $(\beta = 0) \wedge (\exists\ (i',j',\gamma') \neq (i'',j'',\gamma'')$ s.t. $x^*_{i',j',\gamma'} = x^*_{i'',j'',\gamma''})$: abort

$(\mathcal{I}, s_2) \leftarrow \mathcal{A}_2^{\mathsf{Dec}}((\mathbf{c}_1^*, \cdots, \mathbf{c}_n^*), s_1)$; $\widetilde{s}_2 = (\mathsf{pp}, (pk_i, sk_i)_{i\in[n]}, \mathcal{C}, \beta, (\mathbf{c}_1^*, \cdots, \mathbf{c}_n^*), s_2)$

Return $(\mathcal{I}, \widetilde{s}_2)$

$\mathcal{S}_3((\mathbf{m}_i^*[j])_{(i,j)\in\mathcal{I}}, \widetilde{s}_2)$: % When $\beta = 0$

Parse $(\mathbf{m}_i^*[j] = (m^*_{i,j,1}, \cdots, m^*_{i,j,\ell}) \in \{0,1\}^\ell)_{(i,j)\in\mathcal{I}}$

For $(i,j) \in \mathcal{I}$:

  For $\gamma = 1$ to ℓ:

    If $m^*_{i,j,\gamma} = 0$: $\widehat{w}^*_{i,j,\gamma} \leftarrow \mathcal{R}_{\mathsf{SSmpL}}$; $\widehat{r}^{*(\mathcal{X})}_{i,j,\gamma} \leftarrow \mathsf{Explain}(\mathcal{X}, x^*_{i,j,\gamma})$; $\widehat{K}^*_{i,j,\gamma} \leftarrow \mathsf{ReSamp}(\gamma, K^*_{i,j,\neq\gamma}, T^*_{i,j})$; $K^*_{i,j,\gamma} = \widehat{K}^*_{i,j,\gamma}$; $\widehat{r}^{*(\mathcal{K})}_{i,j,\gamma} \leftarrow \mathsf{Explain}(\mathcal{XK}, K^*_{i,j,\gamma})$

    If $m^*_{i,j,\gamma} = 1$: $\widehat{w}^*_{i,j,\gamma} = w^*_{i,j,\gamma}$; $\widehat{r}^{*(\mathcal{X})}_{i,j,\gamma} \leftarrow \mathcal{R}_{\mathsf{SSampX}}$; $\widehat{r}^{*(\mathcal{K})}_{i,j,\gamma} \leftarrow \mathcal{R}_{\mathsf{Sample}}$

  $\widehat{\mathbf{r}}_i^*[j] = (\widehat{r}^{*(\mathcal{X})}_{i,j,\gamma}, \widehat{r}^{*(\mathcal{K})}_{i,j,\gamma}, \widehat{w}^*_{i,j,\gamma})_{\gamma\in[\ell]}$

$out \leftarrow \mathcal{A}_3^{\mathsf{Dec}}((\widehat{\mathbf{r}}_i^*[j], \mathbf{m}_i^*[j])_{(i,j)\in\mathcal{I}}, s_2)$

Return out

$\mathcal{S}_3((\mathbf{m}_i^*)_{i\in\mathcal{I}}, \widetilde{s}_2)$: % When $\beta = 1$

Parse $(\mathbf{m}_i^*[j] = (m^*_{i,j,1}, \cdots, m^*_{i,j,\ell}) \in \{0,1\}^\ell)_{i\in\mathcal{I}, j\in[|\mathbf{m}_i^*|]}$

For $i \in \mathcal{I}$:

  For $\gamma = 1$ to ℓ:

    For $j = 1$ to $|\mathbf{m}_i^*|$:

      If $m^*_{i,j,\gamma} = 1$: $y_{i,j,\gamma} = x^*_{i,j,\gamma}$; $y_{i,j+k,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j+k)$

      If $m^*_{i,j,\gamma} = 0$: $y_{i,j+k,\gamma} = x^*_{i,j,\gamma}$; $y_{i,j,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j+k)$

    If $|\mathbf{m}_i^*| < k$:

      For $j = |\mathbf{m}_i^*| + 1$ to k: $y_{i,j,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j)$; $y_{i,j+k,\gamma} \leftarrow \mathsf{SSmpLS}(\mathsf{prm}, j+k)$

    $hsk'_{i,\gamma} \leftarrow \mathsf{SampHsk}(hsk_{i,\gamma}, td, y_{i,1,\gamma}, \cdots, y_{i,2k,\gamma})$; $hsk_{i,\gamma} := hsk'_{i,\gamma}$

  $sk_i = (hsk_{i,\gamma})_{\gamma\in[\ell]}$

$out \leftarrow \mathcal{A}_3^{\mathsf{Dec}}((sk_i, \mathbf{m}_i^*)_{i\in\mathcal{I}}, s_2)$

Return out

(The description of oracles `MkRec` and `Dec` will be given in Fig. 11).

Fig. 10 Construction of simulator $\mathcal{S}$.

$$
\begin{aligned}
&= |(\Pr[\mathcal{D}(\mathbf{G}_2) = 1 \mid \beta = 0]\Pr[\beta = 0] + \Pr[\mathcal{D}(\mathbf{G}_2) = 1 \mid \beta = 1]\Pr[\beta = 1]) \\
&\quad - (\Pr[\mathcal{D}(\mathbf{G}_{4|\beta=0}) = 1]\Pr[\beta = 0] + \Pr[\mathcal{D}(\mathbf{G}_{4|\beta=1}) = 1]\Pr[\beta = 1])| \qquad (30)
\end{aligned}
$$

$\mathtt{MkRec}()$:
$\perp \leftarrow \mathtt{SimMkRec}()$; $n := n+1$; $(pk_n = (hpk_{n,\gamma})_{\gamma\in[\ell]}, sk_n = (hsk_{n,\gamma})_{\gamma\in[\ell]}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
Return pk_n

$\mathtt{Dec}(i, c = (x_1, \cdots, x_\ell, T))$:
If $(i > n) \vee ((i,c) \in \mathcal{C})$: return $\perp$
Parse $sk_i = (hsk_{i,\gamma})_{\gamma\in[\ell]}$; $\overline{K}_b = \mathsf{H}(pk_i, x_1, \cdots, x_\ell)$
If $\exists\ j \in [|\mathbf{m}_i^*|]$, s.t. $(x_1, \cdots, x_\ell) \neq (x^*_{i,j,1}, \cdots, x^*_{i,j,\ell})$ and $\overline{K}_b = K^*_{b,i,j}$: abort
If $\mathsf{XVer}((K_a, \overline{K}_b), \ell+1, T) = 0$: $\overline{m}_1 = \cdots = \overline{m}_\ell = 0$; return $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$
For $\gamma = 1$ to ℓ:
  $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x_\gamma)$; $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T)$
Return $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$

Fig. 11 The oracles $\mathtt{MkRec}$ and $\mathtt{Dec}$ provided by simulator $\mathcal{S}$ in Fig. 10.

$$
\begin{aligned}
&\leq |\Pr[\mathcal{D}(\mathbf{G}_2)=1 \mid \beta=0] - \Pr[\mathcal{D}(\mathbf{G}_{4|\beta=0})=1]|\Pr[\beta=0] \\
&\quad + |\Pr[\mathcal{D}(\mathbf{G}_2)=1 \mid \beta=1] - \Pr[\mathcal{D}(\mathbf{G}_{4|\beta=1})=1]|\Pr[\beta=1] && (31)\\
&\leq (\big|\Pr[\mathcal{D}(\mathbf{G}_{3|\beta=0})=1] - \Pr[\mathcal{D}(\mathbf{G}_{4|\beta=0})=1]\big| + \frac{n\widetilde{k}\ell(n\widetilde{k}\ell-1)}{2|\mathcal{L}|})\Pr[\beta=0] \\
&\quad + |\Pr[\mathcal{D}(\mathbf{G}_{3|\beta=1})=1] - \Pr[\mathcal{D}(\mathbf{G}_{4|\beta=1})=1]|\Pr[\beta=1] && (32)\\
&\leq \big|\Pr[\mathcal{D}(\mathbf{G}_{3|\beta=0})=1] - \Pr[\mathcal{D}(\mathbf{G}_{4|\beta=0})=1]\big|\Pr[\beta=0] + \frac{n\widetilde{k}\ell(n\widetilde{k}\ell-1)}{2|\mathcal{L}|} \\
&\quad + |\Pr[\mathcal{D}(\mathbf{G}_{3|\beta=1})=1] - \Pr[\mathcal{D}(\mathbf{G}_{4|\beta=1})=1]|\Pr[\beta=1] && (33)\\
&\leq \mathsf{negl}(\lambda) + \frac{n\widetilde{k}\ell(n\widetilde{k}\ell-1)}{2|\mathcal{L}|} + nk\ell(\mathtt{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D}}(\lambda) + \mathtt{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D}}(\lambda)) \\
&\quad + 2q_d\ell\max\{\mathtt{Adv}^{\text{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\text{SUB}}_{\mathsf{XAC}}(\lambda)\} + \mathsf{negl}(\lambda) && (34)\\
&\leq \frac{n\widetilde{k}\ell(n\widetilde{k}\ell-1)}{2|\mathcal{L}|} + nk\ell(\mathtt{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D}}(\lambda) + \mathtt{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D}}(\lambda)) \\
&\quad + 2q_d\ell\max\{\mathtt{Adv}^{\text{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\text{SUB}}_{\mathsf{XAC}}(\lambda)\} + \mathsf{negl}(\lambda), && (35)
\end{aligned}
$$

where $\widetilde{k} = \max_{i=1}^{n} |\mathbf{m}_i^*|$. Eq. (29)-(31) are trivial. Eq. (32) is justified by Eq. (9) and (15). Eq. (33) is trivial. Eq. (34) is obtained via Eq. (14) and (25). Eq. (35) is trivially obtaind from Eq. (34).

Hence, combining Eq. (6)-(8), (28) and (35), we obtain that for any PPT distinguisher $\mathcal{D}$,

$$
\begin{aligned}
&\mathtt{Adv}^{\text{SIM-wBi-SO}_k\text{-CCA}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda) \\
&= \Big|\Pr[\mathcal{D}(\mathtt{Exp}^{\text{wBi-SO-real}}_{\mathsf{PKE},\mathcal{A},k}(\lambda))=1] - \Pr[\mathcal{D}(\mathtt{Exp}^{\text{wBi-SO-ideal}}_{\mathsf{PKE},\mathcal{S},k}(\lambda))=1]\Big| \\
&= |\Pr[\mathcal{D}(\mathbf{G}_0)=1] - \Pr[\mathcal{D}(\mathbf{G}_4)=1]| \\
&\leq |\Pr[\mathcal{D}(\mathbf{G}_2)=1] - \Pr[\mathcal{D}(\mathbf{G}_4)=1]| + \frac{n\ell(n\ell-1)}{2|\mathcal{SK}|} + \mathtt{Adv}^{\text{CR}}_{\mathcal{H}_{\mathsf{prmins}},\mathcal{A}_{\mathsf{H}}}(\lambda)
\end{aligned}
$$

$$\leq \frac{n\widetilde{k}\ell(n\widetilde{k}\ell-1)}{2|\mathcal{L}|} + nk\ell(\mathtt{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D}}(\lambda) + \mathtt{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D}}(\lambda))$$
$$+ 2q_d\ell \max\{\mathtt{Adv}^{\text{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\text{SUB}}_{\mathsf{XAC}}(\lambda)\} + \mathsf{negl}(\lambda)$$
$$+ \frac{n\ell(n\ell-1)}{2|\mathcal{SK}|} + \mathtt{Adv}^{\text{CR}}_{\mathcal{H}_{\mathsf{prmins}},\mathcal{A}_{\mathsf{H}}}(\lambda),$$

which is negligible.

Now we catch up with the proof of Lemma 1 and Lemma 2.

Proof (of Lemma 1). We prove this lemma with another series of games.

Game $\mathbf{G}^{s(0)}_{(i,j,\gamma-1)}$: For convenience, we write that

$$\mathbf{G}^{s(0)}_{(i,j,\gamma-1)} = \mathbf{G}^{s}_{(i,j,\gamma-1)}. \tag{36}$$

Game $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$: $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$ is identical to $\mathbf{G}^{s(0)}_{(i,j,\gamma-1)}$, except that if $(i, j) \in \mathcal{I}$ and $m^*_{i,j,\gamma} = 0$, instead of returning the original $(r^{*(\mathcal{X})}_{i,j,\gamma}, r^{*(\mathcal{K})}_{i,j,\gamma})$ to answer the selective opening query, the challenger returns $(\mathsf{Explain}(\mathcal{X}, x^*_{i,j,\gamma}), \mathsf{Explain}(\mathcal{XK}, K^*_{i,j,\gamma}))$. Obviously,

$$\mathbf{G}^{s(1)}_{(i,j,\gamma-1)} = \mathbf{G}^{s(0)}_{(i,j,\gamma-1)}. \tag{37}$$

Game $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$: $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$ is identical to $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$, except for the decryption oracle. In game $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$, for any decryption query $(i, c = (x_1, \cdots, x_\ell, T))$ satisfying that $i \leq n$ and $(i, c) \notin \mathcal{C}$, the challenger firstly checks whether $\mathsf{XVer}((K_a, \mathsf{H}(pk_i, x_1, \cdots, x_\ell)), T) = 0$. If so, it sets that $\overline{m}_1 = \cdots = \overline{m}_\ell = 0$ and returns $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$. Otherwise, the challenger sets $\overline{m}_\gamma = 0$ directly if $x_\gamma \notin \mathcal{L}$, and behaves just as in $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$ if $x_\gamma \in \mathcal{L}$ (i.e., compute $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x_\gamma)$ and set $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T)$); for any $\gamma' \neq \gamma$, the challenger generates $\overline{m}_{\gamma'}$ as in $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$.

Note that the decryption oracle in $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$ is inefficient, and it doesn't leak any information on $hsk_{i,\gamma}$ beyond $hpk_{i,\gamma}$.

Let $\mathsf{BAD}^{(2)}_{i,j,\gamma}$ (resp. $\mathsf{BAD}^{(1)}_{i,j,\gamma}$) denote the event that in game $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$ (resp. $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$), adversary $\mathcal{A}$ submits a decryption query $(i, c = (x_1, \cdots, x_\ell, T))$, such that $i \leq n$, $(i, c) \notin \mathcal{C}$, $\mathsf{XVer}((K_a, \mathsf{H}(pk_i, x_1, \cdots, x_\ell)), T) = 1$, $x_\gamma \notin \mathcal{L}$ and $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T) = 1$. Note that $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$ and $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$ are identical as long as the respective events $\mathsf{BAD}^{(2)}_{i,j,\gamma}$ and $\mathsf{BAD}^{(1)}_{i,j,\gamma}$ do not

occur, and that $\Pr[\mathsf{BAD}^{(2)}_{i,j,\gamma}] = \Pr[\mathsf{BAD}^{(1)}_{i,j,\gamma}]$. So for any (even unbounded) distinguisher $\mathcal{D}_s$,

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}) = 1]\right| \leq \Pr[\mathsf{BAD}^{(2)}_{i,j,\gamma}].$$

Since $x_\gamma \notin \mathcal{L}$ and HPS is perfectly universal$_{k+1}$, from $\mathcal{A}$'s point of view, $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x_\gamma)$ is uniformly distributed over $\mathcal{K}_{sp} = \mathcal{X}\mathcal{K}$. Thus, security against impersonation attacks of XAC guarantees that the probability that $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T) = 1$ is at most $\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda)$. Considering that $\mathcal{A}$ always makes q_d decryption queries, we have that $\Pr[\mathsf{BAD}^{(2)}_{i,j,\gamma}] \leq q_d \mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda)$. Hence,

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}) = 1]\right| \leq q_d \mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda). \quad (38)$$

Game $\mathbf{G}^{s(3)}_{(i,j,\gamma-1)}$: $\mathbf{G}^{s(3)}_{(i,j,\gamma-1)}$ is the same as $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$, except for the generation of $K^*_{i,j,\gamma}$ during the generation of $c^*_{i,j}$. Specifically, in $\mathbf{G}^{s(3)}_{(i,j,\gamma-1)}$, if $m^*_{i,j,\gamma} = 0$, set $K^*_{i,j,\gamma} = \mathsf{SecEv}(hsk_{i,\gamma}, x^*_{i,j,\gamma})$ and the corresponding random coin of $K^*_{i,j,\gamma}$ is opened as $\mathsf{Explain}(\mathcal{X}\mathcal{K}, K^*_{i,j,\gamma})$.

Note that when $m^*_{i,j,\gamma} = 0$, $x^*_{i,j,\gamma}$ is uniformly sampled from $\mathcal{X}$. Subset sparseness of SSMP implies that $x^*_{i,j,\gamma} \notin \mathcal{L}$ with probability $1 - \mathtt{Spar}_{\mathsf{SSMP}}(\lambda)$. When $x^*_{i,j,\gamma} \notin \mathcal{L}$, perfect universality$_{k+1}$ of HPS guarantees that $K^*_{i,j,\gamma}$ is uniformly distributed over $\mathcal{K}_{sp}$, which is the same as that in $\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}$. Hence, for any (even unbounded) distinguisher $\mathcal{D}_s$,

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}^{s(3)}_{(i,j,\gamma-1)}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}^{s(2)}_{(i,j,\gamma-1)}) = 1]\right| \leq \mathtt{Spar}_{\mathsf{SSMP}}(\lambda). \quad (39)$$

Game $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$: In this game, we modify the generation of $K^*_{i,j,\gamma}$ again. Specifically, if $m^*_{i,j,\gamma} = 0$, after generating $c^*_{i,j} = (x^*_{i,j,1}, \cdots, x^*_{i,j,\ell}, T^*_{i,j})$ as above, the challenger further updates $K^*_{i,j,\gamma}$ as

$$K^*_{i,j,\gamma} \leftarrow \mathsf{ReSamp}(\gamma, K^*_{i,j,\neq\gamma}, T^*_{i,j}).$$

Note that in this case, the corresponding opened random coin of $K^*_{i,j,\gamma}$ is actually $\mathsf{Explain}(\mathcal{X}\mathcal{K}, \mathsf{ReSamp}(\gamma, K^*_{i,j,\neq\gamma}, T^*_{i,j}))$.

Strongness of XAC guarantees that the updated $K^*_{i,j,\gamma}$ in $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$ and the one in $\mathbf{G}^{s(3)}_{(i,j,\gamma-1)}$ are statistically indistinguishable, even given $K^*_{i,j,\neq\gamma}$ and $T^*_{i,j}$. In other words, we have that for any (even unbounded) distinguisher $\mathcal{D}_s$,

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}^{s(3)}_{(i,j,\gamma-1)}) = 1]\right| \leq \mathtt{StD}^{\mathrm{STRN}}_{\mathsf{XAC}}(\lambda), \quad (40)$$

where $\mathtt{StD}^{\mathrm{STRN}}_{\mathsf{XAC}}(\lambda)$ is defined in Definition 4, which is actually the statistical distance between the $K^*_{i,j,\gamma}$ in $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$ and the one in $\mathbf{G}^{s(3)}_{(i,j,\gamma-1)}$.

Game $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$: $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$ is the same as $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$, except that the decryption oracle works with the original decryption rule. In other words, in $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$, for any decryption query $(i, c = (x_1, \cdots, x_\ell, T))$ satisfying that $i \leq n$ and $(i, c) \notin \mathcal{C}$, the challenger firstly checks whether $\mathsf{XVer}((K_a, \mathsf{H}(pk_i, x_1, \cdots, x_\ell)), T) = 0$. If so, it sets that $\overline{m}_1 = \cdots = \overline{m}_\ell = 0$ and returns $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$. Otherwise, for any $\gamma' \in [\ell]$, the challenger computes $\overline{K}_{\gamma'} = \mathsf{SecEv}(hsk_{i,\gamma'}, x_{\gamma'})$, sets $\overline{m}_{\gamma'} = \mathsf{XVer}(\overline{K}_{\gamma'}, T)$, and returns $\overline{m} = (\overline{m}_1, \cdots, \overline{m}_\ell)$.

Notice that the decryption oracle in game $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$ is efficient again.

Similarly, let $\mathsf{BAD}^{(5)}_{i,j,\gamma}$ (resp. $\mathsf{BAD}^{(4)}_{i,j,\gamma}$) denote the event that in game $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$ (resp. $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$), adversary $\mathcal{A}$ submits a decryption query $(i, c = (x_1, \cdots, x_\ell, T))$, such that $i \leq n$, $(i, c) \notin \mathcal{C}$, $\mathsf{XVer}((K_a, \mathsf{H}(pk_i, x_1, \cdots, x_\ell)), T) = 1$, $x_\gamma \notin \mathcal{L}$ and $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T) = 1$. Note that $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$ and $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$ are identical as long as the respective events $\mathsf{BAD}^{(5)}_{i,j,\gamma}$ and $\mathsf{BAD}^{(4)}_{i,j,\gamma}$ do not occur, and that $\Pr[\mathsf{BAD}^{(5)}_{i,j,\gamma}] = \Pr[\mathsf{BAD}^{(4)}_{i,j,\gamma}]$. So for any (even unbounded) distinguisher $\mathcal{D}_s$,

$$\left|\Pr[\mathcal{D}_s(\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}) = 1]\right| \leq \Pr[\mathsf{BAD}^{(4)}_{i,j,\gamma}].$$

We present the following claim with a postponed proof.

Claim 1. $\Pr[\mathsf{BAD}^{(4)}_{i,j,\gamma}] \leq q_d \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\}$.

Hence, we have that

$$\begin{aligned}&\left|\Pr[\mathcal{D}_s(\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}) = 1]\right| \\ \leq\ & q_d \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\}. \end{aligned} \tag{41}$$

Game $\mathbf{G}^{s(6)}_{(i,j,\gamma-1)}$: $\mathbf{G}^{s(6)}_{(i,j,\gamma-1)}$ is the same as $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$, except that during the generation of $\mathbf{c}^*_i[j]$, the challenger computes $x^*_{i,j,\gamma} \leftarrow \mathsf{SSampL}(\mathsf{prm}; w^*_{i,j,\gamma})$ no matter whether $m^*_{i,j,\gamma}$ is 1 or 0. We stress that if $m^*_{i,j,\gamma} = 0$, the corresponding random coin of $x^*_{i,j,\gamma}$ returned to $\mathcal{A}$ is $\mathsf{Explain}(\mathcal{X}, x^*_{i,j,\gamma})$ since the modification introduced in $\mathbf{G}^{s(1)}_{(i,j,\gamma-1)}$. Hence, we obtain that for any PPT distinguisher $\mathcal{D}$,

$$\left|\Pr[\mathcal{D}(\mathbf{G}^{s(6)}_{(i,j,\gamma-1)}) = 1 \mid \beta = 0] - \Pr[\mathcal{D}(\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}) = 1 \mid \beta = 0]\right| \Pr[\beta = 0]$$

$$\leq \mathsf{Adv}^{\text{HARD-1}}_{\mathsf{SSMP},\mathcal{D}_{\mathsf{SSMP}}}(\lambda) + \mathsf{Adv}^{\text{HARD-2}}_{\mathsf{SSMP},\mathcal{D}_{\mathsf{SSMP}}}(\lambda), \tag{42}$$

where $\mathcal{D}_{\mathsf{SSMP}}$ is a suitable PPT adversary built based on $\mathcal{A}$ (more specifically, if $\mathcal{A}_1$ returns $\beta = 1$, $\mathcal{D}_{\mathsf{SSMP}}$ returns a uniformly chosen b'; else, $\mathcal{D}_{\mathsf{SSMP}}$ simulates one of the two games $\mathbf{G}^{s(5)}_{(i,j,\gamma-1)}$ and $\mathbf{G}^{s(6)}_{(i,j,\gamma-1)}$).

Game $\mathbf{G}^{s(7)}_{(i,j,\gamma-1)}$: $\mathbf{G}^{s(7)}_{(i,j,\gamma-1)}$ is the same as $\mathbf{G}^{s(6)}_{(i,j,\gamma-1)}$, except for the generation of $K^*_{i,j,\gamma}$ when $m^*_{i,j,\gamma} = 0$. Specifically, in $\mathbf{G}^{s(7)}_{(i,j,\gamma-1)}$, if $m^*_{i,j,\gamma} = 0$, the challenger computes $K^*_{i,j,\gamma} = \mathsf{PubEv}(hpk_{i,\gamma}, x^*_{i,j,\gamma}, w^*_{i,j,\gamma})$, instead of $\mathsf{SecEv}(hsk_{i,\gamma}, x^*_{i,j,\gamma})$. The projective property of HPS guarantees that the view of $\mathcal{A}$ in $\mathbf{G}^{s(7)}_{(i,j,\gamma-1)}$ are identical to that in $\mathbf{G}^{s(6)}_{(i,j,\gamma-1)}$.

Note that

$$\mathbf{G}^{s}_{(i,j,\gamma)} = \mathbf{G}^{s(7)}_{(i,j,\gamma-1)}. \tag{43}$$

So combining Eq. (36)-(43) finishes the proof of Lemma 1.

What remains is to prove Claim 1.

Proof (of Claim 1). Note that $\mathsf{BAD}^{(4)}_{i,j,\gamma}$ denotes the event that in $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$, $\mathcal{A}$ submits a decryption query $(i, c = (x_1, \cdots, x_\ell, T))$, such that $i \leq n$, $(i, c) \notin \mathcal{C}$, $\mathsf{XVer}((K_a, \mathsf{H}(pk_i, x_1, \cdots, x_\ell)), T) = 1$, $x_\gamma \notin \mathcal{L}$ and $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T) = 1$.

If $x_\gamma \neq x^*_{i,j,\gamma}$, the perfect universality$_{k+1}$ of HPS implies that $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x_\gamma)$ is uniformly distributed over $\mathcal{K}_{sp}$ from $\mathcal{A}$'s point of view, since the only possible information $\mathcal{A}$ has on $hsk_{i,\gamma}$ beyond $hpk_{i\gamma}$ is $K^*_{i,j,\gamma}$, and $K^*_{i,j,\gamma}$ is not equal but related to $\mathsf{SecEvl}(hsk_{i,\gamma}, x^*_{i,j,\gamma})$ in game $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$ (note that $K^*_{i,j,\gamma}$ has been updated with algorithm ReSamp in $\mathbf{G}^{s(4)}_{(i,j,\gamma-1)}$). In this case, the probability that $\mathsf{BAD}^{(4)}_{i,j,\gamma}$ occurs is at most $q_d\mathsf{Adv}^{\text{IMP}}_{\mathsf{XAC}}(\lambda)$.

If $x_\gamma = x^*_{i,j,\gamma}$ and $T = T^*_{i,j}$, then $(x_1, \cdots, x_\ell) \neq (x^*_{i,j,1}, \cdots, x^*_{i,j,\ell})$ (otherwise $(i, c) \in \mathcal{C}$). Because game $\mathbf{G}_2$ excludes hash collisions, we have that $\overline{K}_b \neq K^*_{b,i,j}$. Semi-uniqueness of XAC guarantees that $\overline{m}_1 = \cdots = \overline{m}_\ell = 0$, so in this case, $\mathsf{BAD}^{(4)}_{i,j,\gamma}$ does not occur.

If $x_\gamma = x^*_{i,j,\gamma}$ but $T \neq T^*_{i,j}$, then $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x^*_{i,j,\gamma})$, and $\mathsf{BAD}^{(4)}_{i,j,\gamma}$ occurs only if $\mathsf{XVer}(\overline{K}_\gamma, T) = 1$. Note that in this case, what $\mathcal{A}$ knows about $hsk_{i,\gamma}$ beyond $hpk_{i,\gamma}$ is given by $(K^*_{i,j,1}, \cdots, K^*_{i,j,\ell})$ and $T^*_{i,j}$. Since $K^*_{i,j,\gamma}$ is computed with $\mathsf{ReSamp}(\gamma, K^*_{i,j,\neq\gamma}, T^*_{i,j})$, $\mathcal{A}$'s information about $hsk_{i,\gamma}$ beyond $hpk_{i,\gamma}$ is actually from $K^*_{i,j,\neq\gamma}$ and $T^*_{i,j}$. Hence, for each decryption query, the probability that $\mathcal{A}$ generates a $T \neq T^*_{i,j}$

such that $\mathsf{XVer}(\overline{K}_\gamma, T) = \mathsf{XVer}(\mathsf{SecEv}(hsk_{i,\gamma}, x^*_{i,j,\gamma}), T) = 1$ is at most $\mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)$. So in this case, the probability that $\mathsf{BAD}^{(4)}_{i,j,\gamma}$ occurs is at most $q_d \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)$.

Therefore, $\Pr[\mathsf{BAD}^{(4)}_{i,j,\gamma}] \leq q_d \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\}$. □

□

Proof (of Lemma 2). Let $\mathsf{BAD}_{3.2}$ (resp. $\mathsf{BAD}_{3.1}$) denote the event that in game $\mathbf{G}_{3.2|\beta=1}$ (resp. $\mathbf{G}_{3.1|\beta=1}$), $\mathcal{A}$ submits a decryption query $(i \notin \mathcal{I}_{\mathsf{op}}, c = (x_1, \cdots, x_\ell, T)) \notin \mathcal{C}$ such that there is some $\gamma \in [\ell]$ satisfying that $x_\gamma \notin \mathcal{L}$ but $\overline{m}_\gamma = 1$. Note that $\mathbf{G}_{3.2|\beta=1}$ and $\mathbf{G}_{3.1|\beta=1}$ are identical as long as the respective events $\mathsf{BAD}_{3.2}$ and $\mathsf{BAD}_{3.1}$ do not occur, and that $\Pr[\mathsf{BAD}_{3.2}] = \Pr[\mathsf{BAD}_{3.1}]$. So for any (even unbounded) distinguisher $\mathcal{D}_s$,

$$|\Pr[\mathcal{D}_s(\mathbf{G}_{3.2|\beta=1}) = 1] - \Pr[\mathcal{D}_s(\mathbf{G}_{3.1|\beta=1}) = 1]| \leq \Pr[\mathsf{BAD}_{3.2}]. \quad (44)$$

Now we compute $\Pr[\mathsf{BAD}_{3.2}]$.

For any $\theta \in [q_d]$ and any $\gamma \in [\ell]$, let $\mathsf{BAD}^{\theta,\gamma}_{3.2}$ denote the event that the x_γ of $\mathcal{A}$'s θ-th decryption query $(i, c = (x_1, \cdots, x_\ell, T))$ makes $\mathsf{BAD}_{3.2}$ occur for the first time. So we have $\Pr[\mathsf{BAD}_{3.2}] \leq \sum_{(\theta,\gamma)\in[q_d]\times[\ell]} \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}]$.

Fix $(\theta, \gamma) \in [q_d] \times [\ell]$. For $\mathcal{A}$'s θ-th decryption query $(i, c = (x_1, \cdots, x_\ell, T))$, we assume that $x_\gamma \notin \mathcal{L}$ (as necessary for $\mathsf{BAD}^{\theta,\gamma}_{3.2}$). Let F_1 denote the event that $x_\gamma \notin \{x^*_{i,1,\gamma}, \cdots, x^*_{i,|\mathbf{m}^*_i|,\gamma}\}$, F_2 denote the event that $x_\gamma = x^*_{i,j',\gamma}$ and $T = T^*_{i,j'}$ for some $j' \in [|\mathbf{m}^*_i|]$, and F_3 denote the event that $x_\gamma = x^*_{i,j',\gamma}$ for some $j' \in [|\mathbf{m}^*_i|]$ but $T \neq T^*_{i,j'}$. Thus,

$$\begin{aligned}\Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}] = \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}|\mathsf{F}_1] \cdot \Pr[\mathsf{F}_1] + \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}|\mathsf{F}_2] \cdot \Pr[\mathsf{F}_2] \\ + \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}|\mathsf{F}_3] \cdot \Pr[\mathsf{F}_3]. \quad (45)\end{aligned}$$

When $x_\gamma \notin \{x^*_{i,1,\gamma}, \cdots, x^*_{i,|\mathbf{m}^*_i|,\gamma}\}$ and $x_\gamma \notin \mathcal{L}$, since HPS is perfectly universal$_{k+1}$, $\overline{K}_\gamma = \mathsf{SEcEv}(hsk_{i,\gamma}, x_\gamma)$ is uniformly distributed over $\mathcal{K}_{sp} = \mathcal{X}\mathcal{K}$ from $\mathcal{A}$'s point of view. Thus, security against impersonation attacks of XAC guarantees that the probability that $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T) = 1$ is at most $\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda)$. So we have

$$\Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}|\mathsf{F}_1] \leq \mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda). \quad (46)$$

When there is some $j' \in [|\mathbf{m}^*_i|]$ such that $x_\gamma = x^*_{i,j',\gamma}$ and $T = T^*_{i,j'}$, considering that the decryption oracle does not return $\perp$, we derive that $(x_1, \cdots, x_\ell) \neq (x^*_{i,j',1}, \cdots, x^*_{i,j',\ell})$. Because of the modification introduced

in $\mathbf{G}_2$, $\overline{K}_b \neq K^*_{b,i,j'}$. Since $T = T^*_{i,j'}$, semi-uniqueness of XAC implies that $\Pr[\mathsf{XVer}((K_a, \overline{K}_b), T) = 1] = 0$. Hence,

$$\Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2} | \mathsf{F}_2] = 0. \tag{47}$$

When there is some $j' \in [|\mathbf{m}^*_i|]$ such that $x_\gamma = x^*_{i,j',\gamma}$ but $T \neq T^*_{i,j'}$, our reasoning is as follows. We note that $\overline{K}_\gamma = \mathsf{SecEv}(hsk_{i,\gamma}, x_\gamma)$ is uniformly distributed from $\mathcal{A}$'s point of view (when ignoring $T^*_{i,j'}$), since $x_\gamma = x^*_{i,j',\gamma} \in \mathcal{L}_{j'}$ and $i \notin \mathcal{I}_{\mathsf{op}}$. Hence, security against substitution attacks of XAC guarantees that the probability that $\overline{m}_\gamma = \mathsf{XVer}(\overline{K}_\gamma, T) = 1$ is at most $\mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)$. In other words, we have

$$\Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2} | \mathsf{F}_3] \leq \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda). \tag{48}$$

Combining Eq. (45)-(48), we derive that

$$\begin{aligned} \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}] &= \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2} | \mathsf{F}_1] \cdot \Pr[\mathsf{F}_1] + \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2} | \mathsf{F}_3] \cdot \Pr[\mathsf{F}_3] \\ &= \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\}. \end{aligned} \tag{49}$$

Therefore,

$$\Pr[\mathsf{BAD}_{3.2}] \leq \sum_{(\theta,\gamma) \in [q_d] \times [\ell]} \Pr[\mathsf{BAD}^{\theta,\gamma}_{3.2}] \leq q_d \ell \max\{\mathtt{Adv}^{\mathrm{IMP}}_{\mathsf{XAC}}(\lambda), \mathtt{Adv}^{\mathrm{SUB}}_{\mathsf{XAC}}(\lambda)\}. \quad \square$$

$\square$

F Security Notions for KEM and MAC

We recall the notion of one-way security in the presence of a plaintext-checking oracle (OW-PCA security) [29] for KEM, and the notion of strong unforgeability under one-time chosen message attacks (sUF-OT-CMA security) for MAC as follows.

Definition 12. (OW-PCA for KEM)[29]. *We say that a KEM scheme* $\mathsf{KEM} = (\mathsf{KemGen}, \mathsf{Encap}, \mathsf{Decap})$ *is* OW-PCA secure*, if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathtt{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := \Pr[\mathtt{Exp}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1]$ *is negligible, where experiment* $\mathtt{Exp}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ *is defined in Fig. 12.*

We write that $\mathsf{CollPK}_{\mathsf{KEM},n}(1^\lambda) := \Pr[\exists i \neq i', \text{ s.t. } pk_i = pk_{i'} : (pk_1, sk_1) \leftarrow \mathsf{KemGen}(1^\lambda), \cdots, (pk_n, sk_n) \leftarrow \mathsf{KemGen}(1^\lambda)]$. It is obvious that a PPT adversary $\mathcal{A}$ can be constructed, such that

$$\mathtt{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}}(\lambda) \geq \frac{2}{n} \mathsf{CollPK}_{\mathsf{KEM},n}(1^\lambda).$$

In other words, OW-PCA security guarantees that $\mathsf{CollPK}_{\mathsf{KEM},n}(1^\lambda)$ is negligible.

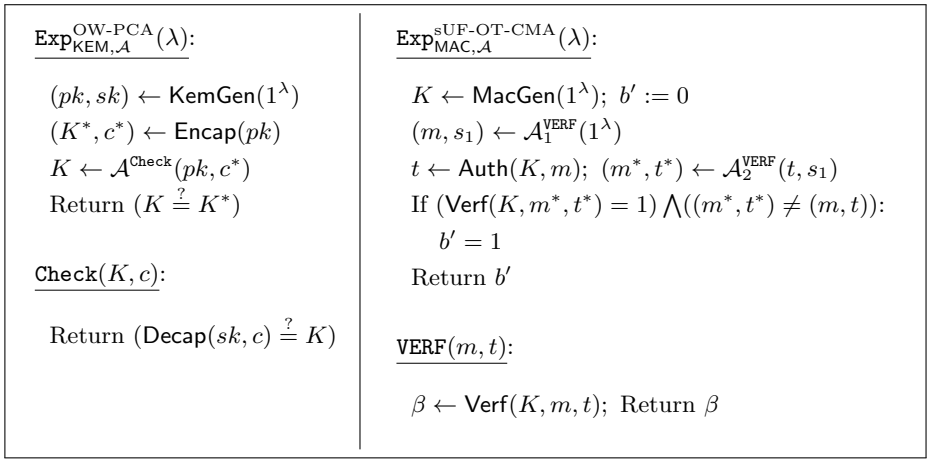

Fig. 12 Experiment for defining OW-PCA security of KEM, and experiment for defining sUF-OT-CMA security of MAC

Definition 13. (sUF-OT-CMA for MAC). *We say that a MAC scheme* $\mathsf{MAC} = (\mathsf{MacGen}, \mathsf{Auth}, \mathsf{Verf})$ *is* sUF-OT-CMA secure*, if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}}(\lambda) = 1]$ *is negligible, where experiment* $\mathbf{Exp}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}}(\lambda)$ *is defined in Fig. 12.*

G Deterred Proof of Theorem 6

We provide an intuition of our proof here. In order to prove SIM-Bi-SO-CCA security, we need to provide adversary $\mathcal{A}$ with some message-independent dummy ciphertexts $(\mathbf{c}_1^*, \cdots, \mathbf{c}_n^*)$, and then open them to the real messages. We proceed in a series of games.

The key point is that for any $i \in [n]$ and $j \in [|\mathbf{m}_i^*|]$, $c_{i,j}^{*sym} = K_{i,j}^{*sym} \oplus \mathbf{m}_i^*[j]$, i.e., given $c_{i,j}^{*sym}$ and $K_{i,j}^{*sym}$, $\mathbf{m}_i^*[j]$ is fixed. Hence, after generating the dummy ciphertext $\mathbf{c}_i^*[j]$ for $\mathcal{A}$ and before $\mathcal{A}$ submits a selective opening query $(\mathcal{I}_S, \mathcal{I}_R)$ such that $(i, j) \in \mathcal{I}_S$ or $i \in \mathcal{I}_R$, we need to block $\mathcal{A}$'s random oracle query on $K_{i,j}^*$, and decryption query on $(i', (c^{kem}, c^{sym}, t))$ satisfying $\mathsf{Decap}(sk_{i'}^{kem}, c^{kem}) = K_{i,j}^*$, since any such call would assign a value to $K_{i,j}^{*sym}$. (We stress that it's possible that $i' \neq i$ but meanwhile $\mathsf{Decap}(sk_{i'}^{kem}, c^{kem}) = K_{i,j}^*$. That's why we require that the MAC tag should be generated on a KEM ciphertext, a SE ciphertext and the public key of KEM.) Hence, from game $\mathbf{G}_3$ on, we will abort once $\mathcal{A}$ submits the above decryption query before querying the random oracle on $K_{i,j}^*$.

Note that if $\mathcal{A}$ did not query the random oracle on $K^*_{i,j}$ before, $K^{*mac}_{i,j}$ is still uniformly distributed. So the probability of abort is negligible because of sUF-OT-CMA security of MAC. Then from game $\mathbf{G}_4$ on, we will abort if $\mathcal{A}$ queries the random oracle on $K^*_{i,j}$. The probability of abort is also negligible due to OW-PCA security of KEM.

Proof (of Theorem 6). For any PPT adversary $\mathcal{A}$, in the real experiment $\mathtt{Exp}^{\text{Bi-SO-real}}_{\mathsf{PKE}_{\mathsf{K\text{-}M}},\mathcal{A}}(\lambda)$, we denote the challenge ciphertexts and their corresponding messages by $(\mathbf{c}^*_i)_{i\in[n]}$ and $(\mathbf{m}^*_i)_{i\in[n]}$, respectively. More specifically, for each $i \in [n]$ and $j \in [|\mathbf{m}^*_i|]$, we write $\mathbf{c}^*_i[j] = (c^{*kem}_{i,j}, c^{*sym}_{i,j}, t^*_{i,j})$. We write $\mathbf{r}^*_i[j]$, $K^*_{i,j}$, $K^{*sym}_{i,j}$ and $K^{*mac}_{i,j}$ similarly. For each $i \in [n]$, we write $pk_i = pk^{kem}_i$, and $sk_i = (pk^{kem}_i, sk^{kem}_i)$. Again, we stress that in the real experiment $\mathtt{Exp}^{\text{Bi-SO-real}}_{\mathsf{PKE}_{\mathsf{K\text{-}M}},\mathcal{A}}(\lambda)$, for any $i \in [n]$, pk_i is employed to encrypt $|\mathbf{m}^*_i|$ messages (i.e., $\mathbf{m}^*_i[1], \cdots, \mathbf{m}^*_i[|\mathbf{m}^*_i|]$).

In Fig. 13, denote by $\mathtt{HASH}_1$ (resp. $\mathtt{HASH}_2$) the random-oracle interface of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$ and $\mathcal{A}_3$), and denote by $\mathtt{Dec}_1$ (resp. $\mathtt{Dec}_2$) the decryption-oracle interface of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$ and $\mathcal{A}_3$).

Without loss of generality, we assume that after receiving $((\mathbf{r}^*_i[j], \mathbf{m}^*_i[j])_{(i,j)\in\mathcal{I}_S}, (sk_i, \mathbf{m}^*_i)_{i\in\mathcal{I}_R})$, the adversary (i.e., $\mathcal{A}_3$) will always query the random oracle H_{RO} on

- $K^*_{i,j}$ for each $(i,j) \in \mathcal{I}_S$, where $K^*_{i,j}$ is from $(K^*_{i,j}, c^{*kem}_{i,j}) \leftarrow \mathsf{Encap}(pk^{kem}_i; \mathbf{r}^*_i[j])$;
- $K_{i,j} = \mathsf{Decap}(sk_i, c^{*kem}_{i,j})$ for each $i \in \mathcal{I}_R$ and $j \in [|\mathbf{m}^*_i|]$.

We also assume that after receiving $(sk_i, \mathbf{m}^*_i)_{i\in\mathcal{I}_R}$, $\mathcal{A}$ will not query the decryption oracle on any $(i', \cdot)$ satisfying $i' \in \mathcal{I}_R$. Note that this is also without loss of generality, because $\mathcal{A}$ can decrypt the ciphertexts with $sk_{i'}$ by itself.

Let q_d (resp. q_r) denote the total number of decryption queries (resp. random-oracle queries) made by $\mathcal{A}$. We write $\widetilde{k} := \max_{i=1}^{n} |\mathbf{m}^*_i|$. We stress that $\widetilde{k}$ is not a fixed value, and it is totally determined by $\mathcal{A}$.

Since H_{RO} is modeled as a random oracle, we assume that the challenger maintains a local array L_{H} and employs it to keep track of issued calls (either by the game or $\mathcal{A}$) of $\mathsf{H}_{\mathrm{RO}}[\cdot]$. Specifically, for a query K, the random oracle returns $\mathsf{H}_{\mathrm{RO}}(K) = (K^{sym}, K^{mac})$ if there is an entry $(K, (K^{sym}, K^{mac})) \in L_{\mathsf{H}}$, otherwise it samples $(K^{sym}, K^{mac}) \leftarrow \{0,1\}^{\ell} \times \mathcal{K}_{\mathsf{MAC}}$, adds $(K, (K^{sym}, K^{mac}))$ to L_{H}, and returns (K^{sym}, K^{mac}) (we write $\mathsf{H}_{\mathrm{RO}}(K) := (K^{sym}, K^{mac})$ and implicitly assume an update operation $L_{\mathsf{H}} := L_{\mathsf{H}} \bigcup \{(K, (K^{sym}, K^{mac}))\}$ to happen in the background).

Now, we proceed in a series of games.

Fig. 13 Games $\mathbf{G}_0$-$\mathbf{G}_4$ in the proof of Theorem 6.

Game $\mathbf{G}_{-1}$: $\mathbf{G}_{-1}$ is the real experiment $\mathbf{Exp}^{\text{Bi-SO-real}}_{\mathsf{PKE}_{\mathsf{K\text{-}M}},\mathcal{A}}(\lambda)$, i.e.,

$$\mathbf{G}_{-1} = \mathbf{Exp}^{\text{Bi-SO-real}}_{\mathsf{PKE}_{\mathsf{K\text{-}M}},\mathcal{A}}(\lambda). \tag{50}$$

Game $\mathbf{G}_0$: $\mathbf{G}_0$ is the same as $\mathbf{G}_{-1}$, except that we abort this game (with output $\perp$) as soon as there are some $i \neq i'$ such that $pk_i^{kem} = pk_{i'}^{kem}$, or there are some $(i,j) \neq (i',j')$ such that $K_{i,j}^* = K_{i',j'}^*$. Since Encap uniformly samples K, by a union bound, we derive that for any PPT distinguisher $\mathcal{D}$,

$$\begin{aligned}|\Pr[\mathcal{D}(\mathbf{G}_0)=1] - \Pr[\mathcal{D}(\mathbf{G}_{-1})=1]| &\leq \mathsf{CollPK}_{\mathsf{KEM},n}(1^\lambda) + \frac{n\widetilde{k}(n\widetilde{k}-1)}{2|\mathcal{K}_{\mathsf{KEM}}|} \\ &\leq \frac{n}{2}\mathsf{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}_{\text{Coll}}}(\lambda) + \frac{n\widetilde{k}(n\widetilde{k}-1)}{2^{\lambda+1}} \quad (51)\end{aligned}$$

for a suitable PPT adversary $\mathcal{A}_{\text{Coll}}$, where $\widetilde{k} := \max_{i=1}^n |\mathbf{m}_i^*|$.

Game $\mathbf{G}_1$: Game $\mathbf{G}_1$ is the same as $\mathbf{G}_0$, except that we abort this game (with output $\perp$) as long as $\mathsf{AbortEARLY}$ occurs. Let $\mathsf{AbortEARLY}_1$ denote the event that $\mathcal{A}_1$ submits a random-oracle query K such that later there is some $K_{i',j'}^*$ (for some $i' \in [n]$ and $j' \in [|\mathbf{m}_{i'}^*|]$), generated by Encap, satisfying $K_{i',j'}^* = K$. Let $\mathsf{AbortEARLY}_2$ denote the event that $\mathcal{A}_1$ submits a decryption query $(i'', (c^{kem}, c^{sym}, t))$ such that later there is some $K_{i',j'}^*$ (for some $i' \in [n]$ and $j' \in [|\mathbf{m}_{i'}^*|]$), generated by Encap, satisfying $\mathsf{Decap}(sk_{i''}^{kem}, c^{kem}) = K_{i',j'}^*$.Obviously, $\mathsf{AbortEARLY}$ occurs if and only if $\mathsf{AbortEARLY}_1$ or $\mathsf{AbortEARLY}_2$ occurs.

Since Encap uniformly samples the session keys, for any $i \in [n]$ and any $j \in [|\mathbf{m}_i^*|]$, $K_{i,j}^*$ is uniformly sampled. Note that $\mathcal{A}_1$ makes its random-oracle queries and decryption queries before seeing the challenge ciphertexts $(\mathbf{c}_1^*, \cdots, \mathbf{c}_n^*)$, So it has no information about $K_{i,j}^*$ for any $i \in [n]$ and any $j \in [|\mathbf{m}_i^*|]$. Therefore, we derive that for any PPT distinguisher $\mathcal{D}$,

$$\begin{aligned}|\Pr[\mathcal{D}(\mathbf{G}_1)=1] - \Pr[\mathcal{D}(\mathbf{G}_0)=1]| &\leq \Pr[\mathsf{AbortEARLY}_1] + \Pr[\mathsf{AbortEARLY}_2] \\ &\leq \sum_{\theta=1}^{q_r} \frac{n\widetilde{k}}{2^\lambda - (\theta-1)} + \sum_{\theta=1}^{q_d} \frac{n\widetilde{k}}{2^\lambda - (\theta-1)} \\ &\leq \frac{n\widetilde{k}q_r}{2^\lambda - q_r} + \frac{n\widetilde{k}q_d}{2^\lambda - q_d}, \quad (52)\end{aligned}$$

where $\widetilde{k} = \max_{i=1}^n |\mathbf{m}_i^*|$.

Game $\mathbf{G}_2$: Game $\mathbf{G}_2$ is the same as $\mathbf{G}_1$, except that (i) the procedures "$(K_{i,j}^{*sym}, K_{i,j}^{*mac}) = \mathsf{H}_{\text{RO}}(K_{i,j}^*)$; $c_{i,j}^{*sym} = K_{i,j}^{*sym} \oplus \mathbf{m}_i^*[j]$" are replaced with "$(c_{i,j}^{*sym}, K_{i,j}^{*mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$" for all $i \in [n]$ and $j \in [|\mathbf{m}_i^*|]$, (ii) "$(\mathsf{H}_{\text{RO}}(K_{i,j}^*) = (c_{i,j}^{*sym} \oplus \mathbf{m}_i^*[j], K_{i,j}^{*mac}))_{(i,j)\in\mathcal{I}_S \bigcup\{(\widetilde{i},\widetilde{j})|\widetilde{i}\in\mathcal{I}_R,\widetilde{j}\in[|\mathbf{m}_{\widetilde{i}}^*|]\}}$" are added the generation of the answer to the selective opening query, and

(iii) if $\mathcal{A}_2$ and $\mathcal{A}_3$ submits a random-oracle query K such that $(K,\cdot) \notin L_{\mathsf{H}}$ and $K = K^*_{i',j'}$ for some $i' \in [n]$ and $j' \in [|\mathbf{m}^*_i|]$, the challenger sets that $\mathsf{H}_{\mathrm{RO}}(K^*_{i,j}) = (c^{*sym}_{i,j} \oplus \mathbf{m}^*_i[j], K^{*mac}_{i,j})$, as shown in Fig. 13.

We claim that for any (even unbounded) distinguisher $\mathcal{D}_s$,

$$\Pr[\mathcal{D}_s(\mathbf{G}_2) = 1] = \Pr[\mathcal{D}_s(\mathbf{G}_1) = 1]. \tag{53}$$

The reasons are as follows. Assuming neither $\mathsf{AbortEARLY}_1$ nor $\mathsf{AbortEARLY}_2$ happens in $\mathbf{G}_2$, for any $i \in [n]$ and $j \in [|\mathbf{m}^*_{i'}|]$, $(K^{*sym}_{i,j}, K^{*mac}_{i,j})$ is uniformly and independently distributed when $\mathcal{A}_1$ outputs $(\mathcal{M}, s_1)$. Hence, for each $i \in [n]$ and $j \in [|\mathbf{m}^*_{i'}|]$, $c^{*sym}_{i,j} = K^{*sym}_{i,j} \oplus \mathbf{m}^*_i[j]$ is also uniformly distributed, and $t^*_{i,j}$ is a valid tag of $(pk^{kem}_i, c^{*kem}_{i,j}, c^{*sym}_{i,j})$ under a key from the uniform distribution. Consequently, during the generation of $(\mathbf{c}^*_1, \cdots, \mathbf{c}^*_n)$, for each $i \in [n]$ and $j \in [|\mathbf{m}^*_{i'}|]$, the challenger can sample $(c^{*sym}_{i,j}, K^{*mac}_{i,j})$ uniformly and compute $t^*_{i,j}$ using $K^{*mac}_{i,j}$, without changing the distribution of $(\mathbf{c}^*_1, \cdots, \mathbf{c}^*_n)$. In order to keep H_{RO} consistent, if $\mathcal{A}_2$ queries the random oracle on $K = K^*_{i',j'}$ for some $i' \in [n]$ and $j' \in [|\mathbf{m}^*_{i'}|]$ (resp. submits a selective opening query $(\mathcal{I}_S, \mathcal{I}_R)$), the challenger sets that $\mathsf{H}_{\mathrm{RO}}(K^*_{i',j'}) = (c^{*sym}_{i',j'} \oplus \mathbf{m}^*_{i'}[j'], K^{*mac}_{i',j'})$ (resp. sets that for all $(i,j) \in \mathcal{I}_S \bigcup \{(\widetilde{i},\widetilde{j}) \mid \widetilde{i} \in \mathcal{I}_R, \widetilde{j} \in [|\mathbf{m}^*_{\widetilde{i}}|]\}$, $\mathsf{H}_{\mathrm{RO}}(K^*_{i,j}) = (c^{*sym}_{i,j} \oplus \mathbf{m}^*_i[j], K^{*mac}_{i,j})$). Therefore, we obtain Eq. (53).

Game $\mathbf{G}_3$: Let BAD denote the event that $\mathcal{A}_2$ or $\mathcal{A}_3$ submits a decryption query $(i', (c^{kem}, c^{sym}, t))$ such that for $\overline{K} = \mathsf{Decap}(sk^{kem}_{i'}, c^{kem})$, there is some (i'', j'') satisfying that

(i) $i'' \in [n]$ and $j'' \in [|\mathbf{m}^*_{i''}|]$,
(ii) $\mathsf{H}_{\mathrm{RO}}(\overline{K})$ has not yet been programmed, and
(iii) $((i'', (c^{kem}, c^{sym}, t)) \notin \mathcal{C}) \wedge (\overline{K} = K^*_{i'',j''}) \wedge (\mathsf{Verf}(K^{*mac}_{i'',j''}, (pk_{i''}, c^{kem}, c^{sym}), t) = 1)$.

Note that without loss of generality, we have already assumed that after receiving $(sk_i, \mathbf{m}^*_i)_{i \in \mathcal{I}_R}$, the adversary will not query the decryption oracle on any $(\widetilde{i}, \cdot)$ satisfying $\widetilde{i} \in \mathcal{I}_R$. So if the decryption query $(i', (c^{kem}, c^{sym}, t))$ is made by $\mathcal{A}_3$, then $i' \notin \mathcal{I}_R$. Obviously game $\mathbf{G}_3$ is the same as $\mathbf{G}_2$, except that we abort this game (with output $\perp$) as long as BAD occurs.

Now we present the following lemma with a postponed proof.

Lemma 4. *There is a sUF-OT-CMA adversary $\mathcal{A}_{\mathsf{MAC}}$ attacking MAC, such that*

$$\Pr[\mathsf{BAD}] \leq n\widetilde{k}\mathtt{Adv}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}_{\mathsf{MAC}}}(\lambda),$$

where $\widetilde{k} = \max_{i=1}^{n} |\boldsymbol{m}^*_i|$.

Hence, for any PPT distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(\mathbf{G}_3)=1]-\Pr[\mathcal{D}(\mathbf{G}_2)=1]| \le \Pr[\mathsf{BAD}] \le n\widetilde{k}\mathtt{Adv}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}_{\mathsf{MAC}}}(\lambda). \quad (54)$$

Game $\mathbf{G}_4$: In this game, a new abort condition is added (as shown in Fig. 13). Specifically, if $\mathcal{A}_2$ or $\mathcal{A}_3$ submits a random-oracle query $K = K^*_{i',j'}$ for some $i' \in [n]$ and $j' \in [|\mathbf{m}^*_{i'}|]$ when $(K,\cdot) \notin L_{\mathsf{H}}$, then the challenger raises the event AbortH and aborts (with output $\perp$). Again, we present the following lemma with a postponed proof.

Lemma 5. *There is an OW-PCA adversary $\mathcal{A}_{\mathsf{KEM}}$ attacking KEM, such that for any PPT distinguisher $\mathcal{D}$,*

$$|\Pr[\mathcal{D}(\mathbf{G}_4)=1]-\Pr[\mathcal{D}(\mathbf{G}_3)=1]| \le \widetilde{k}\widetilde{n}\mathtt{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}_{\mathsf{KEM}}}(\lambda), \quad (55)$$

where $\widetilde{k} = \max_{i=1}^{n} |\boldsymbol{m}^*_i|$ and $\widetilde{n}$ is a polynomially upper bound of the number of receivers that adversary $\mathcal{A}$ creates.

Now, we construct a PPT simulator $\mathcal{S}$ for $\mathcal{A}$, as shown in Fig. 14. We also let $\mathcal{S}$ maintain a local array L_{H} and use it to keep track of issued calls (either by the game of $\mathcal{A}$). Obviously $\mathcal{S}$ simulates $\mathbf{G}_4$ perfectly for $\mathcal{A}$, so we derive that

$$\mathtt{Exp}^{\text{Bi-SO-ideal}}_{\mathsf{PKE}_{\text{K-M}},\mathcal{S}}(\lambda) = \mathbf{G}_4. \quad (56)$$

Therefore, combining Eq. (50)-(56), we derive that for any PPT distinguisher $\mathcal{D}$,

$$\begin{aligned}
&\mathtt{Adv}^{\text{SIM-Bi-SO-CCA}}_{\mathsf{PKE}_{\text{K-M}},\mathcal{A},\mathcal{S},\mathcal{D}}(\lambda)\\
&= \left|\Pr[\mathcal{D}(\mathtt{Exp}^{\text{Bi-SO-real}}_{\mathsf{PKE}_{\text{K-M}},\mathcal{A}}(\lambda))=1]-\Pr[\mathcal{D}(\mathtt{Exp}^{\text{Bi-SO-ideal}}_{\mathsf{PKE}_{\text{K-M}},\mathcal{S}}(\lambda))=1]\right|\\
&= |\Pr[\mathcal{D}(\mathbf{G}_{-1})=1]-\Pr[\mathcal{D}(\mathbf{G}_4)=1]|\\
&\le \frac{n}{2}\mathtt{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}_{\text{Coll}}}(\lambda) + \frac{n\widetilde{k}(n\widetilde{k}-1)}{2^{\lambda+1}} + \frac{n\widetilde{k}q_r}{2^{\lambda}-q_r} + \frac{n\widetilde{k}q_d}{2^{\lambda}-q_d}\\
&\qquad + n\widetilde{k}\mathtt{Adv}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}_{\mathsf{MAC}}}(\lambda) + n\widetilde{k}\mathtt{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}_{\mathsf{KEM}}}(\lambda)
\end{aligned}$$

for two suitable OW-PCA adversaries $\mathcal{A}_{\text{Coll}}, \mathcal{A}_{\mathsf{KEM}}$ and a suitable sUF-OT-CMA adversary $\mathcal{A}_{\mathsf{MAC}}$, where $\widetilde{k} = \max_{i=1}^{n} |\mathbf{m}^*_i|$.

We catch up with the proofs of Lemma 4 and Lemma 5.

Proof (of Lemma 4).

Based on $\mathcal{A}$, we construct a PPT adversary $\mathcal{A}_{\mathsf{MAC}}$ as shown in Fig. 15 and Fig. 16, where we use $\mathtt{VERF}$ to denote $\mathcal{A}_{\mathsf{MAC}}$'s verification oracle.

$\mathcal{S}_1^{\mathtt{SimMkRec}}(1^\lambda)$:
  $\mathsf{pp} = 1^\lambda \leftarrow \mathsf{Setup}(1^\lambda)$; $n := 0$; $\mathcal{C} := \emptyset$; $(\mathcal{M}, s_1) \leftarrow \mathcal{A}_1^{\mathtt{HASH}_1,\mathtt{MkRec},\mathtt{Dec}_1}(\mathsf{pp})$
  If $\exists\ i \neq i'$, s.t. $pk_i^{kem} = pk_{i'}^{kem}$: abort
  $\widehat{s}_1 := ((pk_i, sk_i)_{i\in[n]}, n, \mathcal{C}, L_{\mathsf{H}}, \mathcal{M}, s_1)$; Return $(\mathcal{M}, \widehat{s}_1)$

$\mathcal{S}_2(\mathsf{len} = ((|\mathbf{m}_i^*|, \ell)_{i\in[n]}), \widehat{s}_1)$:
  For $i = 1$ to n:
    For $j = 1$ to $|\mathbf{m}_i^*|$:
      $\mathbf{r}_i^*[j] \leftarrow \mathcal{R}_{\mathsf{Encap}}$; $(K_{i,j}^*, c_{i,j}^{*kem}) \leftarrow \mathsf{Encap}(pk_i^{kem}; \mathbf{r}_i^*[j])$
      If $(K_{i,j}^*, \cdot) \in L_{\mathsf{H}}$: AbortEARLY:=true; abort
  If $\exists\ (i', j') \neq (i'', j'')$, s.t. $K_{i',j'}^* = K_{i'',j''}^*$: abort
  For $i = 1$ to n:
    For $j = 1$ to $|\mathbf{m}_i^*|$:
      $(c_{i,j}^{*sym}, K_{i,j}^{*mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$; $t_{i,j}^* = \mathsf{Auth}(K_{i,j}^{*mac}, (pk_i^{kem}, c_{i,j}^{*kem}, c_{i,j}^{*sym}))$
      $\mathbf{c}_i^*[j] := (c_{i,j}^{*kem}, c_{i,j}^{*sym}, t_{i,j}^*)$; $\mathcal{C} = \mathcal{C} \cup \{(i, \mathbf{c}_i^*[j])\}$
  $(\mathcal{I}_S, \mathcal{I}_R, s_2) \leftarrow \mathcal{A}_2^{\mathtt{HASH}_2,\mathtt{Dec}_2}((\mathbf{c}_i^*)_{i\in[n]}, s_1)$
  $\widehat{s}_2 := ((pk_i, sk_i)_{i\in[n]}, (\mathbf{r}_i^*[j], \mathbf{c}_i^*[j], K_{i,j}^*, , K_{i,j}^{*mac})_{i\in[n], j\in[|\mathbf{m}_i^*|]}, n, \mathcal{C}, L_{\mathsf{H}}, \mathcal{M}, \mathcal{I}_S, \mathcal{I}_R, s_1)$
  Return $(\mathcal{I}_S, \mathcal{I}_R, \widehat{s}_2)$

$\mathcal{S}_3((\mathbf{m}_i^*[j])_{(i,j)\in\mathcal{I}_S}, (\mathbf{m}_i^*)_{i\in\mathcal{I}_R}, \widehat{s}_2)$:
  $(\mathsf{H}_{\mathrm{RO}}(K_{i,j}^*) = (c_{i,j}^{*sym} \oplus \mathbf{m}_i^*[j], K_{i,j}^{*mac}))_{(i,j)\in\mathcal{I}_S \bigcup \{(\widetilde{i},\widetilde{j})\,|\,\widetilde{i}\in\mathcal{I}_R, \widetilde{j}\in[|\mathbf{m}_{\widetilde{i}}^*|]\}}$
  $out \leftarrow \mathcal{A}_3^{\mathtt{HASH}_2,\mathtt{Dec}_2}((\mathbf{r}_i^*[j], \mathbf{m}_i^*[j])_{(i,j)\in\mathcal{I}_S}, (sk_i, \mathbf{m}_i^*)_{i\in\mathcal{I}_R}, s_2)$; Return out

$\underline{\mathtt{HASH}_1(K)}$:
  If $(K, \cdot) \notin L_{\mathsf{H}}$: $(K^{sym}, K^{mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$; $\mathsf{H}_{\mathrm{RO}}(K) := (K^{sym}, K^{mac})$
  Return $\mathsf{H}_{\mathrm{RO}}(K)$

$\underline{\mathtt{HASH}_2(K)}$:
  If $(K, \cdot) \notin L_{\mathsf{H}}$:
    If $K = K_{i',j'}^*$ for some $i' \in [n]$ and $j' \in [|\mathbf{m}_{i'}^*|]$: AbortH := true; abort
    Else: $(K^{sym}, K^{mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$; $\mathsf{H}_{\mathrm{RO}}(K) := (K^{sym}, K^{mac})$
  Return $\mathsf{H}_{\mathrm{RO}}(K)$

$\underline{\mathtt{MkRec}()}$:
  $\mathtt{SimMkRec}()$; $n = n + 1$; $(pk_n^{kem}, sk_n^{kem}) \leftarrow \mathsf{KemGen}(1^\lambda)$; $pk_n := pk_n^{kem}$; $sk_n := (pk_n^{kem}, sk_n^{kem})$
  Return pk_n

$\underline{\mathtt{Dec}_1(i', (c^{kem}, c^{sym}, t))}$:
  If $(i' > n) \vee ((i', (c^{kem}, c^{sym}, t)) \in \mathcal{C})$: return $\perp$
  $\overline{K} = \mathsf{Decap}(sk_{i'}^{kem}, c^{kem})$; $(\overline{K}^{sym}, \overline{K}^{mac}) = \mathtt{HASH}_1(\overline{K})$
  If $\mathsf{Verf}(\overline{K}^{mac}, (pk_{i'}^{kem}, c^{kem}, c^{sym}), t) = 0$: return $\perp$
  Return $\overline{m} = c^{sym} \oplus \overline{K}^{sym}$

$\underline{\mathtt{Dec}_2(i', (c^{kem}, c^{sym}, t))}$:
  If $(i' > n) \vee ((i', (c^{kem}, c^{sym}, t)) \in \mathcal{C})$: return $\perp$
  $\overline{K} = \mathsf{Decap}(sk_{i'}^{kem}, c^{kem})$
  If $(\overline{K} \in \{K_{i,j}^* \mid i \in [n], j \in [|\mathbf{m}_i^*|]\}) \wedge ((\overline{K}, \cdot) \notin L_{\mathsf{H}})$: return $\perp$
  $(\overline{K}^{sym}, \overline{K}^{mac}) = \mathtt{HASH}_2(\overline{K})$
  If $\mathsf{Verf}(\overline{K}^{mac}, (pk_{i'}^{kem}, c^{kem}, c^{sym}), t) = 0$: return $\perp$
  Return $\overline{m} = c^{sym} \oplus \overline{K}^{sym}$

Fig. 14 Simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ in the proof of Theorem 6.

Adversary $\mathcal{A}^{\mathtt{VERF}}_{\mathsf{MAC},1}(1^\lambda)$:

$(\widetilde{m^*}, \widetilde{t^*}) := (\perp, \perp)$; $\mathsf{pp} = 1^\lambda \leftarrow \mathsf{Setup}(1^\lambda)$; $n := 0$; $\mathcal{C} := \emptyset$; $(\mathcal{M}, s_1) \leftarrow \mathcal{A}_1^{\mathtt{HASH}_1, \mathtt{MkRec}, \mathtt{Dec}_1}(\mathsf{pp})$
If $\exists\ i \neq i'$, s.t. $pk_i^{kem} = pk_{i'}^{kem}$: abort
$\boldsymbol{M} := (\mathbf{m}_1^*, \cdots, \mathbf{m}_n^*) \leftarrow \mathcal{M}$; $\widetilde{i} \leftarrow [n]$; $\widetilde{j} \leftarrow [|\mathbf{m}_{\widetilde{i}}^*|]$
For $i = 1$ to n:
  For $j = 1$ to $|\mathbf{m}_i^*|$:
    $\mathbf{r}_i^*[j] \leftarrow \mathcal{R}_{\mathsf{Encap}}$; $(K_{i,j}^*, c_{i,j}^{*kem}) \leftarrow \mathsf{Encap}(pk_i^{kem}; \mathbf{r}_i^*[j])$
    If $(K_{i,j}^*, \cdot) \in L_{\mathsf{H}}$: AbortEARLY:=true; abort
If $\exists\ (i', j') \neq (i'', j'')$, s.t. $K_{i',j'}^* = K_{i'',j''}^*$: abort
$c_{\widetilde{i},\widetilde{j}}^{*sym} \leftarrow \{0,1\}^\ell$; $\widetilde{m} := (pk_{\widetilde{i}}^{kem}, c_{\widetilde{i},\widetilde{j}}^{*kem}, c_{\widetilde{i},\widetilde{j}}^{*sym})$
$\widetilde{s}_1 := ((\widetilde{m^*}, \widetilde{t^*}), (\widetilde{i}, \widetilde{j}), (pk_i, sk_i)_{i\in[n]}, (\mathbf{r}_i^*[j], K_{i,j}^*, c_{i,j}^{*kem})_{i\in[n],j\in[\ell]}, c_{\widetilde{i},\widetilde{j}}^{*sym}, n, \mathcal{C}, L_{\mathsf{H}}, \boldsymbol{M}, \mathcal{M}, s_1)$
Return $(\widetilde{m}, \widetilde{s}_1)$

Adversary $\mathcal{A}^{\mathtt{VERF}}_{\mathsf{MAC},2}(\widetilde{t}, \widetilde{s}_1)$:

$t_{\widetilde{i},\widetilde{j}}^* := \widetilde{t}$; $\mathbf{c}_{\widetilde{i}}^*[\widetilde{j}] := (c_{\widetilde{i},\widetilde{j}}^{*kem}, c_{\widetilde{i},\widetilde{j}}^{*sym}, t_{\widetilde{i},\widetilde{j}}^*)$; $\mathcal{C} = \mathcal{C} \cup \{(\widetilde{i}, \mathbf{c}_{\widetilde{i}}^*[\widetilde{j}])\}$
For $i \in [n] \setminus \{\widetilde{i}\}$:
  For $j = 1$ to $|\mathbf{m}_i^*|$:
    $(c_{i,j}^{*sym}, K_{i,j}^{*mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$; $t_{i,j}^* = \mathsf{Auth}(K_{i,j}^{*mac}, (pk_i^{kem}, c_{i,j}^{*kem}, c_{i,j}^{*sym}))$
    $\mathbf{c}_i^*[j] := (c_{i,j}^{*kem}, c_{i,j}^{*sym}, t_{i,j}^*)$; $\mathcal{C} = \mathcal{C} \cup \{(i, \mathbf{c}_i^*[j])\}$
For $j \in [|\mathbf{m}_{\widetilde{i}}^*|] \setminus \{\widetilde{j}\}$:
  $(c_{\widetilde{i},j}^{*sym}, K_{\widetilde{i},j}^{*mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$; $t_{\widetilde{i},j}^* = \mathsf{Auth}(K_{\widetilde{i},j}^{*mac}, (pk_{\widetilde{i}}^{kem}, c_{\widetilde{i},j}^{*kem}, c_{\widetilde{i},j}^{*sym}))$
  $\mathbf{c}_{\widetilde{i}}^*[j] := (c_{\widetilde{i},j}^{*kem}, c_{\widetilde{i},j}^{*sym}, t_{\widetilde{i},j}^*)$; $\mathcal{C} = \mathcal{C} \cup \{(\widetilde{i}, \mathbf{c}_{\widetilde{i}}^*[j])\}$
$(\mathcal{I}_S, \mathcal{I}_R, s_2) \leftarrow \mathcal{A}_2^{\mathtt{HASH}_2, \mathtt{Dec}_2}((c_{i,j}^*)_{i\in[n],j\in[k]}, s_1)$
If $((\widetilde{i}, \widetilde{j}) \in \mathcal{I}_S) \vee (\widetilde{i} \in \mathcal{I}_R)$: ABORT-Null := true
$(\mathsf{H}_{\mathrm{RO}}(K_{i,j}^*) = (c_{i,j}^{*sym} \oplus \mathbf{m}_i^*[j], K_{i,j}^{*mac}))_{(i,j)\in\mathcal{I}_S \bigcup \{(i'',j'')|i''\in\mathcal{I}_R, j''\in[|\mathbf{m}_{i''}^*|]\}}$
$out \leftarrow \mathcal{A}_3^{\mathtt{HASH}_2, \mathtt{Dec}_2}((\mathbf{r}_i^*[j], \mathbf{m}_i^*[j])_{(i,j)\in\mathcal{I}_S}, (sk_i, \mathbf{m}_i^*[j])_{i\in\mathcal{I}_R, j\in[|\mathbf{m}_i^*|]}, s_2)$
Return $(\widetilde{m^*}, \widetilde{t^*})$

$\mathtt{HASH}_1(K)$:

If $(K, \cdot) \notin L_{\mathsf{H}}$: $(K^{sym}, K^{mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$; $\mathsf{H}_{\mathrm{RO}}(K) := (K^{sym}, K^{mac})$
Return $\mathsf{H}_{\mathrm{RO}}(K)$

$\mathtt{HASH}_2(K)$:

If $(K, \cdot) \notin L_{\mathsf{H}}$:
  If $K = K_{i'',j''}^*$ for some $i'' \in [n]$ and $j'' \in [|\mathbf{m}_{i''}^*|]$:
    If $(i'', j'') \neq (\widetilde{i}, \widetilde{j})$: $\mathsf{H}_{\mathrm{RO}}(K_{i'',j''}^*) = (c_{i'',j''}^{*sym} \oplus \mathbf{m}_{i''}^*[j''], K_{i'',j''}^{*mac})$
    If $(i'', j'') = (\widetilde{i}, \widetilde{j})$: ABORT-Null := true
  Else:
    $(K^{sym}, K^{mac}) \leftarrow \{0,1\}^\ell \times \mathcal{K}_{\mathsf{MAC}}$; $\mathsf{H}_{\mathrm{RO}}(K) := (K^{sym}, K^{mac})$
Return $\mathsf{H}_{\mathrm{RO}}(K)$

$\mathtt{MkRec}()$:

$n = n + 1$; $(pk_n^{kem}, sk_n^{kem}) \leftarrow \mathsf{KemGen}(1^\lambda)$; $pk_n := pk_n^{kem}$; $sk_n := (pk_n^{kem}, sk_n^{kem})$
Return pk_n

(The procedures of the decryption oracles $\mathtt{Dec}_1$ and $\mathtt{Dec}_2$ are given in Fig. 16.)

Fig. 15 Adversary $\mathcal{A}_{\mathsf{MAC}} = (\mathcal{A}_{\mathsf{MAC},1}, \mathcal{A}_{\mathsf{MAC},2})$ attacking MAC.

For each $i \in [n]$ and each $j \in [|\mathbf{m}_i^*|]$, denote by $\mathsf{BAD}_{i,j}$ the event BAD which is caused by the query $(i', (c^{kem}, c^{sym}, t))$ satisfying $\mathsf{Decap}(sk_{i'}^{kem},$

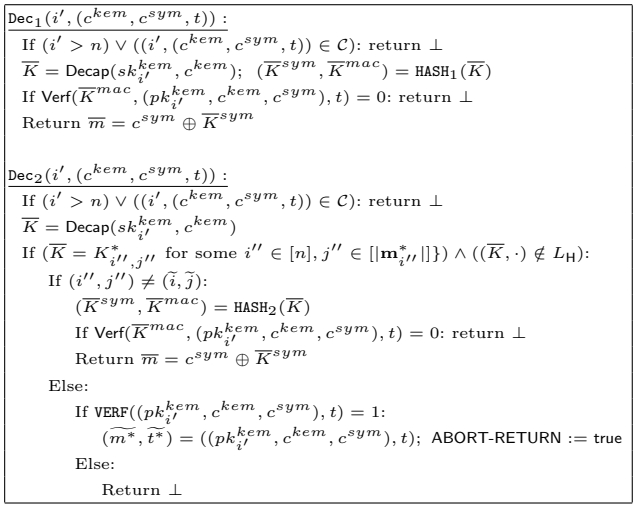

Fig. 16 The decryption oracles provided by $\mathcal{A}_{\mathsf{MAC}}$ in Fig. 15.

$c^{kem}) = K^*_{i,j}$. Thus we have $\Pr[\mathsf{BAD}] \leq \sum_{i=1}^{n} \sum_{j=1}^{|\mathbf{m}^*_i|} \Pr[\mathsf{BAD}_{i,j}]$. We also introduce two special events, ABORT-Null and ABORT-RETURN (as shown in Fig. 15 and Fig. 16), and require that when ABORT-Null (resp. ABORT-RETURN) is set true, $\mathcal{A}_{\mathsf{MAC}}$ immediately terminates the simulation and returns $(\perp, \perp)$ (resp. $(\widetilde{m^*}, \widetilde{t^*})$) as its final output.

Now we compute $\Pr[\mathsf{BAD}_{i,j}]$ for each $i \in [n]$ and each $j \in [|\mathbf{m}^*_i|]$.

Firstly, we note that if neither ABORT-Null nor ABORT-RETURN is set true, then $\mathcal{A}_{\mathsf{MAC}}$ perfectly simulates game $\mathbf{G}_2$ for $\mathcal{A}$.

Secondly, ABORT-RETURN is set true only if $(K^*_{\widetilde{i},\widetilde{j}}, \cdot) \notin L_{\mathsf{H}}$ (i.e., ABORT-RETURN is set true only if neither $\mathcal{A}$ nor the game has queried the random oracle on $K = K^*_{\widetilde{i},\widetilde{j}}$). Hence, the termination incurred by ABORT-Null will not influence the probability of ABORT-RETURN = true. In other words, if we introduce the same event ABORT-RETURN in $\mathbf{G}_2$, then $\Pr[\mathsf{ABORT\text{-}RETURN}]$ in $\mathbf{G}_2$ is the same as $\Pr[\mathsf{ABORT\text{-}RETURN}]$ in the game simulated by $\mathcal{A}_{\mathsf{MAC}}$.

Thirdly, note that for any fixed $i'' \in [n]$ and any fixed $j'' \in [|\mathbf{m}^*_{i''}|]$, $\mathsf{BAD}_{i'',j''}$ occurs if and only if ABORT-RETURN occurs *when* $(\widetilde{i}, \widetilde{j})$ *is fixed to be* (i'', j''). In other words,

$$\Pr[\mathsf{BAD}_{i'',j''}] = \Pr[\mathsf{ABORT\text{-}RETURN} \mid (\widetilde{i}, \widetilde{j}) = (i'', j'')].$$

When ABORT-RETURN is set true, $(\widetilde{m^*}, \widetilde{t^*}) = ((pk_{i'}^{kem}, c^{kem}, c^{sym}), t)$ and $\mathtt{VERF}(\widetilde{m^*}, \widetilde{t^*}) = 1$. If $i' \neq \widetilde{i}$, we derive that $pk_{i'}^{kem} \neq pk_{\widetilde{i}}^{kem}$ because of

the change introduced in $\mathbf{G}_0$. So

$$(\widetilde{m^*}, \widetilde{t^*}) = ((pk_{i'}^{kem}, c^{kem}, c^{sym}), t) \neq ((pk_{\widetilde{i}}^{kem}, c_{\widetilde{i},\widetilde{j}}^{*kem}, c_{\widetilde{i},\widetilde{j}}^{*sym}), t_{\widetilde{i},\widetilde{j}}^{*}) = (\widetilde{m}, \widetilde{t}).$$

If $i' = \widetilde{i}$, since $(i', (c^{kem}, c^{sym}, t)) \notin \mathcal{C}$ and $(\widetilde{i}, (c_{\widetilde{i},\widetilde{j}}^{*kem}, c_{\widetilde{i},\widetilde{j}}^{*sym}, t_{\widetilde{i},\widetilde{j}}^{*})) \in \mathcal{C}$, we derive that $(c^{kem}, c^{sym}, t) \neq (c_{\widetilde{i},\widetilde{j}}^{*kem}, c_{\widetilde{i},\widetilde{j}}^{*sym}, t_{\widetilde{i},\widetilde{j}}^{*})$, which also implies

$$(\widetilde{m^*}, \widetilde{t^*}) = ((pk_{i'}^{kem}, c^{kem}, c^{sym}), t) \neq ((pk_{\widetilde{i}}^{kem}, c_{\widetilde{i},\widetilde{j}}^{*kem}, c_{\widetilde{i},\widetilde{j}}^{*sym}), t_{\widetilde{i},\widetilde{j}}^{*}) = (\widetilde{m}, \widetilde{t}).$$

Therefore,

$$\mathtt{Adv}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}_{\mathsf{MAC}}}(\lambda) = \Pr[\mathsf{ABORT\text{-}RETURN}].$$

Hence, we obtain that

$$\begin{aligned}
&\Pr[\mathsf{BAD}] \\
&\leq \sum_{i=1}^{n} \sum_{j=1}^{|\mathbf{m}_i^*|} \Pr[\mathsf{BAD}_{i,j}] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{|\mathbf{m}_i^*|} \Pr[\mathsf{ABORT\text{-}RETURN} \mid (\widetilde{i}, \widetilde{j}) = (i,j)] \\
&\leq n\widetilde{k} \sum_{i=1}^{n} \sum_{j=1}^{|\mathbf{m}_i^*|} \Pr[\mathsf{ABORT\text{-}RETURN} \mid (\widetilde{i}, \widetilde{j}) = (i,j)] \Pr[(\widetilde{i}, \widetilde{j}) = (i,j)] \\
&\leq n\widetilde{k} \Pr[\mathsf{ABORT\text{-}RETURN}] \\
&= n\widetilde{k} \mathtt{Adv}^{\text{sUF-OT-CMA}}_{\mathsf{MAC},\mathcal{A}_{\mathsf{MAC}}}(\lambda),
\end{aligned}$$

where $\widetilde{k} = \max_{i=1}^{n} |\mathbf{m}_i^*|$. □

Proof (of Lemma 5). $\mathbf{G}_4$ and $\mathbf{G}_3$ are identical except that $\mathsf{AbortH} = \mathsf{true}$, i.e., for any PPT distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(\mathbf{G}_4) = 1] - \Pr[\mathcal{D}(\mathbf{G}_3) = 1]| \leq \Pr[\mathsf{AbortH}].$$

Assmue that $\widetilde{n}$ is a polynomially upper bound of the number of receivers that adversary $\mathcal{A}$ creates. We show an OW-PCA adversary $\mathcal{A}_{\mathsf{KEM}}$, attacking KEM, in Fig. 17 and Fig. 18. We introduce three special events $\mathsf{ABORT\text{-}RETURN}$, $\mathsf{AbortH}_{\text{fail}}$ and $\mathsf{ABORT\text{-}Null}$, and require that when any one of these three events is set true, $\mathcal{A}_{\mathsf{KEM}}$ immediately terminates the simulation and returns the current $\widetilde{K}$ as its final output.

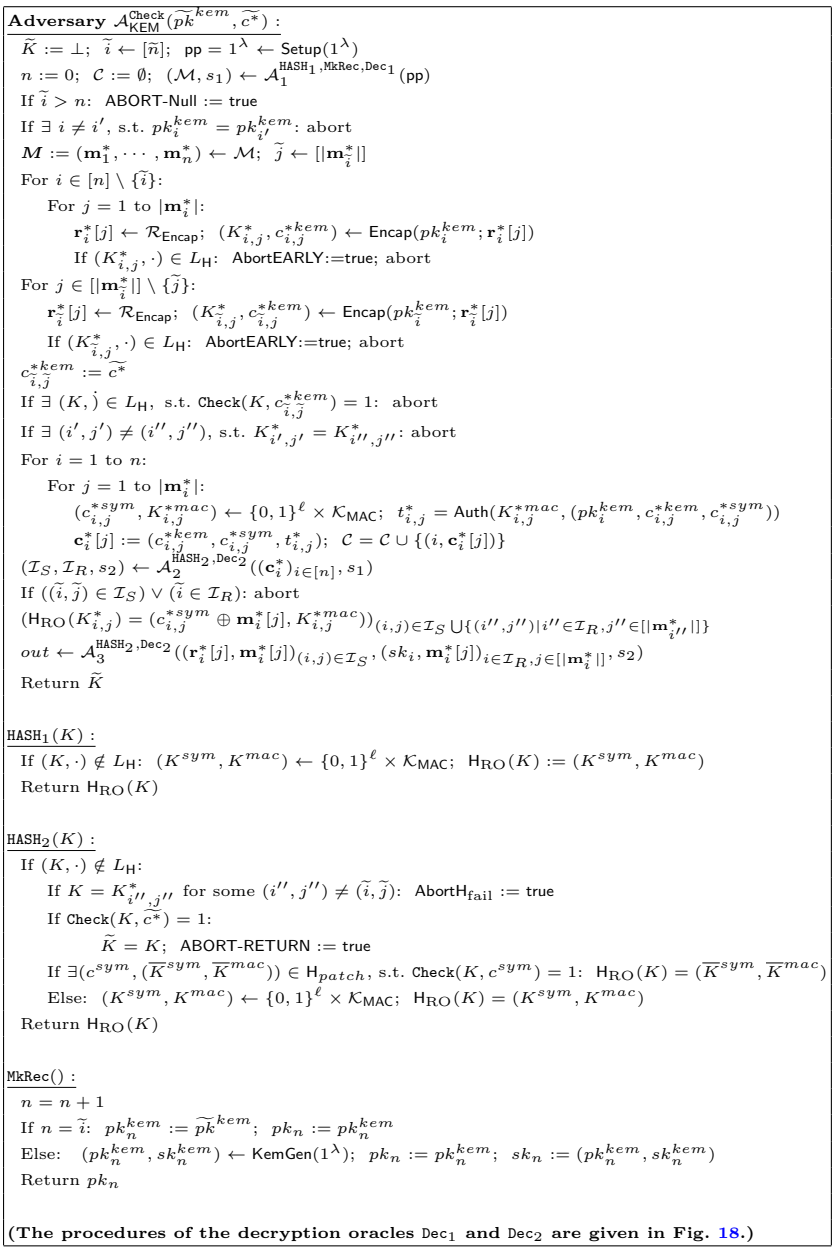

Fig. 17 Adversary $\mathcal{A}_{\mathsf{KEM}}$ attacking KEM.

Now we take a look at adversary $\mathcal{A}_{\mathsf{KEM}}$ in Fig. 17 and Fig. 18.

$\underline{\mathtt{Dec}_1(i', (c^{kem}, c^{sym}, t)) :}$
If $(i' > n) \vee ((i', (c^{kem}, c^{sym}, t)) \in \mathcal{C})$: return $\perp$
If $i' \neq \widetilde{i}$:
$\overline{K} = \mathsf{Decap}(sk_{i'}^{kem}, c^{kem})$; $(\overline{K}^{sym}, \overline{K}^{mac}) = \mathtt{HASH}_1(\overline{K})$
If $i' = \widetilde{i}$:
If $\exists (K, \cdot) \in L_{\mathsf{H}}$, s.t. $\mathtt{Check}(K, c^{kem}) = 1$:
$\overline{K} = K$; $(\overline{K}^{sym}, \overline{K}^{mac}) = \mathtt{HASH}_1(\overline{K})$
Else:
$(\overline{K}^{sym}, \overline{K}^{mac}) \leftarrow \{0,1\}^{\ell} \times \mathcal{K}_{\mathsf{MAC}}$; Add $(c^{kem}, (\overline{K}^{sym}, \overline{K}^{mac}))$ to H_{patch}
If $\mathsf{Verf}(\overline{K}^{mac}, (pk_{i'}^{kem}, c^{kem}, c^{sym}), t) = 0$: return $\perp$
Return $\overline{m} = c^{sym} \oplus \overline{K}^{sym}$

$\underline{\mathtt{Dec}_2(i', (c^{kem}, c^{sym}, t)) :}$
If $((i', (c^{kem}, c^{sym}, t)) \in \mathcal{C}) \vee (i' \in \mathcal{I}'_R)$: return $\perp$
If $i' \neq \widetilde{i}$:
$\overline{K} = \mathsf{Decap}(sk_{i'}^{kem}, c^{kem})$
If $(\overline{K} \in \{K^*_{i,j} \mid i \in [n], j \in [|\mathbf{m}_i^*|], (i,j) \neq (\widetilde{i}, \widetilde{j})\}) \wedge ((\overline{K}, \cdot) \notin L_{\mathsf{H}})$: return $\perp$
If $((\mathtt{Check}(\overline{K}, \widetilde{c^*}) = 1) \wedge ((\overline{K}, \cdot) \notin L_{\mathsf{H}})$: return $\perp$
$(\overline{K}^{sym}, \overline{K}^{mac}) = \mathtt{HASH}_2(\overline{K})$
If $\mathsf{Verf}(\overline{K}^{mac}, (pk_{i'}^{kem}, c^{kem}, c^{sym}), t) = 0$: return $\perp$
Return $\overline{m} = c^{sym} \oplus \overline{K}^{sym}$
If $i' = \widetilde{i}$:
If $\exists\ K \in \{K^*_{i,j} \mid i \in [n], j \in [|\mathbf{m}_i^*|], (i,j) \neq (\widetilde{i}, \widetilde{j})\}$ s.t. $(\mathtt{Check}(K, c^{kem}) = 1) \wedge ((\overline{K}, \cdot) \notin L_{\mathsf{H}})$:
Return $\perp$
If $(c^{kem} = \widetilde{c^*}) \wedge (\forall (K, \cdot) \in L_{\mathsf{H}},\ \mathtt{Check}(K, c^{kem}) = 0)$: return $\perp$
If $\exists (K, \cdot) \in L_{\mathsf{H}}$, s.t. $\mathtt{Check}(K, c^{kem}) = 1$:
$\overline{K} = K$; $(\overline{K}^{sym}, \overline{K}^{mac}) = \mathtt{HASH}_2(\overline{K})$
If $\mathsf{Verf}(\overline{K}^{mac}, (pk_{i'}^{kem}, c^{kem}, c^{sym}), t) = 0$: return $\perp$
Return $\overline{m} = c^{sym} \oplus \overline{K}^{sym}$
Else:
$(\overline{K}^{sym}, \overline{K}^{mac}) \leftarrow \{0,1\}^{\ell} \times \mathcal{K}_{\mathsf{MAC}}$; Add $(c^{kem}, (\overline{K}^{sym}, \overline{K}^{mac}))$ to H_{patch}
If $\mathsf{Verf}(\overline{K}^{mac}, (pk_{i'}^{kem}, c^{kem}, c^{sym}), t) = 0$: return $\perp$
Return $\overline{m} = c^{sym} \oplus \overline{K}^{sym}$

Fig. 18 The decryption oracles provided by $\mathcal{A}_{\mathsf{KEM}}$ in Fig. 17.

Upon receiving $(\widetilde{pk}^{kem}, \widetilde{c^*})$, where $\widetilde{c^*}$ is an encapsulation of some key $\widetilde{K^*}$ that $\mathcal{A}_{\mathsf{KEM}}$ aims to find out, $\mathcal{A}_{\mathsf{KEM}}$ runs $\mathcal{A}$ as $\mathcal{A}$ is run in $\mathbf{G}_4$, except for the following differences:

(1) At the beginning, $\mathcal{A}_{\mathsf{KEM}}$ samples a random $\widetilde{i} \leftarrow [\widetilde{n}]$. If $\widetilde{i} > n$, then ABORT-Null is set true. Obviously,

$$\Pr[\neg\mathsf{ABORT\text{-}Null}] = \Pr[\widetilde{i} \leq n] = \frac{n}{\widetilde{n}}. \tag{57}$$

From now on, when ABORT-Null $\neq$ true, we write that $\mathcal{I}_{\neq(\widetilde{i},\widetilde{j})} := \{(i'', j'') \mid i'' \in [n], j'' \in [|\mathbf{m}^*_{i''}|], (i'', j'') \neq (\widetilde{i}, \widetilde{j})\}$.

(2) Note that when ABORT-Null $\neq$ true, $\mathcal{A}_{\mathsf{KEM}}$ successfully sets that $pk_{\widetilde{i}} = \widetilde{pk}^{kem}$ and $c^{*kem}_{\widetilde{i},\widetilde{j}} := \widetilde{c^*}$, where $\widetilde{j} \leftarrow [|\mathbf{m}^*_{\widetilde{i}}|]$.

(3) When $\mathsf{ABORT\text{-}Null} \neq \mathsf{true}$, since $\mathcal{A}_{\mathsf{KEM}}$ knows all the $(K^*_{i,j})_{(i,j)\in\mathcal{I}_{\neq(\widetilde{i},\widetilde{j})}}$ and can access to the $\mathtt{Check}(\cdot,\cdot)$ oracle, it can check by itself whether there exist $(i',j') \neq (i'',j'')$ such that $K^*_{i',j'} = K^*_{i'',j''}$, as in $\mathbf{G}_4$.

(4) When $\mathsf{ABORT\text{-}Null} \neq \mathsf{true}$, if $\mathcal{A}$ submits a selective opening query $(\mathcal{I}_S, \mathcal{I}_R)$ such that $(\widetilde{i},\widetilde{j}) \in \mathcal{I}_S$ or $\widetilde{i} \in \mathcal{I}_R$, then $\mathcal{A}_{\mathsf{KEM}}$ apparently has guessed $(\widetilde{i},\widetilde{j})$ wrong and aborts the simulation immediately. That's because when $(\widetilde{i},\widetilde{j}) \in \mathcal{I}_S$ or $\widetilde{i} \in \mathcal{I}_R$, AbortH will not happen anymore. So this termination will not affect the probability that AbortH occurs.

(5) If $\mathcal{A}_1$ submits a decryption query $(i', (c^{kem}, c^{sym}, t))$ to $\mathtt{Dec}_1$:
- If $i' \neq \widetilde{i}$, $\mathcal{A}_{\mathsf{KEM}}$ can compute $\overline{K} = \mathsf{Decap}(sk^{kem}_{i'}, c^{kem})$ by itself. Again, since $\mathcal{A}_{\mathsf{KEM}}$ knows all the $(K^*_{i,j})_{(i,j)\in\mathcal{I}_{\neq(\widetilde{i},\widetilde{j})}}$ and can access to the $\mathtt{Check}(\cdot,\cdot)$ oracle, it can check whether $\overline{K} \in \{K^*_{i,j} \mid i \in [n], j \in [|\mathbf{m}^*_i|]\}$ and whether $(\overline{K},\cdot) \notin L_{\mathsf{H}}$. So it can answer this decryption query directly by itself.
- If $i' = \widetilde{i}$, $\mathcal{A}_{\mathsf{KEM}}$ cannot compute $\overline{K} = \mathsf{Decap}(sk^{kem}_{i'}, c^{kem})$ since it does not have $sk^{kem}_{i'} = sk^{kem}_{\widetilde{i}}$. In order to obtain $(\overline{K}^{sym}, \overline{K}^{mac})$, $\mathcal{A}_{\mathsf{KEM}}$ firstly checks whether "$\exists\ (K,\cdot) \in L_{\mathsf{H}}$, s.t. $\mathtt{Check}(K, c^{kem}) = 1$" or not. If so, then $\overline{K} = K$, thus $\mathcal{A}_{\mathsf{KEM}}$ can finish the remaining decryption procedures as in $\mathbf{G}_4$. If not, we make use of the "oracle patching technique" from [9]. Specifically, $\mathcal{A}_{\mathsf{KEM}}$ samples $(\overline{K}^{sym}, \overline{K}^{mac}) \leftarrow \{0,1\}^{\ell} \times \mathcal{K}_{\mathsf{MAC}}$, uses the keys to answer the decryption query, and meanwhile maintains a dedicated list H_{patch}, adding $(c^{kem}, (\overline{K}^{sym}, \overline{K}^{mac}))$ to H_{patch}. Then $\mathcal{A}_{\mathsf{KEM}}$ can finish the remaining decryption procedures.

(6) When $\mathsf{ABORT\text{-}Null} \neq \mathsf{true}$, if $\mathcal{A}_2$ or $\mathcal{A}_3$ submits a decryption query $(i', (c^{kem}, c^{sym}, t))$ to $\mathtt{Dec}_2$:
- If $i' \neq \widetilde{i}$, $\mathcal{A}_{\mathsf{KEM}}$ can compute $\overline{K} = \mathsf{Decap}(sk^{kem}_{i'}, c^{kem})$ by itself. Again, since $\mathcal{A}_{\mathsf{KEM}}$ knows all the $(K^*_{i,j})_{(i,j)\in\mathcal{I}_{\neq(\widetilde{i},\widetilde{j})}}$ and can access to the $\mathtt{Check}(\cdot,\cdot)$ oracle, it can check whether $\overline{K} \in \{K^*_{i,j} \mid i \in [n], j \in [|\mathbf{m}^*_i|]\}$ and whether $(\overline{K},\cdot) \notin L_{\mathsf{H}}$. So it can answer this decryption query directly by itself.
- If $i' = \widetilde{i}$, $\mathcal{A}_{\mathsf{KEM}}$ needs to check whether "$(\overline{K} \in \{K^*_{i,j} \mid i \in [n], j \in [|\mathbf{m}^*_i|]\}) \wedge ((\overline{K},\cdot) \notin L_{\mathsf{H}})$" without computing $\overline{K} = \mathsf{Decap}(sk^{kem}_{\widetilde{i}}, c^{kem})$. We notice that $\mathcal{A}_{\mathsf{KEM}}$ knows all the $(K^*_{i,j})_{(i,j)\in\mathcal{I}_{\neq(\widetilde{i},\widetilde{j})}}$ and can access to the $\mathtt{Check}(\cdot,\cdot)$ oracle, so it can check if "$(\overline{K} \in \{K^*_{i,j} \mid (i,j) \in \mathcal{I}_{\neq(\widetilde{i},\widetilde{j})}\}) \wedge ((\overline{K},\cdot) \notin L_{\mathsf{H}})$" by checking if "$\exists\ K \in \{K^*_{i,j} \mid i \in [n], j \in [|\mathbf{m}^*_i|], (i,j) \neq (\widetilde{i},\widetilde{j})\}$ s.t. $(\mathtt{Check}(K, c^{kem}) = 1) \wedge ((\overline{K},\cdot) \notin L_{\mathsf{H}})$". On the other hand, since KEM has unique encapsulations,

$\mathcal{A}_{\mathsf{KEM}}$ can check if "$(\overline{K} = K^*_{\widetilde{i},\widetilde{j}}) \wedge ((\overline{K}, \cdot) \notin L_{\mathsf{H}})$" by checking if "$(c^{kem} = \widetilde{c^*}) \wedge (\forall (K, \cdot) \in L_{\mathsf{H}},\ \mathtt{Check}(K, c^{kem}) = 0)$". Therefore, $\mathcal{A}_{\mathsf{KEM}}$ can finish the procedures "if $(\overline{K} \in \{K^*_{i,j} \mid i \in [n], j \in [|\mathbf{m}^*_i|]\}) \wedge ((\overline{K}, \cdot) \notin L_{\mathsf{H}})$" when $i' = \widetilde{i}$ in $\mathbf{G}_4$.
Next, in order to obtain $(\overline{K}^{sym}, \overline{K}^{mac})$, $\mathcal{A}_{\mathsf{KEM}}$ firstly checks whether "$\exists\, (K, \cdot) \in L_{\mathsf{H}}$, s.t. $\mathtt{Check}(K, c^{kem}) = 1$" or not. If so, then $\overline{K} = K$, thus $\mathcal{A}_{\mathsf{KEM}}$ can finish the remaining decryption procedures as in $\mathbf{G}_4$. If not, we make use of the "oracle patching technique" from [9]. $\mathcal{A}_{\mathsf{KEM}}$ samples $(\overline{K}^{sym}, \overline{K}^{mac}) \leftarrow \{0,1\}^{\ell} \times \mathcal{K}_{\mathsf{MAC}}$, uses the keys to answer the decryption query, and meanwhile maintains a dedicated list H_{patch}, adding $(c^{kem}, (\overline{K}^{sym}, \overline{K}^{mac}))$ to H_{patch}. Then $\mathcal{A}_{\mathsf{KEM}}$ can finish the remaining decryption procedures
In order to keep the responses to the random-oracle queries and to the decryption-oracle queries consistent, on each random-oracle query K (to $\mathtt{HASH}_2(\cdot)$) satisfying $(K, \cdot) \notin L_{\mathsf{H}}$, $\mathcal{A}_{\mathsf{KEM}}$ checks whether there is an entry $(\widehat{c}^{kem}, (\widehat{K}^{sym}, \widehat{K}^{mac})) \in \mathsf{H}_{patch}$ such that $\mathtt{Check}(K, \widehat{c}^{kem}) = 1$: if there is such an entry, then $\mathcal{A}_{\mathsf{KEM}}$ sets $\mathsf{H}_{\mathrm{RO}}(K) = (\widehat{K}^{sym}, \widehat{K}^{mac})$; otherwise, $\mathcal{A}_{\mathsf{KEM}}$ generates $\mathsf{H}_{\mathrm{RO}}(K)$ as before.

Note that when assuming $\mathsf{ABORT\text{-}Null} \neq \mathsf{true}$, the probability that AbortH occurs in the experiment simulated by $\mathcal{A}_{\mathsf{KEM}}$ for $\mathcal{A}$ (denoted as $\mathbf{G}_{\mathrm{sim}}$) is the same as that in $\mathbf{G}_4$, and that

$$\begin{aligned}
\mathtt{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}_{\mathsf{KEM}}}(\lambda) &= \Pr[\mathsf{ABORT\text{-}RETURN}] \\
&\geq \frac{1}{n\widetilde{k}} \Pr[\mathsf{AbortH} \mid \text{in } \mathbf{G}_{\mathrm{sim}}] \\
&= \frac{1}{n\widetilde{k}} \frac{n}{\widetilde{n}} \Pr[\mathsf{AbortH} \mid \text{in } \mathbf{G}_4] \\
&= \frac{1}{\widetilde{k}\widetilde{n}} \Pr[\mathsf{AbortH}],
\end{aligned}$$

where $\widetilde{k} = \max_{i=1}^{n} |\mathbf{m}^*_i|$. Therefore, for any PPT distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(\mathbf{G}_4) = 1] - \Pr[\mathcal{D}(\mathbf{G}_3) = 1]| \leq \Pr[\mathsf{AbortH}] \leq \widetilde{k}\widetilde{n}\mathtt{Adv}^{\text{OW-PCA}}_{\mathsf{KEM},\mathcal{A}_{\mathsf{KEM}}}(\lambda).$$

□
□