# Anonymous Public Key Encryption under Corruptions*

Zhengan Huang[1], Junzuo Lai[2], Shuai Han[3], Lin Lyu[4], and Jian Weng[2]

[1] Peng Cheng Laboratory, Shenzhen, China
zhahuang.sjtu@gmail.com
[2] College of Information Science and Technology, Jinan University, Guangzhou, China
{laijunzuo,cryptjweng}@gmail.com
[3] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
dalen17@sjtu.edu.cn
[4] Bergische Universität Wuppertal, Wuppertal, Germany
lin.lyu@uni-wuppertal.de

**Abstract.** Anonymity of public key encryption (PKE) requires that, in a multi-user scenario, the PKE ciphertexts do not leak information about which public keys are used to generate them. Corruptions are common threats in the multi-user scenario but anonymity of PKE under corruptions is less studied in the literature. In TCC 2020, Benhamouda et al. first provide a formal characterization for anonymity of PKE under a specific type of corruption. However, no known PKE scheme is proved to meet their characterization.

To the best of our knowledge, all the PKE application scenarios which require anonymity also require confidentiality. However, in the work by Benhamouda et al., different types of corruptions for anonymity and confidentiality are considered, which can cause security pitfalls. What's worse, we are not aware of any PKE scheme which can provide both anonymity and confidentiality under the same types of corruptions.

In this work, we introduce a new security notion for PKE called ANON-$\text{RSO}_k$&C security, capturing anonymity under corruptions. We also introduce SIM-$\text{RSO}_k$&C security which captures confidentiality under the same types of corruptions. We provide a generic framework of constructing PKE scheme which can achieve the above two security goals simultaneously based on a new primitive called key and message non-committing encryption (KM-NCE). Then we give a general construction of KM-NCE utilizing a variant of hash proof system (HPS) called Key-Openable HPS. We also provide Key-Openable HPS instantiations based on the matrix decisional Diffie-Hellman assumption. Therefore, we can obtain various concrete PKE instantiations achieving the two security goals in the standard model with *compact* ciphertexts. Furthermore, for some PKE instantiation, its security reduction is *tight*.

* A preliminary version of this paper appears in ASIACRYPT 2022. This is the full version.

## 1 Introduction

**Anonymity of PKE under corruptions.** The (single-user) IND-CCA security has been the de facto standard security for public-key encryption (PKE) schemes and is the target security of NIST PKE standardization for the next decades. It provides message confidentiality under CCA attacks. Meanwhile, *anonymity* is another security requirement for PKE and is not provided by the IND-CCA security. Roughly speaking, anonymity of PKE requires that, in a multi-user scenario, the PKE ciphertexts do not leak information about which public keys are used to generate them. The IK-CPA/CCA security given by Bellare et al. [BBDP01] is the first formalization of anonymity of PKE.

In such multi-user scenarios, multiple key pairs are generated, potentially correlated plaintexts are encrypted and sent to many receivers. Both the secret keys and the encrypted messages could be leaked due to accidents and/or adversarial attacks, which affects both the confidentiality and the anonymity of the PKE scheme. Researchers capture such threats by formalizing different types of *corruptions* in different multi-user scenarios. Many efforts have been made to establish confidentiality under corruptions and the study to selective-opening attacks are such examples.

However, anonymity of PKE under corruptions is much less studied. To the best of our knowledge, it is not considered until recently by Benhamouda et al. [BGG+20] in TCC 2020. They propose anonymity against selective-opening for PKE which is the first (and, to the best of our knowledge, also the only) formal definition of anonymity for PKE under corruptions. We will call this security as ANON-COR (anonymity under corruptions) security in this work. The ANON-COR security defined in [BGG+20] is as follows. Given $n$ public keys of $n$ users, an adversary submits $t$ messages of its choice, and then receives $t$ challenge ciphertexts, which are encryptions of the $t$ messages under $t$ distinct random public keys out of the $n$ user public keys. Next, the adversary can adaptively corrupt $Q < n$ users one at a time, obtaining their secret keys. (We will call such kind of corruption as *post-challenge user corruption*.) ANON-COR security requires that no feasible adversary can corrupt more than $\frac{Q}{n} + \epsilon$ (for some constant $\epsilon > 0$) fraction of the ciphertext-encrypting keys with non-negligible probability.

Unfortunately, no known PKE scheme is proved to have ANON-COR security. Actually, Benhamouda et al. [BGG+20] only prove that their suggested PKE scheme achieves a simplified version of ANON-COR security (where the adversary is restricted to corrupt some users at once) and conjecture that it also achieves the ANON-COR security. They leave constructing an ANON-COR secure PKE scheme as an interesting problem.

Furthermore, we think the ANON-COR security is restricted in the following sense.

- Non-adaptive. The ANON-COR security is non-adaptive in the sense that the adversary is *not* allowed to obtain any user secret key *before* seeing the challenge ciphertexts. This restricts its application scenario since, in the real-world, some users may be fully controlled by the adversary from the very beginning and the adversary may corrupt other users at any time.

– Single-challenge. The ANON-COR security considers a single-challenge setting where each public key is used *only once* to encrypt a single challenge message. This restriction limits its application scenario since, in practice, each public key is often used multiple times (for example, the application scenario in [BGG+20][1]).

Thus, we raise the following research question.

> *Q1: For PKE schemes, can we provide an achievable security formalization which provides anonymity under more adaptive corruptions in the multi-challenge setting?*

**Anonymity and confidentiality under the same types of corruptions.** We are not aware of any application scenario which only requires anonymity but not confidentiality of PKE schemes[2]. To the best of our knowledge, all the PKE application scenarios in the real world which require anonymity also require confidentiality. As an example, Benhamouda et al. [BGG+20] consider a blockchain application scenario which requires both of the two security guarantees. However, Benhamouda et al. capture the two security guarantees under *different* types of corruptions. More precisely, as shown in [BGG+20, Section 2.6], the scheme $\mathcal{E}_1$ requires both anonymity under post-challenge user corruption (ANON-COR security) and confidentiality under *the receiver selective opening (RSO)* corruption.

Although the ANON-COR security is called "anonymous against selective-opening" in [BGG+20], we want to note that the post-challenge user corruption considered in ANON-COR security is different from the RSO corruption considered for confidentiality. The RSO corruption [BDWY12, HPW15] considers an adversary, after seeing many challenge ciphertexts for different receivers (together with their public keys), is able to open a subset of the challenge ciphertexts (via corrupting a subset of the receivers to obtain their secret keys and received messages). However, the ANON-COR adversary is not able to specify some challenge ciphertexts and open them.

When the two security guarantees (anonymity and confidentiality) are both required, it is more desirable to capture them under the *same* types of corruptions. Taking [BGG+20] as an example, where anonymity and confidentiality are both required for the scheme $\mathcal{E}_1$ in [BGG+20], it does not make sense for

[1] In the Committee-Selection phase of the evolving-committee proactive secret sharing scheme considered in [BGG+20], some users are selected as committee members. Each committee member will encrypt one fresh secret key using its long term public key ($\mathsf{ct} \leftarrow \mathcal{E}_1.\mathsf{Enc}_{\mathsf{pk}}(\mathsf{esk})$). Since the same user may be selected as a committee member multiple times, the user's public key may be used multiple times to encrypt multiple messages.

[2] Actually, it does not make sense to *only* consider the anonymity of some PKE without considering its confidentiality. If confidentiality can be sacrificed, one can trivially achieve anonymity by assigning the identity map as the encryption and decryption algorithm, so that the ciphertext equals the message and is independent of any public key.

the adversary to attack anonymity *only* using the post-challenge user corruption and attack confidentiality *only* using the RSO corruption. Actually, there is no anonymity guarantee under the RSO corruption and no confidentiality guarantee under the post-challenge user corruption. This implies that, when the adversary is able to use both post-challenge user corruption and RSO corruption, it is possible that neither anonymity nor confidentiality holds for the PKE scheme. Consequently, when the two security guarantees are required under corruptions, they should be captured under the same types of corruptions.

Unfortunately, we are not aware of any PKE schemes which can provide the two security guarantees under the same types of corruptions. Thus, we raise our second research question.

> *Q2: Can we construct a PKE scheme which provides both anonymity and confidentiality under the same types of corruptions?*

We answer the above two research questions affirmatively in this work.

**Our contributions.** In this work:

- We formalize the notion of *ANONymity under Receiver Selective Opening attacks (in the $k$-challenge setting), adaptive user Corruptions and Chosen Plaintext / Ciphertext Attacks*, which we call ANON-RSO$_k$&C-CPA/CCA security for short. To capture confidentiality under the same types of corruptions, we also formalize the notion of SIM-RSO$_k$&C-CPA/CCA security.
- We provide a generic framework of constructing PKE schemes, achieving both ANON-RSO$_k$&C-CCA security and SIM-RSO$_k$&C-CCA security (we denote them as AC-RSO$_k$&C-CCA security for simplicity), based on a new primitive called *key and message non-committing encryption* (KM-NCE).
- We give a general construction of KM-NCE utilizing a variant of hash proof system (HPS) [CS02] which we call *Key-Openable HPS*.
- Finally, we provide Key-Openable HPS instantiations from the matrix decisional Diffie-Hellman (MDDH) assumption [EHK+13].

When plugging the HPS instantiations into the general construction framework, we can obtain an AC-RSO$_k$&C-CCA secure PKE scheme in the standard model which provides anonymity and confidentiality simultaneously under both adaptive user corruptions and RSO corruptions. Moreover, our scheme enjoys the properties that 1) the ciphertext is *compact* (i.e., ciphertext overhead[3] is the size of a constant number of group elements [HJR16], or more generally, is independent of the message length [HKM+18]), and 2) the security reduction is *tight*[4]. To the best of our knowledge, our scheme is the first PKE scheme achieving anonymity under adaptive corruptions (which is stronger than the ANON-COR security), thus solving the problem raised by Benhamouda et al.

[3] Ciphertext overhead means the ciphertext bitlength minus plaintext bitlength [HJR16].

[4] Tight reduction means that the security loss of the reduction is independent of the number of users, the number of challenges and the number of queries raised by the adversary.

[BGG+20] in TCC 2020. Also, our scheme is the first PKE scheme achieving $\mathrm{RSO}_k$-CCA security in the standard model with compact ciphertexts and tight security.

*AC-$\mathrm{RSO}_k$&C security derived from KM-NCE.* We take the approach of non-committing encryption [CFGN96, CHK05, HKM+18] to achieve AC-$\mathrm{RSO}_k$&C security. We introduce a new primitive called *key and message non-committing encryption* (KM-NCE), which is some kind of "message & public key-non-committing" encryption. Informally, KM-NCE allows one to generate fake ciphertexts via a fake encryption algorithm, and enables one to open $k$ fake ciphertexts to any $k$ messages under any public key (by showing an appropriate secret key) via an opening algorithm.

We formalize two security properties for KM-NCE. One is a single-user and $k$-challenge security notion called $\mathrm{KMNC}_k$-CPA/CCA security (c.f., Definition 4), and the other is robustness (c.f., Definition 5). Intuitively, $\mathrm{KMNC}_k$-CPA/CCA security requires that the real secret key together with $k$ real ciphertexts (encrypting $k$ messages chosen by the adversary) should be computationally indistinguishable from the opened secret key and $k$ fake ciphertexts.

KM-NCE serves as our core technical tool, and we show that $\mathrm{KMNC}_k$-CPA/CCA secure and robust KM-NCE implies AC-$\mathrm{RSO}_k$&C-CPA/CCA secure PKE. Due to the relative simplicity of $\mathrm{KMNC}_k$-CPA/CCA security (single-user, no simulator) in comparison to AC-$\mathrm{RSO}_k$&C-CPA/CCA security (multi-user, simulation-based), it is easier and conceptually simpler to construct KM-NCE and prove its security first than constructing AC-$\mathrm{RSO}_k$&C-CPA/CCA secure PKE directly.

*Generic construction of KM-NCE.* To construct KM-NCE, we propose a new building block called *Key-Openable HPS*, by equipping Hash Proof System (HPS) [CS02] with a hashing key opening algorithm $\mathsf{HOpen}_k$. Informally, given $k$ instances, $k$ hash values and the random coins used to sample them, a projection key (public key of HPS), and a corresponding hashing key (secret key of HPS) as the input, $\mathsf{HOpen}_k$ can output another hashing key such that 1) the outputted hashing key corresponds to the same projection key and 2) the given $k$ hash values are exactly hash values of the $k$ instances under the outputted hashing key. We also define some new properties for the key-openable HPS, including $openability_k$ (c.f., Definition 9) and $universality_{k+1}$ (c.f., Definition 10). By using key-openable HPS as an essential building block, we present a generic construction of $\mathrm{KMNC}_k$-CCA secure KM-NCE.

*Instantiations.* For concrete instantiations, we provide key-openable HPS instantiations based on the MDDH assumption. Due to the good versatility of the MDDH assumption, we can obtain various concrete instantiations of KM-NCE. Plugging the concrete instantiations into our general framework, we obtain AC-$\mathrm{RSO}_k$&C-CCA secure PKE schemes with *compact* ciphertexts in the standard model. For some concrete PKE instantiation, we can even *tightly* prove its AC-$\mathrm{RSO}_k$&C-CCA security.

**Related works.** The anonymity of PKE is first formalized by Bellare et al. [BBDP01] and they call it "key-privacy". Many follow up works continue research in this direction, such as [HT05, ABN10, Moh10]. Anonymity for PKE under corruptions is firstly considered by Benhamouda et al. [BGG+20].

The IND-CCA security in the multi-user setting with adaptive user corruptions except challenge is given in [BS20, LLP20]. Lee et al. [LLP20] propose the first PKE scheme in the random oracle model with tight IND-CCA security reduction in the multi-user setting with adaptive user corruptions except challenge.

In the research area of receiver selective opening (RSO) corruption for PKE, Bellare et al. [BDWY12] point out that IND-CPA security does not imply SIM-RSO-CPA security. Hazay et al. [HPW15] show that RSO security can be achieved from variants of non-committing encryption. Subsequent works [JLL16, JLL17, HKM+18, HLC+19] consider CCA security in the RSO setting and provide PKE schemes with RSO-CCA security. Yang et al. [YLH+20] consider RSO-CCA security in the multi-challenge setting. SIM-RSO-CCA secure PKE schemes with compact ciphertexts are proposed by Hara et al. [HKM+18] and Huang et al. [HLC+19].

## 2 Preliminaries

We assume that the security parameter $\lambda$ is an (implicit) input to all algorithms. For any positive integer $n$, we use $[n]$ to denote the set $\{1, \cdots, n\}$. For a finite set $\mathcal{S}$, we use $|\mathcal{S}|$ to denote the size of $\mathcal{S}$. For random variables $\mathcal{X}$ and $\mathcal{Y}$ over a finite set $\mathcal{S}$, their statistical distance is $\Delta(\mathcal{X}, \mathcal{Y}) := \frac{1}{2} \sum_{s \in \mathcal{S}} |\Pr[\mathcal{X} = s] - \Pr[\mathcal{Y} = s]|$.

We recall the formal definitions of PKE, collision-resistant hash functions and universal hash functions together with the leftover hash lemma in Appendix A.1, A.2 and A.3, respectively.

## 3 Anonymity and Confidentiality under Corruptions

In this section, we firstly introduce the notion of Anonymity under Receiver Selective Opening attacks (in the multi-challenge setting), adaptive user Corruptions and Chosen Plaintext / Ciphertext Attacks, which we call ANON-RSO$_k$&C-CPA/CCA security ($k \in \mathbb{N}$). Then, we introduce the notion of SIM-RSO$_k$&C-CPA/CCA security ($k \in \mathbb{N}$), to capture confidentiality under the same types of corruptions. Finally, we also introduce the notion of AC-RSO$_k$&C-CPA/CCA security, to capture ANON-RSO$_k$&C-CPA/CCA security and SIM-RSO$_k$&C-CPA/CCA security in one notion for convenience.

### 3.1 Anonymity under Corruptions

**ANON-RSO$_k$&C security.** We formalize a simulation-based anonymity definition under receiver selective opening attacks and adaptive user corruptions, which we call ANON-RSO$_k$&C security ($k \in \mathbb{N}$).

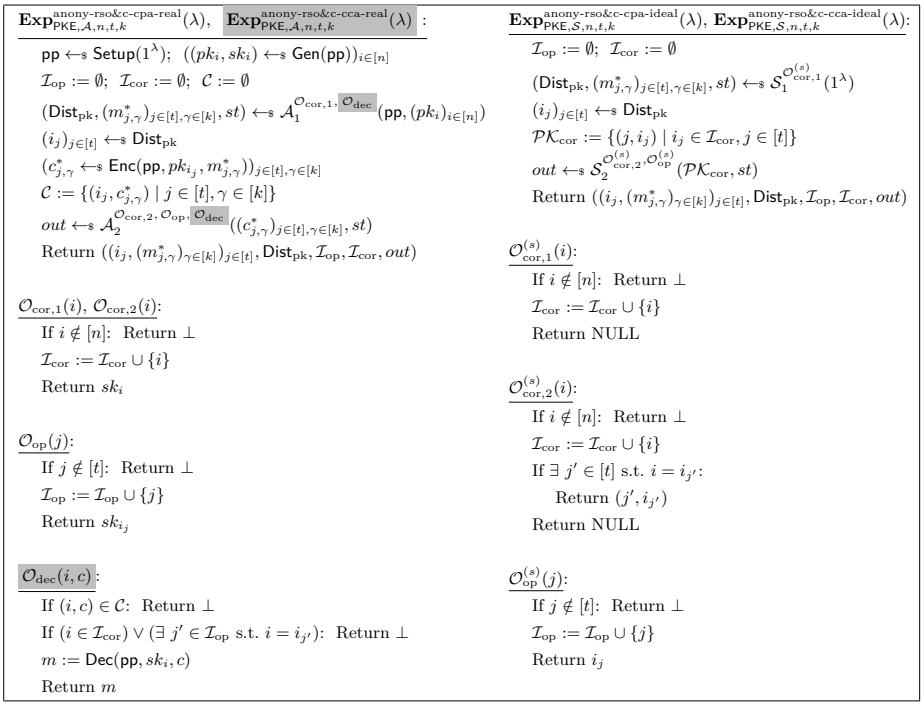


**Fig. 1** Experiments for defining ANON-RSO$_k$&C-CPA/CCA security of scheme PKE.

Informally speaking, assume that there are $n$ users, and that a PPT adversary is allowed to (i) adaptively corrupt the users (i.e., obtaining their secret keys) at any time, and (ii) make receiver selective opening queries (i.e., obtaining the corresponding secret keys and the challenge messages) after seeing a challenge ciphertext vector of length $t < n$. ANON-RSO$_k$&C security requires that whatever the adversary (seeing the challenge ciphertext vector) deduces about which public keys are used to generate the challenge ciphertext vector, can also be deduced without seeing any challenge ciphertexts.

Formal definition is as follows.

**Definition 1. (ANON-RSO$_k$&C-CPA/CCA).** *A PKE scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ANON-RSO$_k$&C-ATK secure *(where* ATK $\in \{\text{CPA}, \text{CCA}\}$ *and* $k \in \mathbb{N}$ *is a constant), if for any polynomially bounded* $n, t$ *(where* $0 < t \leq n$*), and any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there is a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$*, such that for any PPT distinguisher* $\mathcal{D}$*, the advantage* $\mathbf{Adv}^{\text{anon-rso\&c-atk}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D},n,t,k}(\lambda) :=$

$$\left|\Pr[\mathcal{D}(\mathbf{Exp}^{\text{anon-rso\&c-atk-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathbf{Exp}^{\text{anon-rso\&c-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)) = 1]\right|$$

*is negligible, where* $\mathbf{Exp}^{\text{anon-rso\&c-atk-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$ *and* $\mathbf{Exp}^{\text{anon-rso\&c-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)$ *are defined in Fig. 1, and* $\text{atk} \in \{\text{cpa}, \text{cca}\}$*. In both of the experiments, we require that*

*for all* $\mathsf{Dist}_{\mathrm{pk}}$ *output by* $\mathcal{A}_1$ *and* $\mathcal{S}_1$*, it holds that (1)* $\mathsf{Dist}_{\mathrm{pk}}$ *is efficiently samplable, and (2) for all* $(i_j)_{j\in[t]} \leftarrow_{\$} \mathsf{Dist}_{\mathrm{pk}}$*,* $i_{j_1} \neq i_{j_2}$ *for any distinct* $j_1, j_2 \in [t]$*.*

*Remark 1.* Our security notion ANON-RSO$_k$&C-CPA/CCA grants the adversary multiple, adaptive opening queries (i.e., $\mathcal{O}_{\mathrm{op}}$), like [BHK12].

*Remark 2.* In $\mathbf{Exp}^{\text{anon-rso\&c-atk-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$ where $\mathrm{atk} \in \{\mathrm{cpa}, \mathrm{cca}\}$, there are totally $n$ public keys $(pk_i)_{i\in[n]}$, and only $t$ of them (i.e., $(pk_{i_j})_{j\in[t]}$) are used to generate the challenge ciphertexts $(c^*_{j,\gamma})_{j\in[t],\gamma\in[k]}$. Note that (i) by querying the opening oracle $\mathcal{O}_{\mathrm{op}}$ on $j \in [t]$ directly, $\mathcal{A}$ can obtain $sk_{i_j}$ corresponding to some specified $(c^*_{j,\gamma})_{\gamma\in[k]}$; (ii) by querying the corruption oracle $\mathcal{O}_{\mathrm{cor},1}$ or $\mathcal{O}_{\mathrm{cor},2}$, $\mathcal{A}$ can obtain some corresponding secret keys of the $n$ public keys, but cannot ask for the secret key corresponding to some specified $c^*_{j,\gamma}$ since it may not know the value of $i_j$.

**ANON-RSO$_k$&C-CPA $\Rightarrow$ ANON-COR.** We show that ANON-RSO$_k$&C-CPA security implies the ANON-COR security [BGG+20].

Informally, the experiment for defining ANON-COR security is as follows. At the beginning, the challenger generates $n$ public keys $(pk_i)_{i\in[n]}$, and sends them to an adversary $\mathcal{A}$. After receiving $t$ $(t < n)$ messages from $\mathcal{A}$, the challenger randomly samples $t$ distinct public keys from $(pk_i)_{i\in[n]}$, uses them to encrypt the $t$ messages respectively, and sends the $t$ ciphertexts back to $\mathcal{A}$. Then, $\mathcal{A}$ can access to a corruption oracle adaptively, by querying it on any $i \in [n]$ and receiving $sk_i$ as a response. Denote by $Q$ the total number of corruption queries made by $\mathcal{A}$. ANON-COR security requires that for any $\epsilon > 0$ and any $\lambda < t$, $Q < n(1-\epsilon)$, no PPT adversary $\mathcal{A}$ can compromise more than $\frac{Q}{n} + \epsilon$ fraction of the ciphertext-encrypting keys with non-negligible probability. Formal definition of ANON-COR security is given in Appendix B.

Note that any ANON-COR adversary can be seen as an ANON-RSO$_k$&C-CPA adversary $\mathcal{A}$ which (i) ignores $(c^*_{j,\gamma})_{2\leq\gamma\leq k}$ for all $j \in [t]$ if $k > 1$, (ii) does not query $\mathcal{O}_{\mathrm{cor},1}$ or $\mathcal{O}_{\mathrm{op}}$, (iii) queries $\mathcal{O}_{\mathrm{cor},2}$ $Q$ times, and (iv) the output distribution $\mathsf{Dist}_{\mathrm{pk}}$ always samples $t$ distinct indexes $i_1, \cdots, i_t$ uniformly random from $[n]$. The fraction of the ciphertext-encrypting keys that ANON-COR adversary compromises over $(pk_{i_j})_{j\in[t]}$ can be computed directly from experiment $\mathbf{Exp}^{\text{anon-rso\&c-cpa-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$. ANON-RSO$_k$&C-CPA security guarantees that there is a simulator $\mathcal{S}$ such that $\mathbf{Exp}^{\text{anon-rso\&c-cpa-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)$ and $\mathbf{Exp}^{\text{anon-rso\&c-cpa-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$ are indistinguishable. Note that in $\mathbf{Exp}^{\text{anon-rso\&c-cpa-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)$, $\mathcal{S}$ has no information about $(i_j)_{j\in[t]}$ except for the responses obtained via querying $\mathcal{O}^{(s)}_{\mathrm{cor},1}$, $\mathcal{O}^{(s)}_{\mathrm{cor},2}$. Hence, the fraction of the "ciphertext-encrypting" indexes that $\mathcal{S}$ compromises over $(i_j)_{j\in[t]}$ is nearly $\frac{Q}{n}$. Therefore, the indistinguishability between $\mathbf{Exp}^{\text{anon-rso\&c-cpa-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)$ and $\mathbf{Exp}^{\text{anon-rso\&c-cpa-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$ implies the advantage of the ANON-COR adversary is negligible.

### 3.2 Confidentiality under Corruptions

**SIM-RSO$_k$&C security.** In order to capture confidentiality under the same corruptions which are considered in ANON-RSO$_k$&C security, we introduce a

new security notion, called SIM-RSO$_k$&C security. We stress that SIM-RSO$_k$&C security is similar to SIM-RSO$_k$ security [YLH+20], except that the SIM-RSO$_k$&C adversary is allowed to corrupt the receivers *at any time* (i.e., even before seeing the challenge ciphertexts).

Informally, assume that there are $n$ users, and that a PPT adversary is allowed to (i) adaptively corrupt the users (i.e., obtaining their secret keys) at any time, and (ii) make receiver selective opening queries (i.e., obtaining the corresponding secret keys and the challenge messages) after seeing a challenge ciphertext vector of length $n$. SIM-RSO$_k$&C security requires that whatever the adversary (seeing the challenge ciphertext vector) deduces about the challenge messages, can also be deduced without seeing any challenge ciphertexts.

Formal definition is as follows.

**Definition 2. (SIM-RSO$_k$&C-CPA/CCA).** *A PKE scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* SIM-RSO$_k$&C-ATK secure *(where* ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$ *and* $k \in \mathbb{N}$ *is a constant), if for any polynomially bounded* $n > 0$*, and any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there is a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$*, such that for any PPT distinguisher* $\mathcal{D}$*, the advantage* $\mathbf{Adv}^{\text{sim-rso\&c-atk}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D},n,k}(\lambda) :=$

$$\left|\Pr[\mathcal{D}(\mathbf{Exp}^{\text{sim-rso\&c-atk-real}}_{\mathsf{PKE},\mathcal{A},n,k}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathbf{Exp}^{\text{sim-rso\&c-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,k}(\lambda)) = 1]\right|$$

*is negligible, where* $\mathbf{Exp}^{\text{sim-rso\&c-atk-real}}_{\mathsf{PKE},\mathcal{A},n,k}(\lambda)$ *and* $\mathbf{Exp}^{\text{sim-rso\&c-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,k}(\lambda)$ *are defined in Fig. 2, and* $\mathrm{atk} \in \{\mathrm{cpa}, \mathrm{cca}\}$*. In both of the experiments, we require that for all* $\mathsf{Dist}_\mathrm{m}$ *output by* $\mathcal{A}_1$ *and* $\mathcal{S}_1$*,* $\mathsf{Dist}_\mathrm{m}$ *is efficiently samplable.*

**SIM-RSO$_k$&C-ATK $\Rightarrow$ SIM-RSO$_k$-ATK.** We claim that SIM-RSO$_k$&C-ATK security (ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$) implies simulation-based RSO security in the multi-challenge setting (i.e., SIM-RSO$_k$-ATK security) [YLH+20].

Generally, SIM-RSO$_k$-ATK security requires that for any PPT adversary $\mathcal{A}$ in the real experiment of SIM-RSO$_k$-ATK, there is a simulator $\mathcal{S}$, such that the final output of the ideal experiment and that of the real experiment are indistinguishable. Standard SIM-RSO-ATK security [HPW15, HKM+18, HLC+19] is a special case of SIM-RSO$_k$-ATK security (i.e., $k = 1$). For completeness, formal definition of SIM-RSO$_k$-ATK security is given in Appendix B.

The reason that SIM-RSO$_k$&C-ATK security implies SIM-RSO$_k$-ATK security is as follows. Note that any SIM-RSO$_k$-ATK adversary $\mathcal{A}$ can be seen as a SIM-RSO$_k$&C-ATK adversary which does not query the corruption oracles $\mathcal{O}_{\mathrm{cor},1}, \mathcal{O}_{\mathrm{cor},2}$. SIM-RSO$_k$&C-ATK security guarantees the existence of a simulator $\mathcal{S}'$, such that the final output of the ideal experiment and that of the real experiment are indistinguishable. Hence, for the final output of the ideal experiment $((m^*_{i,\gamma})_{i\in[n],\gamma\in[k]}, \mathsf{Dist}_\mathrm{m}, \mathcal{I}_\mathrm{op}, \mathcal{I}_\mathrm{cor}, out)$, it also holds that $\mathcal{I}_\mathrm{cor} = \emptyset$ (i.e., $\mathcal{S}'$ has never queried $\mathcal{O}^{(s)}_{\mathrm{cor},1}, \mathcal{O}^{(s)}_{\mathrm{cor},2}$). Hence, a SIM-RSO$_k$-ATK simulator $\mathcal{S}$ can be constructed from $\mathcal{S}'$.

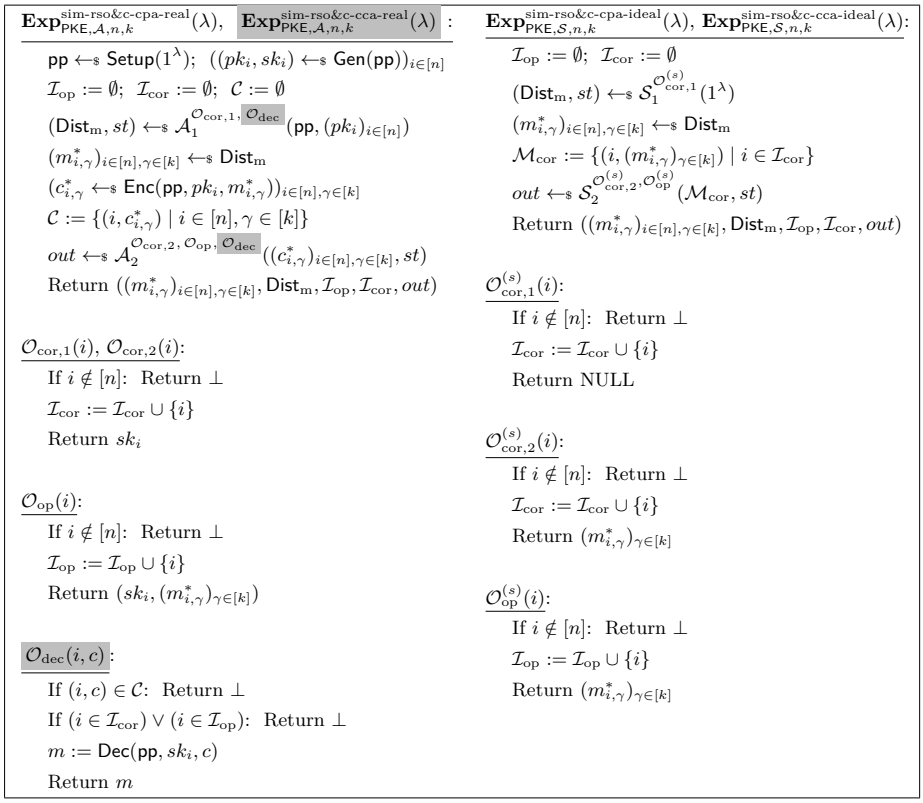


**Fig. 2** Experiments for defining SIM-RSO$_k$&C-CPA/CCA security of scheme PKE.

### 3.3 Combining Anonymity and Confidentiality under Corruptions

We introduce the notion of AC-RSO$_k$&C-CPA/CCA security, to capture ANON-RSO$_k$&C-CPA/CCA security and SIM-RSO$_k$&C-CPA/CCA security in one notion for convenience.

Informally, assume that there are $n$ users, and that a PPT adversary is allowed to (i) adaptively corrupt the users (i.e., obtaining their secret keys) at any time, and (ii) make receiver selective opening queries (i.e., obtaining the corresponding secret keys and the challenge messages) after seeing a challenge ciphertext vector of length $t < n$. AC-RSO$_k$&C security requires that whatever the adversary (seeing the challenge ciphertext vector) deduces about which public keys or messages are used to generate the challenge ciphertext vector, can also be deduced without seeing any challenge ciphertexts.

Formal definition is as follows.

**Definition 3. (AC-RSO$_k$&C-CPA/CCA).** *A PKE scheme* PKE = (Setup, Gen, Enc, Dec) *is* AC-RSO$_k$&C-ATK secure *(where* ATK $\in$ {CPA, CCA} *and* $k \in \mathbb{N}$ *is a constant), if for any polynomially bounded* $n, t$ *(where* $0 < t \leq n$*),*

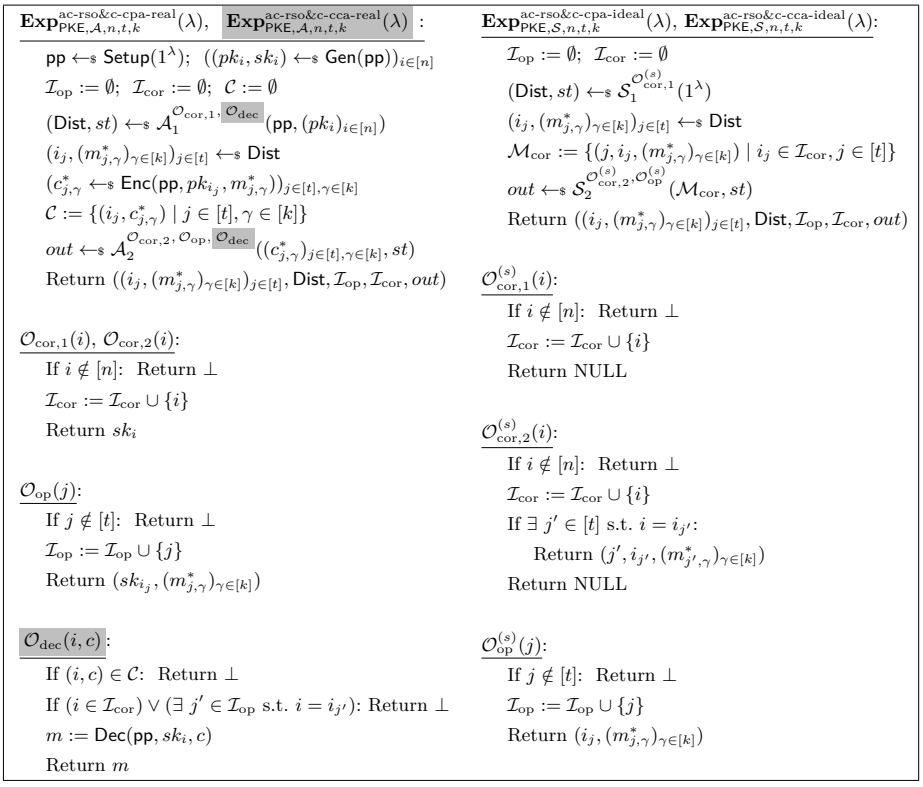


**Fig. 3** Experiments for defining AC-RSO$_k$&C-CPA/CCA security of scheme PKE.

*and any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there is a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$*, such that for any PPT distinguisher* $\mathcal{D}$*, the advantage* $\mathbf{Adv}^{\text{ac-rso\&c-atk}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D},n,t,k}(\lambda) :=$

$$\left|\Pr[\mathcal{D}(\mathbf{Exp}^{\text{ac-rso\&c-atk-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathbf{Exp}^{\text{ac-rso\&c-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)) = 1]\right|$$

*is negligible, where* $\mathbf{Exp}^{\text{ac-rso\&c-atk-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$ *and* $\mathbf{Exp}^{\text{ac-rso\&c-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)$ *are defined in Fig. 3, and* $\text{atk} \in \{\text{cpa}, \text{cca}\}$*. In both of the experiments, we require that for all* Dist *output by* $\mathcal{A}_1$ *and* $\mathcal{S}_1$*, it holds that (1)* Dist *is efficiently samplable, and (2) for all* $(i_j, (m^*_{j,\gamma})_{\gamma\in[k]})_{j\in[t]} \leftarrow\!\!\$\ \mathsf{Dist}$*,* $i_{j_1} \neq i_{j_2}$ *for any distinct* $j_1, j_2 \in [t]$*.*

Note that, AC-RSO$_k$&C security can be easily simplified to guarantee only ANON-RSO$_k$&C security (when the adversary chooses a distribution Dist that has no entropy in the message part) and can also be simplified to guarantee only SIM-RSO$_k$&C security (by letting $n = t$).

# 4 AC-RSO$_k$&C Secure PKE from KM-NCE

In this section, we introduce a new primitive called key and message non-committing encryption (KM-NCE), and two security requirements, KMNC$_k$-CPA/CCA and robustness, for it. Then, we show that KMNC$_k$-CPA/CCA secure and robust KM-NCE implies AC-RSO$_k$&C-CPA/CCA secure PKE.

## 4.1 Key and Message Non-Committing Encryption

Now we provide the definition of key and message non-committing encryption (KM-NCE) and security properties for this primitive. Informally, a KM-NCE scheme is a PKE scheme with the property that there is a way to generate fake ciphertexts without any public key, such that any $k$ fake ciphertexts can be later opened to any $k$ messages (by showing an appropriate secret key). This primitive is an extension of receiver non-committing encryption (RNCE) in [CHK05, HPW15, HKM+18]. Generally speaking, the main differences between KM-NCE and RNCE are that (i) KM-NCE is defined in the $k$-challenge setting, for some constant $k$, and (ii) the algorithm, generating fake ciphertexts, of KM-NCE does not take any public key as input, while that of RNCE needs the public key.

For $k \in \mathbb{N}$, a key and message non-committing encryption scheme $\mathsf{KM\text{-}NCE}$ in the $k$-challenge setting, with a message space $\mathcal{M}$, consists of six PPT algorithms $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Open}_k)$.

- $\mathsf{Setup}$: The setup algorithm, given a security parameter $1^\lambda$, outputs a public parameter $\mathsf{pp}$.
- $\mathsf{Gen}$: The key generation algorithm, given $\mathsf{pp}$, outputs a public key $pk$, a secret key $sk$ and a trapdoor key $tk$.
- $\mathsf{Enc}$: The encryption algorithm, given $\mathsf{pp}$, $pk$ and a message $m \in \mathcal{M}$, outputs a ciphertext $c$.
- $\mathsf{Dec}$: The (deterministic) decryption algorithm, given $\mathsf{pp}$, $sk$ and $c$, outputs $m \in \mathcal{M} \cup \{\perp\}$.
- $\mathsf{Fake}$: The fake encryption algorithm, given $\mathsf{pp}$, outputs a fake ciphertext $c'$ and a trapdoor $td$.
- $\mathsf{Open}_k$: The opening algorithm, given $(\mathsf{pp}, tk, pk, sk)$, $k$ fake ciphertexts $(c'_\gamma)_{\gamma\in[k]}$, $k$ trapdoors $(td_\gamma)_{\gamma\in[k]}$ correponding to $(c'_\gamma)_{\gamma\in[k]}$, and $k$ messages $(m_\gamma)_{\gamma\in[k]}$, outputs a secret key $sk'$.

For KM-NCE, standard correctness is required. Formally, we require that for any $\mathsf{pp}$ generated by $\mathsf{Setup}$, any $(pk, sk, tk)$ generated by $\mathsf{Gen}(\mathsf{pp})$ and any $m \in \mathcal{M}$, it holds that $\mathsf{Dec}(\mathsf{pp}, sk, \mathsf{Enc}(\mathsf{pp}, pk, m)) = m$.

**Definition 4. (KMNC$_k$-CPA/CCA).** *For $k \in \mathbb{N}$, a KM-NCE scheme* $\mathsf{KM\text{-}NCE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Open}_k)$*, in the $k$-challenge setting, is* KMNC$_k$-ATK secure *(where* ATK $\in \{\text{CPA}, \text{CCA}\}$*), if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, the advantage* $\mathbf{Adv}^{\text{kmnc-atk}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda) :=$

$$\left|\Pr[\mathbf{Exp}^{\text{kmnc-atk-real}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{kmnc-atk-sim}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda) = 1]\right|$$

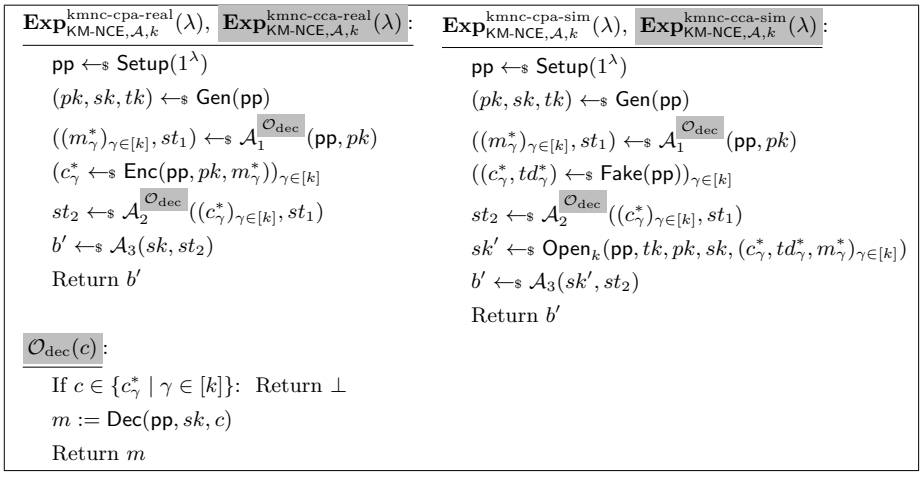


**Fig. 4** Experiments for defining $\text{KMNC}_k$-CPA/CCA security of scheme KM-NCE.

*is negligible, where experiment* $\mathbf{Exp}^{\text{kmnc-atk-real}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda)$ *and* $\mathbf{Exp}^{\text{kmnc-atk-sim}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda)$ *are defined in Fig. 4, and* $\text{atk} \in \{\text{cpa}, \text{cca}\}$.

We also define a statistical robustness for KM-NCE.

**Definition 5 (Robustness).** *A KM-NCE scheme* $\mathsf{KM\text{-}NCE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Open}_k)$*, in the k-challenge setting* ($k \in \mathbb{N}$)*, is* robust*, if the probability* $\boldsymbol{\epsilon}^{\text{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda) :=$

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda), (pk, sk, tk) \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp}),\\ (c, td) \leftarrow\!\!\$\ \mathsf{Fake}(\mathsf{pp})\end{array} : \mathsf{Dec}(\mathsf{pp}, sk, c) \neq \perp\right]$$

*is negligible.*

### 4.2 Generic Construction of AC-RSO$_k$&C Secure PKE from KM-NCE

In this section, we show that for $k \in \mathbb{N}$, a $\text{KMNC}_k$-CPA (resp. $\text{KMNC}_k$-CCA) secure and robust KM-NCE scheme implies an AC-$\text{RSO}_k$&C-CPA (resp. AC-$\text{RSO}_k$&C-CCA) secure PKE scheme. Specifically, we have the following theorem.

**Theorem 1.** *If a KM-NCE scheme* $\mathsf{KM\text{-}NCE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Open}_k)$*, in the k-challenge setting* ($k \in \mathbb{N}$)*, is* $\text{KMNC}_k$*-CPA (resp.* $\text{KMNC}_k$*-CCA) secure and robust, then* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an AC-*$\text{RSO}_k$*&C-CPA (resp. AC-*$\text{RSO}_k$*&C-CCA) secure PKE scheme.*[5]

[5] For $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we require that (i) the public parameter pp generated by Setup can be used for multiple users, and (ii) Gen does not output $tk$ (i.e., the key generation algorithm of PKE firstly invokes the key generation algorithm of KM-NCE to generate $(pk, sk, tk)$, and then outputs $(pk, sk)$, ignoring $tk$).

**Proof of Theorem 1.** We just prove that a KMNC$_k$-CCA secure and robust KM-NCE scheme implies an AC-RSO$_k$&C-CCA secure PKE scheme. The proof for the case of CPA is analogous and much easier, so we omit the details here.

Let $n$ and $t$ be arbitrary polynomials satisfying $0 < t \leq n$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any PPT adversary attacking $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ in the sense of AC-RSO$_k$&C-CCA, and $\mathcal{D}$ be any PPT distinguisher. Without loss of generality, we assume that $\mathcal{A}$ never repeats an oracle query. Specifically, we assume that if $\mathcal{A}_1$ has queried oracle $\mathcal{O}_{\text{cor},1}$ on some $i$, then $\mathcal{A}_2$ will not query $\mathcal{O}_{\text{cor},2}$ on $i$.

We proceed in a series of games.

**Game $\mathsf{G}_{-1}$:** This is exactly the $\mathbf{Exp}^{\text{ac-rso\&c-cca-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$ experiment, i.e., $\mathsf{G}_{-1} = \mathbf{Exp}^{\text{ac-rso\&c-cca-real}}_{\mathsf{PKE},\mathcal{A},n,t,k}(\lambda)$.

More specifically, in $\mathsf{G}_{-1}$, the challenger firstly generates $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$ and $((pk_i, sk_i, tk_i) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp}))_{i\in[n]}$, and sends $(\mathsf{pp}, (pk_i)_{i\in[n]})$ to $\mathcal{A}_1$. The challenger initiates $\mathcal{I}_{\text{op}} := \emptyset$ and $\mathcal{I}_{\text{cor}} := \emptyset$, and keeps track of all $\mathcal{A}$'s issued queries to $\mathcal{O}_{\text{cor},1}, \mathcal{O}_{\text{cor},2}, \mathcal{O}_{\text{op}}$ by maintaining these two sets. Then, the challenger answers $\mathcal{A}_1$'s $\mathcal{O}_{\text{cor},1}, \mathcal{O}_{\text{dec}}$ oracle queries with $(sk_i)_{i\in[n]}$. After receiving $\mathsf{Dist}$, the challenger samples $(i_j, (m^*_{j,\gamma})_{\gamma\in[k]})_{j\in[t]} \leftarrow \mathsf{Dist}$, computes $(c^*_{j,\gamma} \leftarrow_{\$} \mathsf{Enc}(\mathsf{pp}, pk_{i_j}, m^*_{j,\gamma}))_{j\in[t],\gamma\in[k]}$, sets that $\mathcal{C} := \{(i_j, c^*_{j,\gamma}) \mid j \in [t], \gamma \in [k]\}$, and sends $(c^*_{j,\gamma})_{j\in[t],\gamma\in[k]}$ to $\mathcal{A}_2$. Then, the challenger continues to answer $\mathcal{A}_2$'s $\mathcal{O}_{\text{cor},2}, \mathcal{O}_{\text{op}}, \mathcal{O}_{\text{dec}}$ oracle queries with $(sk_i)_{i\in[n]}$. Finally, when $\mathcal{A}_2$ returns $out$, the challenger returns $((i_j, (m^*_{j,\gamma})_{\gamma\in[k]})_{j\in[t]}, \mathsf{Dist}, \mathcal{I}_{\text{op}}, \mathcal{I}_{\text{cor}}, out)$ as its final output.

**Game $\mathsf{G}_0$:** Game $\mathsf{G}_0$ is the same as $\mathsf{G}_{-1}$, except that two sets $\mathcal{I}_{\text{op-sk}}$ and $\mathcal{I}_{\text{cor-sk}}$ are introduced in $\mathsf{G}_0$. Informally, $\mathcal{I}_{\text{op-sk}}$ is introduced to ensure that if $\mathcal{A}_2$ submits a query $\mathcal{O}_{\text{cor},2}(i)$ such that the secret key corresponding to $pk_i$ has already been given to $\mathcal{A}$ via oracle $\mathcal{O}_{\text{op}}$, then the challenger will directly return the secret key previously given to $\mathcal{A}_2$; $\mathcal{I}_{\text{cor-sk}}$ is introduced to ensure that if $\mathcal{A}_2$ submits a query $\mathcal{O}_{\text{op}}(j)$ such that the secret key corresponding to $pk_{i_j}$ has already been exposed to $\mathcal{A}$ in a previous corruption query, then the challenger will directly return the secret key previously given to $\mathcal{A}_2$.

Specifically, the differences between $\mathsf{G}_0$ and $\mathsf{G}_{-1}$ are as follows. The challenger additionally initiates $\mathcal{I}_{\text{op-sk}} := \emptyset$ and $\mathcal{I}_{\text{cor-sk}} := \emptyset$ at the beginning, and answers $\mathcal{A}$'s $\mathcal{O}_{\text{cor},1}, \mathcal{O}_{\text{cor},2}, \mathcal{O}_{\text{op}}$ oracle queries as below:

- on a query $\mathcal{O}_{\text{cor},1}(i)$ where $i \in [n]$, the challenger sets $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}$ and $\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk_i)\}$, and returns $sk_i$ to $\mathcal{A}_1$;
- on a query $\mathcal{O}_{\text{cor},2}(i)$ where $i \in [n]$, the challenger firstly sets $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}$. If there is some $j' \in \mathcal{I}_{\text{op}}$ such that $i_{j'} = i$, then there must be some tuple $(j', i, \overline{sk_i}) \in \mathcal{I}_{\text{op-sk}}$, and in this case the challenger sets $\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, \overline{sk_i})\}$, and returns $\overline{sk_i}$ to $\mathcal{A}_2$; otherwise, it sets $\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk_i)\}$, and returns $sk_i$ to $\mathcal{A}_2$;
- on a query $\mathcal{O}_{\text{op}}(j)$ where $j \in [t]$, the challenger firstly sets $\mathcal{I}_{\text{op}} := \mathcal{I}_{\text{op}} \cup \{j\}$. If $i_j \in \mathcal{I}_{\text{cor}}$, there must be some tuple $(i_j, \overline{sk_{i_j}}) \in \mathcal{I}_{\text{cor-sk}}$, and in this case the challenger sets $\mathcal{I}_{\text{op-sk}} := \mathcal{I}_{\text{op-sk}} \cup \{(j, i_j, \overline{sk_{i_j}})\}$, and returns $\overline{sk_{i_j}}$ to $\mathcal{A}_2$; otherwise, it sets $\mathcal{I}_{\text{op-sk}} := \mathcal{I}_{\text{op-sk}} \cup \{(j, i_j, sk_{i_j})\}$, and returns $sk_{i_j}$ to $\mathcal{A}_2$.

Since all the secret keys $(sk_i)_{i\in[n]}$ are generated at the beginning and will not be updated during the proceedings of $\mathsf{G}_{-1}$, the modifications introduced in game $\mathsf{G}_0$ do not change $\mathcal{A}$'s view. Hence, $\Pr[\mathcal{D}(\mathsf{G}_0)=1]=\Pr[\mathcal{D}(\mathsf{G}_{-1})=1]$.

**Game $\mathsf{G}_{\widehat{i}}$ ($\widehat{i}\in[n]$):** For all $\widehat{i}\in[n]$, $\mathsf{G}_{\widehat{i}}$ is the same as $\mathsf{G}_{\widehat{i}-1}$, except that

(1) when generating the challenge ciphertexts, if there is some $j'\in[t]$ such that $(i_{j'}\notin\mathcal{I}_{\mathrm{cor}})\wedge(i_{j'}=\widehat{i})$, the challenger generates $(c^*_{j',\gamma})_{\gamma\in[k]}$ with algorithm $\mathsf{Fake}$ instead of $\mathsf{Enc}$, i.e., $((c^*_{j',\gamma},td^*_{j',\gamma})\leftarrow_{\$}\mathsf{Fake}(\mathsf{pp}))_{\gamma\in[k]}$;
(2) for $\mathcal{A}_2$'s each $\mathcal{O}_{\mathrm{cor},2}$ oracle query $i$, if there is some $j'\in[t]$ satisfying $(j'\notin\mathcal{I}_{\mathrm{op}})\wedge(i_{j'}=\widehat{i})$, the challenger returns $sk'_{i_{j'}}\leftarrow_{\$}\mathsf{Open}_k(\mathsf{pp},tk_{i_{j'}},pk_{i_{j'}},sk_{i_{j'}},(c^*_{j',\gamma},td^*_{j',\gamma},m^*_{j',\gamma})_{\gamma\in[k]})$ to $\mathcal{A}_2$; otherwise, it answers this query as in $\mathsf{G}_{\widehat{i}-1}$;
(3) for $\mathcal{A}_2$'s each $\mathcal{O}_{\mathrm{op}}$ oracle query $j$, if the corresponding $i_j$ satisfies $(i_j\notin\mathcal{I}_{\mathrm{cor}})\wedge(i_j=\widehat{i})$, the challenger returns $sk'_{i_j}\leftarrow_{\$}\mathsf{Open}_k(\mathsf{pp},tk_{i_j},pk_{i_j},sk_{i_j},(c^*_{j,\gamma},td^*_{j,\gamma},m^*_{j,\gamma})_{\gamma\in[k]})$ to $\mathcal{A}_2$; otherwise, it answers this query as in $\mathsf{G}_{\widehat{i}-1}$.

**Game $\mathsf{G}_{n+\widehat{i}}$ ($\widehat{i}\in[n]$):** For all $\widehat{i}\in[n]$, game $\mathsf{G}_{n+\widehat{i}}$ is the same as $\mathsf{G}_{n+\widehat{i}-1}$, except that for $\mathcal{A}_2$'s each $\mathcal{O}_{\mathrm{dec}}$ oracle query $(i,c)$, if $(\exists(i_j,c^*_{j,\gamma})\in\mathcal{C}$ s.t. $i_j=\widehat{i}\wedge c^*_{j,\gamma}=c)\wedge(i\notin\mathcal{I}_{\mathrm{cor}})$, the challenger returns $\perp$ to $\mathcal{A}_2$; otherwise, it answers this query as in game $\mathsf{G}_{n+\widehat{i}-1}$.

We present the following two lemmas whose proofs are given in Appendix C.1 and C.2.

**Lemma 1.** *For each $\widehat{i}\in[n]$, $|\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}})=1]-\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1})=1]|\le\mathbf{Adv}^{\text{kmnc-cca}}_{\mathsf{KM\text{-}NCE},\mathcal{B},k}(\lambda)$ for some PPT adversary $\mathcal{B}$.*

**Lemma 2.** *For each $\widehat{i}\in[n]$, $|\Pr[\mathcal{D}(\mathsf{G}_{n+\widehat{i}})=1]-\Pr[\mathcal{D}(\mathsf{G}_{n+\widehat{i}-1})=1]|\le t\cdot k\cdot\boldsymbol{\epsilon}^{\mathrm{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda)$.*

Note that in game $\mathsf{G}_{2n}$, (i) when generating the challenge ciphertexts, for each $j\in[t]$ such that $i_j\notin\mathcal{I}_{\mathrm{cor}}$, the corresponding challenge ciphertexts $(c^*_{j,\gamma})_{\gamma\in[k]}$ are generated with algorithm $\mathsf{Fake}$; (ii) any $\mathcal{O}_{\mathrm{cor},2}$ oracle query $i\in[n]$ such that $i=i_{j'}$ for some $j'\notin\mathcal{I}_{\mathrm{op}}$ is answered with algorithm $\mathsf{Open}_k$; (iii) any $\mathcal{O}_{\mathrm{op}}$ oracle query $j\in[t]$ such that $i_j\notin\mathcal{I}_{\mathrm{cor}}$ is answered with algorithms $\mathsf{Open}_k$; (iv) any $\mathcal{O}_{\mathrm{dec}}$ oracle query $(i,c)$ is answered with $\perp$ if there is some $j\in[t]$ and $\gamma\in[k]$ such that $(i_j,c^*_{j,\gamma}=c)\in\mathcal{C}$ and $c^*_{j,\gamma}$ is generated with algorithm $\mathsf{Fake}$. Now, a PPT simulator $\mathcal{S}=(\mathcal{S}_1,\mathcal{S}_2)$ can be constructed as in Fig. 5.

Since $\mathcal{S}$ simulates $\mathsf{G}_{2n}$ perfectly for $\mathcal{A}$, we derive that

$$\mathbf{Exp}^{\text{ac-rso\&c-cca-ideal}}_{\mathsf{PKE},\mathcal{S},n,t,k}(\lambda)=\mathsf{G}_{2n}.$$

Therefore, $\mathbf{Adv}^{\text{ac-rso\&c-cca}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D},n,t,k}(\lambda)=|\Pr[\mathcal{D}(\mathsf{G}_{-1})=1]-\Pr[\mathcal{D}(\mathsf{G}_{2n})=1]|$

$$\le n\cdot\mathbf{Adv}^{\text{kmnc-cca}}_{\mathsf{KM\text{-}NCE},\mathcal{B}',k}(\lambda)+n\cdot t\cdot k\cdot\boldsymbol{\epsilon}^{\mathrm{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda)\tag{1}$$

for some PPT adversary $\mathcal{B}'$. This completes the proof of Theorem 1. ■

$\underline{\mathcal{S}_1^{\mathcal{O}^{(s)}_{\text{cor},1}}(1^\lambda):}$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$; $((pk_i, sk_i, tk_i) \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp}))_{i\in[n]}$
$\mathcal{I}_{\text{op}} := \emptyset$; $\mathcal{I}_{\text{cor}} := \emptyset$; $\mathcal{I}_{\text{op-sk}} := \emptyset$; $\mathcal{I}_{\text{cor-sk}} := \emptyset$; $\mathcal{C}_{\mathcal{S}} := \emptyset$; $(\mathsf{Dist}, st) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\text{cor},1},\mathcal{O}_{\text{dec}}}(\mathsf{pp}, (pk_i)_{i\in[n]})$
$st_{\mathcal{S}} = (\mathsf{pp}, (pk_i, sk_i, tk_i)_{i\in[n]}, \mathcal{I}_{\text{op}}, \mathcal{I}_{\text{cor}}, \mathcal{I}_{\text{op-sk}}, \mathcal{I}_{\text{cor-sk}}, \mathcal{C}_{\mathcal{S}}, \mathsf{Dist}, st)$
Return $(\mathsf{Dist}, st_{\mathcal{S}})$

$\underline{\mathcal{S}_2^{\mathcal{O}^{(s)}_{\text{cor},2},\mathcal{O}^{(s)}_{\text{op}}}(\mathcal{M}_{\text{cor}}, st_{\mathcal{S}}):}$
For $j \in [t]$:
If $\exists\ (j, i_j, (m^*_{j,\gamma})_{\gamma\in[k]}) \in \mathcal{M}_{\text{cor}}$: $(c^*_{j,\gamma} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pp}, pk_{i_j}, m^*_{j,\gamma}))_{\gamma\in[k]}$
Else: $((c^*_{j,\gamma}, td^*_{j,\gamma}) \leftarrow\!\!\$\ \mathsf{Fake}(\mathsf{pp}))_{\gamma\in[k]}$; $\mathcal{C}_{\mathcal{S}} := \mathcal{C}_{\mathcal{S}} \cup \{c^*_j\}$ // the information about $td^*_j$ is stored by $\mathcal{S}_2$
$out \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\text{cor},2},\mathcal{O}_{\text{op}},\mathcal{O}_{\text{dec}}}((c^*_{j,\gamma})_{j\in[t],\gamma\in[k]}, st)$
Return $out$

On query $\mathcal{O}_{\text{cor},1}(i)$:
If $i \notin [n]$: Return $\perp$
$\mathcal{O}^{(s)}_{\text{cor},1}(i)$; $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}$; $\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk_i)\}$
Return $sk_i$

On query $\mathcal{O}_{\text{cor},2}(i)$:
If $i \notin [n]$: Return $\perp$
$\mathsf{resp} \leftarrow\!\!\$\ \mathcal{O}^{(s)}_{\text{cor},2}(i)$; $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}$
If $\mathsf{resp} = (j', i_{j'} = i, (m^*_{j',\gamma})_{\gamma\in[k]})$ for some $j'$:
If $j' \in \mathcal{I}_{\text{op}}$: //in this case, there must be some $(j', i_{j'}, \overline{sk}_{i_{j'}}) \in \mathcal{I}_{\text{op-sk}}$
$\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, \overline{sk}_{i_{j'}})\}$; Return $\overline{sk}_{i_{j'}}$
Else:
$sk'_{i_{j'}} \leftarrow\!\!\$\ \mathsf{Open}_k(\mathsf{pp}, tk_{i_{j'}}, pk_{i_{j'}}, sk_{i_{j'}}, (c^*_{j',\gamma}, td^*_{j',\gamma}, m^*_{j',\gamma})_{\gamma\in[k]})$; $\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk'_{i_{j'}})\}$
Return $sk'_{i_{j'}}$
If $\mathsf{resp} = \text{NULL}$:
$\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk_i)\}$; Return $sk_i$

On query $\mathcal{O}_{\text{op}}(j)$:
If $j \notin [t]$: Return $\perp$
$(i_j, m^*_j) \leftarrow\!\!\$\ \mathcal{O}^{(s)}_{\text{op}}(j)$; $\mathcal{I}_{\text{op}} := \mathcal{I}_{\text{op}} \cup \{j\}$
If $i_j \in \mathcal{I}_{\text{cor}}$: //in this case, there must be some $(i_j, \overline{sk}_{i_j}) \in \mathcal{I}_{\text{cor-sk}}$
$\mathcal{I}_{\text{op-sk}} := \mathcal{I}_{\text{op-sk}} \cup \{(j, i_j, \overline{sk}_{i_j})\}$; Return $(\overline{sk}_{i_j}, (m^*_{j,\gamma})_{\gamma\in[k]})$
Else:
$sk'_{i_j} \leftarrow\!\!\$\ \mathsf{Open}_k(\mathsf{pp}, tk_{i_j}, pk_{i_j}, sk_{i_j}, (c^*_{j,\gamma}, td^*_{j,\gamma}, m^*_{j,\gamma})_{\gamma\in[k]})$; $\mathcal{I}_{\text{op-sk}} = \mathcal{I}_{\text{op-sk}} \cup \{(j, i_j, sk'_{i_j})\}$
Return $(sk'_{i_j}, m^*_j)$

On query $\mathcal{O}_{\text{dec}}(i, c)$:
If $\exists\ (j, i_j, (m^*_{j,\gamma})_{\gamma\in[k]}) \in \mathcal{M}_{\text{cor}}$ s.t. $(i = i_j) \wedge (c \in \{c^*_{j,\gamma} \mid \gamma \in [k]\})$: Return $\perp$
If $(c \in \mathcal{C}_{\mathcal{S}}) \vee (i \in \mathcal{I}_{\text{cor}}) \vee (\exists\ j' \in \mathcal{I}_{\text{op}}$ s.t. $i = i_{j'})$: Return $\perp$
$m := \mathsf{Dec}(\mathsf{pp}, sk_i, c)$
Return $m$

**Fig. 5** Simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ in the proof of Theorem 1.

## 5 KM-NCE from Key-Openable Hash Proof System

In this section, we present a generic construction of KM-NCE that is needed in the AC-RSO$_k$&C secure PKE construction in Sect. 4.2. Our main building block is a new variant of Hash Proof System (HPS), called *Key-Openable HPS*. We firstly recall the definition of HPS from [CS02], and then formalize our new Key-Openable HPS. Next, we show how to construct KM-NCE from Key-Openable

HPS. Jumping ahead, we will give concrete instantiations of Key-Openable HPS from the matrix decisional Diffie-Hellman assumption in Sect. 6.

### 5.1 Recall: Hash Proof System

In this subsection, we recall the formal definition of HPS according to [CS02]. For applications in constructing KM-NCE, we require that HPS has two parameter generation algorithms, a master parameter generation algorithm $\mathsf{MPar}$ and an (ordinary) parameter generation algorithm $\mathsf{Par}$.

**Definition 6 (Hash Proof System).** *A hash proof system* $\mathsf{HPS} = (\mathsf{MPar}, \mathsf{Par}, \mathsf{Pub}, \mathsf{Priv})$ *consists of a tuple of PPT algorithms:*

- $\mathsf{mpar} \leftarrow_{\$} \mathsf{MPar}(1^{\lambda})$*: The master parameter generation algorithm outputs a master public parameter* $\mathsf{mpar}$*, which implicitly defines the universe set* $\mathcal{X}$ *and the hash value space* $\Pi$*.*

  *We assume that there are PPT algorithms for sampling* $x \leftarrow_{\$} \mathcal{X}$ *uniformly and sampling* $\pi \leftarrow_{\$} \Pi$ *uniformly. We require* $\mathsf{mpar}$ *to be an implicit input of other algorithms.*
- $\mathsf{par} \leftarrow_{\$} \mathsf{Par}(\mathsf{mpar})$*: The (ordinary) parameter generation algorithm takes* $\mathsf{mpar}$ *as input, and outputs an (ordinary) public parameter* $\mathsf{par}$*, which implicitly defines* $(\mathcal{L}, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)}, \alpha)$*, where* $\mathcal{L} \subseteq \mathcal{X}$ *is an NP-language,* $\mathcal{SK}$ *is the hashing key space,* $\mathcal{PK}$ *is the projection key space,* $\Lambda_{(\cdot)} : \mathcal{X} \longrightarrow \Pi$ *is a family of hash functions indexed by a hashing key* $\mathsf{sk} \in \mathcal{SK}$*, and* $\alpha : \mathcal{SK} \longrightarrow \mathcal{PK}$ *is the projection function.*

  *We assume that* $\Lambda_{(\cdot)}$ *and* $\alpha$ *are efficiently computable and there are PPT algorithms for sampling* $x \leftarrow_{\$} \mathcal{L}$ *uniformly together with a witness* $w$*, and sampling* $\mathsf{sk} \leftarrow_{\$} \mathcal{SK}$ *uniformly. We require* $\mathsf{par}$ *to be an implicit input of other algorithms.*
- $\pi \leftarrow \mathsf{Pub}(\mathsf{pk}, x, w)$*: The public evaluation algorithm outputs the hash value* $\pi = \Lambda_{\mathsf{sk}}(x) \in \Pi$ *of* $x \in \mathcal{L}$*, with the help of a projection key* $\mathsf{pk} = \alpha(\mathsf{sk})$ *and a witness* $w$ *for* $x \in \mathcal{L}$*.*
- $\pi \leftarrow \mathsf{Priv}(\mathsf{sk}, x)$*: The private evaluation algorithm outputs the hash value* $\pi = \Lambda_{\mathsf{sk}}(x) \in \Pi$ *of* $x \in \mathcal{X}$*, directly using the hashing key* $\mathsf{sk}$*.*

Perfect correctness *(a.k.a. projectiveness) of* $\mathsf{HPS}$ *requires that, for all possible* $\mathsf{mpar} \leftarrow_{\$} \mathsf{MPar}(1^{\lambda})$ *and* $\mathsf{par} \leftarrow_{\$} \mathsf{Par}(\mathsf{mpar})$*, all hashing keys* $\mathsf{sk} \in \mathcal{SK}$ *with* $\mathsf{pk} := \alpha(\mathsf{sk})$ *the corresponding projection key, all* $x \in \mathcal{L}$ *with all possible witnesses* $w$*, it holds that* $\mathsf{Pub}(\mathsf{pk}, x, w) = \Lambda_{\mathsf{sk}}(x) = \mathsf{Priv}(\mathsf{sk}, x)$*.*

HPS is associated with a subset membership problem (SMP), which asks whether an element is uniformly chosen from $\mathcal{L}$ or $\mathcal{X}$. SMP can be extended to multi-fold SMP by considering multiple elements.

**Definition 7 (Multi-fold SMP).** *The multi-fold SMP related to* $\mathsf{HPS}$ *is hard, if for any PPT adversary* $\mathcal{A}$ *and any polynomial* $Q$*, it holds that* $\mathbf{Adv}^{Q\text{-msmp}}_{\mathsf{HPS},\mathcal{A}}(\lambda) :=$

$\left|\Pr[\mathcal{A}(\mathsf{mpar}, \mathsf{par}, \{x_\gamma\}_{\gamma\in[Q]}) = 1] - \Pr[\mathcal{A}(\mathsf{mpar}, \mathsf{par}, \{x'_\gamma\}_{\gamma\in[Q]}) = 1]\right| \leq \mathsf{negl}(\lambda)$, *where* $\mathsf{mpar} \leftarrow_\$ \mathsf{MPar}(1^\lambda)$, $\mathsf{par} \leftarrow_\$ \mathsf{Par}(\mathsf{mpar})$, $x_\gamma \leftarrow_\$ \mathcal{L}$ *and* $x'_\gamma \leftarrow_\$ \mathcal{X}$ *for each* $\gamma \in [Q]$.

*Tag-based HPS.* We recall a tag-based variant of HPS from [CS02, QLC15], by allowing the hash functions $\Lambda_{(\cdot)}$ to have an additional element called label/tag as input. More precisely, in a tag-based HPS, the public parameter $\mathsf{par}$ also implicitly defines a tag space $\mathcal{T}$. Meanwhile, the hash functions $\Lambda_{(\cdot)}$, the public evaluation algorithm $\mathsf{Pub}$ and the private evaluation algorithm $\mathsf{Priv}$ also take a tag $\tau \in \mathcal{T}$ as input. Accordingly, perfect correctness requires $\mathsf{Pub}(\mathsf{pk}, x, w, \tau) = \Lambda_{\mathsf{sk}}(x, \tau) = \mathsf{Priv}(\mathsf{sk}, x, \tau)$ for all tags $\tau \in \mathcal{T}$.

### 5.2 Key-Openable HPS

We present the formal definition of our new *Key-Openable HPS*.

**Definition 8 (Key-Openable Hash Proof System).** *Let* $k \in \mathbb{N}$. *A key-openable hash proof system* $\mathsf{HPS} = (\mathsf{MPar}, \mathsf{Par}, \mathsf{Pub}, \mathsf{Priv}, \mathsf{HOpen}_k)$ *consists of a tuple of PPT algorithms:*

- $(\mathsf{MPar}, \mathsf{Par}, \mathsf{Pub}, \mathsf{Priv})$ *is a hash proof system as per Definition 6. Recall that the master parameter* $\mathsf{mpar}$ *output by* $\mathsf{MPar}(1^\lambda)$ *implicitly defines* $(\mathcal{X}, \Pi)$, *and there are PPT algorithms for sampling* $x \leftarrow_\$ \mathcal{X}$ *uniformly and sampling* $\pi \leftarrow_\$ \Pi$ *uniformly. We denote by* $R_\mathcal{X}$ *and* $R_\Pi$ *the randomness spaces for sampling* $x \leftarrow_\$ \mathcal{X}$ *and* $\pi \leftarrow_\$ \Pi$ *respectively.*
- *In addition to public parameter* $\mathsf{par}$, $\mathsf{Par}(\mathsf{mpar})$ *also outputs a trapdoor information* $\mathsf{td}$, *which will be later used by* $\mathsf{HOpen}_k$.
- $\mathsf{sk}'/\perp \leftarrow_\$ \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x_\gamma, r_{x_\gamma}, \pi_\gamma, r_{\pi_\gamma})_{\gamma\in[k]})$*: The hashing key opening algorithm takes as input the trapdoor* $\mathsf{td}$, *a projection key* $\mathsf{pk} \in \mathcal{PK}$, *a hashing key* $\mathsf{sk} \in \mathcal{SK}$ *satisfying* $\mathsf{pk} = \alpha(\mathsf{sk})$, *and* $k$ *tuples* $(x_\gamma, r_{x_\gamma}, \pi_\gamma, r_{\pi_\gamma})_{\gamma\in[k]}$ *where* $x_\gamma \in \mathcal{X}$ *with sampling randomness* $r_{x_\gamma} \in R_\mathcal{X}$ *and* $\pi_\gamma \in \Pi$ *with sampling randomness* $r_{\pi_\gamma} \in R_\Pi$ *for each* $\gamma \in [k]$, *and outputs another hashing key* $\mathsf{sk}' \in \mathcal{SK}$ *satisfying* $\mathsf{pk} = \alpha(\mathsf{sk}')$ *and* $\pi_\gamma = \Lambda_{\mathsf{sk}'}(x_\gamma)$ *for each* $\gamma \in [k]$, *or a special symbol* $\perp$ *indicating the failure of opening.*

*Tag-based Key-Openable HPS.* A key-openable $\mathsf{HPS} = (\mathsf{MPar}, \mathsf{Par}, \mathsf{Pub}, \mathsf{Priv}, \mathsf{HOpen}_k)$ is a tag-based key-openable $\mathsf{HPS}$, if $(\mathsf{MPar}, \mathsf{Par}, \mathsf{Pub}, \mathsf{Priv})$ is a tag-based HPS (cf. Sect. 5.1), and $\mathsf{HOpen}_k$ also takes a set of tags $(\tau_\gamma)_{\gamma\in[k]}$ as input so that its output $\mathsf{sk}'$ satisfies $\mathsf{pk} = \alpha(\mathsf{sk}')$ and $\pi_\gamma = \Lambda_{\mathsf{sk}'}(x_\gamma, \tau_\gamma)$ for each $\gamma \in [k]$.

Below we define a new statistical property for (tag-based) key-openable HPS, called *openability*$_k$. It stipulates the statistical indistinguishability between $(\mathsf{sk}^{(0)}, (\pi_\gamma^{(0)})_{\gamma\in[k]})$ and $(\mathsf{sk}^{(1)}, (\pi_\gamma^{(1)})_{\gamma\in[k]})$, where $\mathsf{sk}^{(0)}$ is a uniformly sampled hashing key, $\pi_\gamma^{(0)} = \Lambda_{\mathsf{sk}_0}(x_\gamma)$ for $x_\gamma \leftarrow_\$ \mathcal{X}$ with randomness $r_{x_\gamma}$, $\pi_\gamma^{(1)}$ is uniformly sampled from $\Pi$ with randomness $r_{\pi_\gamma^{(1)}}$, and $\mathsf{sk}^{(1)}$ is generated by $\mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}^{(0)}, (x_\gamma, r_{x_\gamma}, \pi_\gamma^{(1)}, r_{\pi_\gamma^{(1)}})_{\gamma\in[k]})$. Here the subscript $k$ indicates the opening of hashing

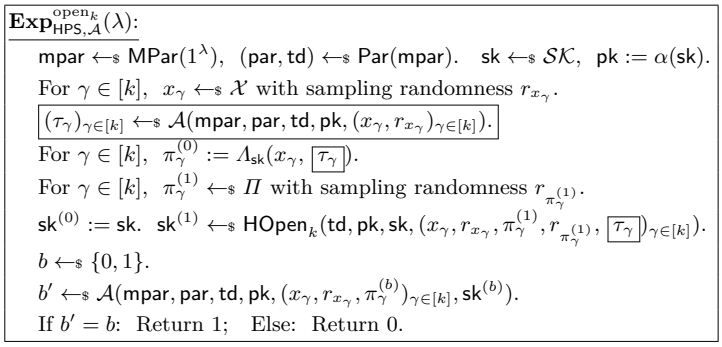


**Fig. 6** Experiment for defining the Openability$_k$ property of (tag-based) key-openable HPS, where the framed parts only appear in the experiment for tag-based HPS.

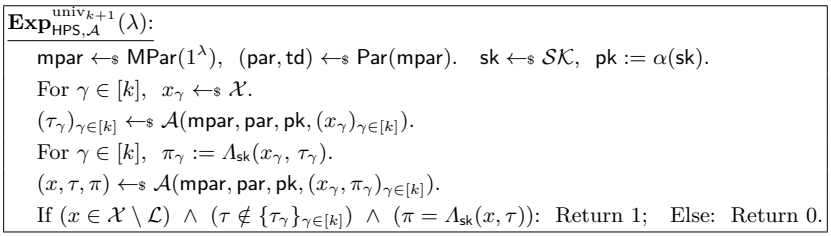


**Fig. 7** Experiment for the Universal$_{k+1}$ property of tag-based key-openable HPS.

key w.r.t. $k$ hash values. For tag-based key-openable HPS, the adversary can additionally determine the tags $(\tau_\gamma)_{\gamma\in[k]}$ w.r.t. which the hash values are computed. It is not hard to see that this property implies the usual smoothness property of HPS [CS02] and also implies that $\mathcal{L}$ is a sparse subset of $\mathcal{X}$.

**Definition 9 (Openability$_k$).** *A (tag-based) key-openable* HPS *is openable$_k$, if for any (unbounded) adversary $\mathcal{A}$, it holds that* $\boldsymbol{\epsilon}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{A}}(\lambda) := |\Pr[\mathbf{Exp}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{A}}(\lambda) = 1] - 1/2| \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{Exp}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{A}}(\lambda)$ *is defined in Fig. 6.*

Next we define a statistical property for tag-based HPS, called *universal$_{k+1}$*, which is an extension of the universal$_2$ property proposed in [CS02].

**Definition 10 (Universal$_{k+1}$).** *A tag-based key-openable* HPS *is universal$_{k+1}$, if for any (unbounded) adversary $\mathcal{A}$, it holds that* $\boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\mathsf{HPS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\mathsf{HPS},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\mathsf{HPS},\mathcal{A}}(\lambda)$ *is defined in Fig. 7.*

Finally, we define a statistical property, called *efficient randomness resampling on $\Pi$*, which demands that besides the (aforementioned) sampling algorithm of $\Pi$ which samples uniform element $\pi \in \Pi$ with sampling randomness $r_\pi$, there is a randomness resampling algorithm $\mathsf{ReSmp}_\Pi$ that takes as input $\pi \in \Pi$ and outputs a sampling randomness $r_\pi$. These two ways of sampling/resampling are statistically indistinguishable.

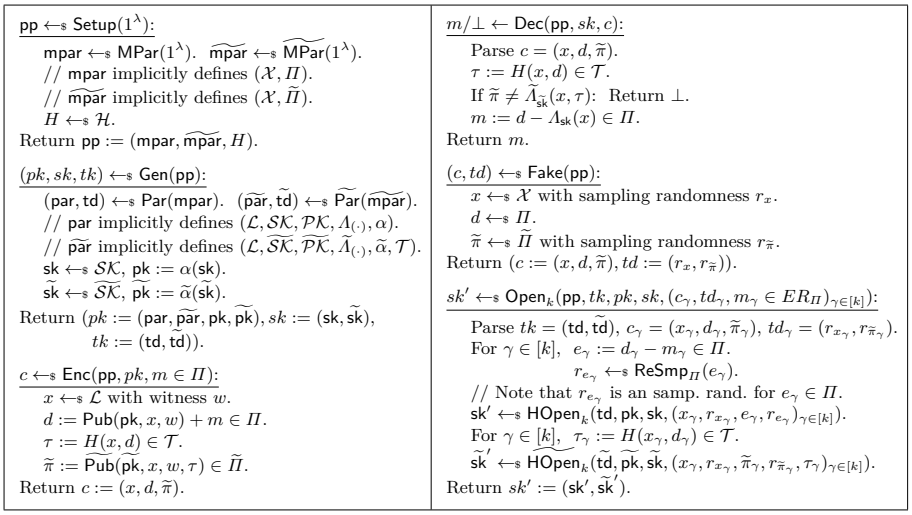


**Fig. 8** Construct. of $\mathsf{KM\text{-}NCE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Open}_k)$ from $\mathsf{HPS}, \widetilde{\mathsf{HPS}}, \mathcal{H}$.

**Definition 11 (Efficient Randomness Resampling on $\Pi$).** *The hash value space $\Pi$ of* HPS *supports efficient randomness resampling, if there exists a PPT algorithm* $\mathsf{ReSmp}_\Pi$*, s.t. the statistical distance* $\epsilon^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda) := \Delta((\pi, r_\pi), (\pi', r'_{\pi'})) \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{mpar} \leftarrow\!\!\$\ \mathsf{MPar}(1^\lambda)$*,* $\pi \leftarrow\!\!\$\ \Pi$ *with sampling randomness* $r_\pi$*,* $\pi' \leftarrow\!\!\$\ \Pi$ *and* $r'_{\pi'} \leftarrow\!\!\$\ \mathsf{ReSmp}_\Pi(\pi')$*.*

### 5.3 Generic Construction of KM-NCE from Key-Openable HPS

The building blocks for constructing KM-NCE are as follows.

- Let $\mathsf{HPS} = (\mathsf{MPar}, \mathsf{Par}, \mathsf{Pub}, \mathsf{Priv}, \mathsf{HOpen}_k)$ be a key-openable HPS, whose hash value space $\Pi$ is an (additive) group and has an efficient randomness resampling algorithm $\mathsf{ReSmp}_\Pi$.
- Let $\widetilde{\mathsf{HPS}} = (\widetilde{\mathsf{MPar}}, \widetilde{\mathsf{Par}}, \widetilde{\mathsf{Pub}}, \widetilde{\mathsf{Priv}}, \widetilde{\mathsf{HOpen}}_k)$ be a tag-based key-openable HPS, which shares same universe $\mathcal{X}$ and same language $\mathcal{L}$ with HPS.
- Let $\mathcal{H} = \{H : \mathcal{X} \times \Pi \to \mathcal{T}\}$ be a family of collision-resistant hash functions (cf. Definition 16 in Appendix A.2), where $\Pi$ is the hash value space of HPS and $\mathcal{T}$ is the tag space of $\widetilde{\mathsf{HPS}}$.

We present the generic construction of $\mathsf{KM\text{-}NCE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Open}_k)$ from HPS, $\widetilde{\mathsf{HPS}}$ and $\mathcal{H}$ in Fig. 8. The message space is $\Pi$. Note that our generic construction of KM-NCE from key-openable HPS is reminiscent of [HLLG19], which constructs PKE scheme from another variant of HPS (the so-called quasi-adaptive HPS).

The perfect correctness of KM-NCE follows from those of HPS and $\widetilde{\mathsf{HPS}}$ directly. Next, we show its $\mathrm{KMNC}_k$-CCA security.

**Theorem 2 (KMNC$_k$-CCA security of KM-NCE).** *Assume that (1)* HPS *is openable$_k$, has a hard multi-fold SMP, supports efficient randomness resampling on $\Pi$, (2)* $\widetilde{\mathsf{HPS}}$ *is universal$_{k+1}$ and openable$_k$, (3) $\mathcal{H}$ is collision-resistant. Then the* KM-NCE *in Fig. 8 is KMNC$_k$-CCA secure.*

*Concretely, for any PPT adversary $\mathcal{A}$ against the KMNC$_k$-CCA security of* KM-NCE *that makes at most $Q_d$ decryption queries, there exist PPT adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and unbounded adversaries $\mathcal{B}_3$, $\mathcal{B}_4$, $\mathcal{B}_5$, s.t.*

$$\begin{aligned}\mathbf{Adv}^{\text{kmnc-cca}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda) \le\ & \mathbf{Adv}^{k\text{-msmp}}_{\mathsf{HPS},\mathcal{B}_1}(\lambda) + 2\cdot\mathbf{Adv}^{\text{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda) + 2Q_d\cdot\boldsymbol{\epsilon}^{\text{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda) \\ &+\ 2\boldsymbol{\epsilon}^{\text{open}_k}_{\mathsf{HPS},\mathcal{B}_4}(\lambda) + 2\boldsymbol{\epsilon}^{\text{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{B}_5}(\lambda) + 2k\cdot\boldsymbol{\epsilon}^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda).\end{aligned} \tag{2}$$

**Proof of Theorem 2.** We prove the theorem by defining a sequence of games $\mathsf{G}_0$-$\mathsf{G}_8$, with $\mathsf{G}_0 = \mathbf{Exp}^{\text{kmnc-cca-real}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda)$ and $\mathsf{G}_8 = \mathbf{Exp}^{\text{kmnc-cca-sim}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda)$, and showing adjacent games indistinguishable. By $\Pr_i[\cdot]$ we denote the probability of a particular event occurring in game $\mathsf{G}_i$.

**Game $\mathsf{G}_0$:** This is the $\mathbf{Exp}^{\text{kmnc-cca-real}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda)$ experiment. Thus, $\Pr[\mathsf{G}_0 = 1] = \Pr[\mathbf{Exp}^{\text{kmnc-cca-real}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda) = 1]$.

In this game, when receiving $(m^*_\gamma)_{\gamma\in[k]}$ from $\mathcal{A}$, the challenger generates $c^*_\gamma$ using the real encryption algorithm $\mathsf{Enc}(\mathsf{pp}, pk, m^*_\gamma)$. More precisely, it samples $x^*_\gamma \leftarrow_\$ \mathcal{L}$ with witness $w^*_\gamma$, computes $d^*_\gamma := \mathsf{Pub}(\mathsf{pk}, x^*_\gamma, w^*_\gamma) + m^*_\gamma$, $\tau^*_\gamma := H(x^*_\gamma, d^*_\gamma)$, $\widetilde{\pi}^*_\gamma := \widetilde{\mathsf{Pub}}(\widetilde{\mathsf{pk}}, x^*_\gamma, w^*_\gamma, \tau^*_\gamma)$, and sets $c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma)$. It returns $(c^*_\gamma)_{\gamma\in[k]}$ to $\mathcal{A}$. When answering decryption queries $\mathcal{O}_{\text{dec}}(c)$ for $\mathcal{A}$ with $c = (x, d, \widetilde{\pi})$, the challenger computes $\tau := H(x, d)$, and outputs $\perp$ immediately if $c \in \{c^*_\gamma\}_{\gamma\in[k]} \vee \widetilde{\pi} \neq \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$. Otherwise, it computes $m := d - \Lambda_{\mathsf{sk}}(x)$ and returns $m$ to $\mathcal{A}$. In the last step of this game, the challenger sends the real secret key $sk = (\mathsf{sk}, \widetilde{\mathsf{sk}})$ to $\mathcal{A}$.

**Game $\mathsf{G}_1$:** It is the same as $\mathsf{G}_0$, except that, for each $\gamma \in [k]$, when generating $c^*_\gamma = (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma)$, the challenger computes $d^*_\gamma$ and $\widetilde{\pi}^*_\gamma$ using $sk = (\mathsf{sk}, \widetilde{\mathsf{sk}})$ instead of using the witness $w^*_\gamma$ of $x^*_\gamma$. Namely, $d^*_\gamma := \Lambda_{\mathsf{sk}}(x^*_\gamma) + m^*_\gamma$ and $\widetilde{\pi}^*_\gamma := \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x^*_\gamma, \tau^*_\gamma)$. By the perfect correctness of HPS and $\widetilde{\mathsf{HPS}}$, this change is conceptual. So $\Pr[\mathsf{G}_1 = 1] = \Pr[\mathsf{G}_0 = 1]$.

**Game $\mathsf{G}_2$:** It is the same as $\mathsf{G}_1$, except that, for each $\gamma \in [k]$, when generating $c^*_\gamma = (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma)$, the challenger samples $x^*_\gamma \leftarrow_\$ \mathcal{X}$ instead of $x^*_\gamma \leftarrow_\$ \mathcal{L}$. Note that neither the witness $w^*_\gamma$ of $x^*_\gamma$ (if $x^*_\gamma \leftarrow_\$ \mathcal{L}$) nor the sampling randomness $r_{x^*_\gamma}$ of $x^*_\gamma$ (if $x^*_\gamma \leftarrow_\$ \mathcal{X}$) is needed in $\mathsf{G}_1$ and $\mathsf{G}_2$, thus it is straightforward to construct a PPT adversary $\mathcal{B}_1$ against the multi-fold SMP, such that $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_1 = 1]| \le \mathbf{Adv}^{k\text{-msmp}}_{\mathsf{HPS},\mathcal{B}_1}(\lambda)$.

**Game $\mathsf{G}_3$:** It is the same as $\mathsf{G}_2$, except that, when answering decryption queries $\mathcal{O}_{\text{dec}}(c)$ for $\mathcal{A}$ with $c = (x, d, \widetilde{\pi})$, the challenger adds a new rejection rule: it outputs $\perp$ immediately if $\tau \in \{\tau^*_\gamma\}_{\gamma\in[k]}$, where $\tau = H(x, d)$ and $\tau^*_\gamma = H(x^*_\gamma, d^*_\gamma)$ for each $\gamma \in [k]$.

Let Bad denote the event that $\mathcal{A}$ ever queries $\mathcal{O}_{\text{dec}}(c)$ with $c = (x, d, \widetilde{\pi})$, such that $(x, d) \notin \{(x^*_\gamma, d^*_\gamma)\}_{\gamma\in[k]}$ but $\tau \in \{\tau^*_\gamma\}_{\gamma\in[k]}$. We first show that $\mathsf{G}_2$ and $\mathsf{G}_3$ are

identical if $\mathsf{Bad}$ does not occur, i.e., either $(x,d) = (x^*_{\gamma_0}, d^*_{\gamma_0})$ for some $\gamma_0 \in [k]$ or $\tau \notin \{\tau^*_\gamma\}_{\gamma\in[k]}$. In the case that $(x,d) = (x^*_{\gamma_0}, d^*_{\gamma_0})$ for some $\gamma_0 \in [k]$, $\mathcal{O}_{\mathrm{dec}}(c)$ would be rejected both in $\mathsf{G}_2$ and $\mathsf{G}_3$ due to $c = c^*_{\gamma_0} \in \{c^*_\gamma\}_{\gamma\in[k]} \vee \widetilde{\pi} \neq \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x,\tau)$. In the case that $\tau \notin \{\tau^*_\gamma\}_{\gamma\in[k]}$, the new rejection rule added in $\mathsf{G}_3$ does not apply, so $\mathcal{O}_{\mathrm{dec}}(c)$ is the same in $\mathsf{G}_2$ and $\mathsf{G}_3$. Overall, $\mathsf{G}_2$ and $\mathsf{G}_3$ are identical when $\mathsf{Bad}$ does not occur, thus by the difference lemma, $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq \Pr_3[\mathsf{Bad}]$.

To bound $\Pr_3[\mathsf{Bad}]$, it is straightforward to construct a PPT adversary $\mathcal{B}_2$ against the collision resistance of $\mathcal{H}$, so that $\Pr_3[\mathsf{Bad}] \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$. Consequently, $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$.

**Game $\mathsf{G}_4$:** It is the same as $\mathsf{G}_3$, except that, when answering decryption queries $\mathcal{O}_{\mathrm{dec}}(c)$ for $\mathcal{A}$ with $c = (x, d, \widetilde{\pi})$, the challenger adds a second new rejection rule: it outputs $\perp$ immediately if $x \in \mathcal{X} \setminus \mathcal{L}$. We note that this new rule may not be PPT checkable, thus the challenger may not be PPT. This does not matter, since the following arguments (before this rule is removed) are statistical.

Let $\mathsf{Forge}$ denote the event that $\mathcal{A}$ ever queries $\mathcal{O}_{\mathrm{dec}}(c)$ with $c = (x, d, \widetilde{\pi})$, such that $\tau \notin \{\tau^*_\gamma\}_{\gamma\in[k]}$, $\widetilde{\pi} = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x,\tau)$ but $x \in \mathcal{X} \setminus \mathcal{L}$. Clearly, $\mathsf{G}_3$ and $\mathsf{G}_4$ are identical unless $\mathsf{Forge}$ occurs, thus by the difference lemma, $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \Pr_4[\mathsf{Forge}]$.

To bound $\Pr_4[\mathsf{Forge}]$, we analyze the information about $\widetilde{\mathsf{sk}}$ that $\mathcal{A}$ may obtain in game $\mathsf{G}_4$ before it finishes the $\mathcal{O}_{\mathrm{dec}}$ queries: $\mathcal{A}$ obtains $\widetilde{\mathsf{pk}} = \alpha(\widetilde{\mathsf{sk}})$ in $pk$ and obtains $\{\widetilde{\pi}^*_\gamma = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x^*_\gamma, \tau^*_\gamma)\}_{\gamma\in[k]}$ in $\{c^*_\gamma\}_{\gamma\in[k]}$; for $\mathcal{O}_{\mathrm{dec}}$ queries, the challenger will not output $m$ unless $x \in \mathcal{L}$ (due to the new rejection rule added in $\mathsf{G}_4$), thus $\mathcal{O}_{\mathrm{dec}}$ reveals nothing about $\widetilde{\mathsf{sk}}$ beyond $\widetilde{\mathsf{pk}} = \alpha(\widetilde{\mathsf{sk}})$.

Then by the universal$_{k+1}$ property of tag-based $\widetilde{\mathsf{HPS}}$, for one $\mathcal{O}_{\mathrm{dec}}(c)$ query made by $\mathcal{A}$, it holds that $\tau \notin \{\tau^*_\gamma\}_{\gamma\in[k]}$, $\widetilde{\pi} = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x,\tau)$ but $x \in \mathcal{X} \setminus \mathcal{L}$ with probability at most $\boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda)$. By a union bound over at most $Q_d$ number of $\mathcal{O}_{\mathrm{dec}}$ queries, we get that $\Pr_4[\mathsf{Forge}] \leq Q_d \cdot \boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda)$. Thus, $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq Q_d \cdot \boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda)$. For completeness, we provide a description of the reduction algorithm $\mathcal{B}_3$ in Appendix C.3.

**Game $\mathsf{G}_5$:** It is the same as $\mathsf{G}_4$, except that, for each $\gamma \in [k]$, when generating $c^*_\gamma = (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma)$, the challenger samples $d^*_\gamma \leftarrow_{\$} \Pi$ uniformly (instead of $d^*_\gamma := \Lambda_{\mathsf{sk}}(x^*_\gamma) + m^*_\gamma$). Moreover, in the last step of this game, the challenger computes $e^*_\gamma := d^*_\gamma - m^*_\gamma \in \Pi$ and resamples $r_{e^*_\gamma} \leftarrow_{\$} \mathsf{ReSmp}_\Pi(e^*_\gamma)$ for each $\gamma \in [k]$, then invokes $\mathsf{sk}' \leftarrow_{\$} \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x^*_\gamma, r_{x^*_\gamma}, e^*_\gamma, r_{e^*_\gamma})_{\gamma\in[k]})$, and sends $(\mathsf{sk}', \widetilde{\mathsf{sk}})$ to $\mathcal{A}$. We have the following lemma whose proof is given in Appendix C.4.

**Lemma 3.** *There exists an unbounded $\mathcal{B}_4$ against the openable$_k$ property of $\mathsf{HPS}$, s.t. $|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq 2 \cdot \boldsymbol{\epsilon}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{B}_4}(\lambda) + 2k \cdot \boldsymbol{\epsilon}^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda)$.*

**Game $\mathsf{G}_6$:** It is the same as $\mathsf{G}_5$, except that, for each $\gamma \in [k]$, when generating $c^*_\gamma = (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma)$, the challenger samples $\widetilde{\pi}^*_\gamma \leftarrow_{\$} \widetilde{\Pi}$ uniformly with randomness $r_{\widetilde{\pi}^*_\gamma}$ (instead of $\widetilde{\pi}^*_\gamma := \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x^*_\gamma, \tau^*_\gamma)$). Moreover, in the last step of this game,

the challenger computes $\widetilde{\mathsf{sk}}' \leftarrow_{\$} \widetilde{\mathsf{HOpen}}_k(\widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, (x^*_\gamma, r_{x^*_\gamma}, \widetilde{\pi}^*_\gamma, r_{\widetilde{\pi}^*_\gamma}, \tau^*_\gamma)_{\gamma\in[k]})$, and sends $(\mathsf{sk}', \widetilde{\mathsf{sk}}')$ to $\mathcal{A}$. We have the following lemma. The proof of this lemma is similar to that of Lemma 3, and is given in Appendix C.5.

**Lemma 4.** *There exists an unbounded $\mathcal{B}_5$ against the openable$_k$ property of tag-based $\widetilde{\mathsf{HPS}}$, s.t. $|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_5 = 1]| \le 2 \cdot \boldsymbol{\epsilon}^{\mathrm{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{B}_5}(\lambda)$.*

**Game $\mathsf{G}_7$:** It is the same as $\mathsf{G}_6$, except that, when answering decryption queries $\mathcal{O}_{\mathrm{dec}}(c)$ for $\mathcal{A}$ with $c = (x, d, \widetilde{\pi})$, the challenger removes the second new rejection rule added in $\mathsf{G}_4$. In other words, it does not check whether $x \in \mathcal{L}$ or $x \in \mathcal{X} \setminus \mathcal{L}$ anymore. We note that the challenger in $\mathsf{G}_7$ is now PPT again.

The change from $\mathsf{G}_6$ to $\mathsf{G}_7$ is symmetric to that from $\mathsf{G}_3$ to $\mathsf{G}_4$. By a similar argument, we get $|\Pr[\mathsf{G}_7 = 1] - \Pr[\mathsf{G}_6 = 1]| \le Q_d \cdot \boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda)$.

**Game $\mathsf{G}_8$:** It is the same as $\mathsf{G}_7$, except that, when answering decryption queries $\mathcal{O}_{\mathrm{dec}}(c)$ for $\mathcal{A}$ with $c = (x, d, \widetilde{\pi})$, the challenger removes the first new rejection rule added in $\mathsf{G}_3$. In other words, it does not check whether $\tau \in \{\tau^*_\gamma\}_{\gamma\in[k]}$ or not anymore.

The change from $\mathsf{G}_7$ to $\mathsf{G}_8$ is symmetric to the change from $\mathsf{G}_2$ to $\mathsf{G}_3$. Similarly, we have that $|\Pr[\mathsf{G}_8 = 1] - \Pr[\mathsf{G}_7 = 1]| \le \mathbf{Adv}^{\mathrm{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$.

Finally, we note that $\mathsf{G}_8$ is exactly the $\mathbf{Exp}^{\text{kmnc-cca-sim}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda)$ experiment.

- For each $\gamma \in [k]$, $c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma)$, where $x^*_\gamma \leftarrow_{\$} \mathcal{X}$ with sampling randomness $r_{x^*_\gamma}$, $d^*_\gamma \leftarrow_{\$} \Pi$, and $\widetilde{\pi}^*_\gamma \leftarrow_{\$} \widetilde{\Pi}$ with randomness $r_{\widetilde{\pi}^*_\gamma}$, the same as the $c^*_\gamma$ generated by $\mathsf{Fake}(\mathsf{pp})$.
- $\mathcal{O}_{\mathrm{dec}}(c)$ queries are answered by $\mathsf{Dec}(\mathsf{pp}, sk, c)$ when $c \notin \{c^*_\gamma\}_{\gamma\in[k]}$.
- In the last step, $(\mathsf{sk}', \widetilde{\mathsf{sk}}')$ is generated by first computing $e^*_\gamma := d^*_\gamma - m^*_\gamma \in \Pi$ and resampling $r_{e^*_\gamma} \leftarrow_{\$} \mathsf{ReSmp}_\Pi(e^*_\gamma)$ for each $\gamma \in [k]$, then invoking $\mathsf{sk}' \leftarrow_{\$} \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x^*_\gamma, r_{x^*_\gamma}, e^*_\gamma, r_{e^*_\gamma})_{\gamma\in[k]})$ and $\widetilde{\mathsf{sk}}' \leftarrow_{\$} \widetilde{\mathsf{HOpen}}_k(\widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, (x^*_\gamma, r_{x^*_\gamma}, \widetilde{\pi}^*_\gamma, r_{\widetilde{\pi}^*_\gamma}, \tau^*_\gamma)_{\gamma\in[k]})$ with $\tau^*_\gamma := H(x^*_\gamma, d^*_\gamma)$, the same as $\mathsf{Open}_k(\mathsf{pp}, tk, pk, sk, (c^*_\gamma, r_{c^*_\gamma}, m^*_\gamma)_{\gamma\in[k]})$ where $r_{c^*_\gamma} = (r_{x^*_\gamma}, r_{\widetilde{\pi}^*_\gamma})$.

Thus, $\Pr[\mathsf{G}_8 = 1] = \Pr[\mathbf{Exp}^{\text{kmnc-cca-sim}}_{\mathsf{KM\text{-}NCE},\mathcal{A},k}(\lambda) = 1]$.

Taking all things together, we obtain (2), thus Theorem 2 follows. ■

Finally, we show the robustness.

**Theorem 3 (Robustness of KM-NCE).** *The proposed KM-NCE in Fig. 8 is robust (cf. Definition 5) with $\boldsymbol{\epsilon}^{\mathrm{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda) \le 1/|\widetilde{\Pi}|$, where $\widetilde{\Pi}$ is the hash value space of $\widetilde{\mathsf{HPS}}$.*

**Proof of Theorem 3.** For $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$, $(pk, sk, tk) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$, $(c, td) \leftarrow_{\$} \mathsf{Fake}(\mathsf{pp})$, we analyze the probability $\boldsymbol{\epsilon}^{\mathrm{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda) = \Pr[\mathsf{Dec}(\mathsf{pp}, sk, c) \neq \perp]$.

- For $(pk, sk, tk) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$, we have $sk = (\mathsf{sk}, \widetilde{\mathsf{sk}})$ where $\widetilde{\mathsf{sk}} \leftarrow_{\$} \widetilde{\mathcal{SK}}$.

- For $(c, td) \leftarrow_{\$} \mathsf{Fake}(\mathsf{pp})$, we have $c = (x, d, \widetilde{\pi})$ where $x \leftarrow_{\$} \mathcal{X}$, $d \leftarrow_{\$} \Pi$ and $\widetilde{\pi} \leftarrow_{\$} \widetilde{\Pi}$.
- Then in $\mathsf{Dec}(\mathsf{pp}, sk, c)$, it first checks whether or not $\widetilde{\pi} = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$ holds, where $\tau := H(x, d)$, and returns $\perp$ if the check fails.

Since $\widetilde{\pi}$ is uniformly chosen from $\widetilde{\Pi}$ and independent of $x, d$ and $\widetilde{\mathsf{sk}}$, so the check $\widetilde{\pi} = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$ passes with probability $1/|\widetilde{\Pi}|$. Overall, we have $\boldsymbol{\epsilon}^{\mathrm{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda) = \Pr[\mathsf{Dec}(\mathsf{pp}, sk, c) \neq \perp] \leq \Pr[\widetilde{\pi} = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)] = 1/|\widetilde{\Pi}|$. ■

## 6 Concrete Instantiations

In this section, we show concrete instantiations of key-openable HPS based on the matrix decisional Diffie-Hellman (MDDH) assumption [EHK+13]. As a result, we can obtain concrete instantiations of KM-NCE, which in turn yields AC-RSO$_k$&C-CCA secure PKE schemes with compact ciphertexts. For certain instantiation, the resulting PKE can even achieve tight AC-RSO$_k$&C-CCA security.

### 6.1 Recall: Matrix Distribution

We recall the definition of matrix distribution defined in [EHK+13].

In this section, we use bold uppercase letters to represent matrices and bold lowercase letters to represent (column) vectors. Let $\mathsf{GGen}$ be a PPT algorithm that on input $1^\lambda$ returns $\mathcal{G} = (\mathbb{G}, q, P)$, a description of an (additive) cyclic group $\mathbb{G}$ with a generator $P$ of order $q$ which is a $\lambda$-bit prime. For $a \in \mathbb{Z}_q$, define $[a] := aP \in \mathbb{G}$ as the *implicit representation* of $a$ in $\mathbb{G}$. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n \times m}$, we define $[\mathbf{A}]$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}$, i.e., $[\mathbf{A}] := (a_{ij}P) \in \mathbb{G}^{n \times m}$. Note that from $[a] \in \mathbb{G}$ it is generally hard to compute the value $a$ (discrete logarithm problem is hard in $\mathbb{G}$). Obviously, given $[a], [b] \in \mathbb{G}$ and a scalar $x \in \mathbb{Z}$, one can efficiently compute $[ax] \in \mathbb{G}$ and $[a + b] \in \mathbb{G}$. Similarly, for $\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{B} \in \mathbb{Z}_q^{n \times t}, \mathbf{AB} \in \mathbb{Z}_q^{m \times t}$, given $[\mathbf{A}], \mathbf{B}$ one can efficiently compute $[\mathbf{A}]\mathbf{B} := [\mathbf{AB}] \in \mathbb{G}^{m \times t}$ and given $\mathbf{A}, [\mathbf{B}]$, one can efficiently compute $\mathbf{A}[\mathbf{B}] := [\mathbf{AB}] \in \mathbb{G}^{m \times t}$.

**Definition 12 (Matrix Distribution).** *Let $d, k \in \mathbb{N}^+$. $\mathcal{D}_{d+k,d}$ is called a* matrix distribution *if it outputs matrices in $\mathbb{Z}_q^{(d+k) \times d}$ of full rank $d$ in polynomial time.*

As in [EHK+13], let $\mathcal{U}_{d+k,d}$ be the uniform distribution over $\mathbb{Z}_q^{(d+k) \times d}$. Without loss of generality, for $\mathbf{A} \leftarrow_{\$} \mathcal{D}_{d+k,d}$, we assume that $\overline{\mathbf{A}}$ (the upper square submatrix of $\mathbf{A}$) is invertible.

**Definition 13 (The $\mathcal{D}_{d+k,d}$-Matrix Decision Diffie-Hellman Assumption, $\mathcal{D}_{d+k,d}$-MDDH).** *Let $\mathcal{D}_{d+k,d}$ be a matrix distribution. The $\mathcal{D}_{d+k,d}$-*Matrix Decision Diffie-Hellman ($\mathcal{D}_{d+k,d}$-MDDH) Assumption *holds relative to $\mathsf{GGen}$ if for each PPT adversary $\mathcal{A}$, the advantage*

$$\mathbf{Adv}^{\mathrm{mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{Aw}]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{u}]) = 1]|$$

*is negligible, where the probability is taken over* $\mathcal{G} \leftarrow_{\$} \mathsf{GGen}(1^\lambda), \mathbf{A} \leftarrow_{\$} \mathcal{D}_{d+k,d}$, $\mathbf{w} \leftarrow_{\$} \mathbb{Z}_q^d$ and $\mathbf{u} \leftarrow_{\$} \mathbb{Z}_q^{d+k}$.

As shown in [EHK+13], $\mathcal{D}_{d+k,d}$-MDDH assumption is a generalization of a large range of assumptions. By setting the matrix distribution $\mathcal{D}_{\ell,k}$ to specific distributions, $\mathcal{D}_{d+k,d}$-MDDH assumption can capture DDH assumption, $k$-Linear assumption, $k$-Cascade assumption and many other assumptions.

The MDDH assumption can be generalized into a multi-instance version. We recall the $Q$-fold MDDH assumption as defined in [EHK+13].

**Definition 14 ($Q$-fold $\mathcal{D}_{d+k,d}$-Matrix Decision Diffie-Hellman Assumption).** *Let $Q$ be a positive integer and $\mathcal{D}_{d+k,d}$ be a matrix distribution. The* $Q$-fold $\mathcal{D}_{d+k,d}$-Matrix Decision Diffie-Hellman Assumption *holds relative to* $\mathsf{GGen}$ *if for each PPT adversary $\mathcal{A}$, the advantage*

$$\mathbf{Adv}^{\text{Q-mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathcal{G},[\mathbf{A}],[\mathbf{AW}])=1] - \Pr[\mathcal{A}(\mathcal{G},[\mathbf{A}],[\mathbf{U}])=1]|$$

*is negligible, where the probability is taken over* $\mathcal{G} \leftarrow_{\$} \mathsf{GGen}(1^\lambda), \mathbf{A} \leftarrow_{\$} \mathcal{D}_{d+k,d}$, $\mathbf{W} \leftarrow_{\$} \mathbb{Z}_q^{d\times Q}$ and $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{(d+k)\times Q}$.

### 6.2 Openable$_k$ HPS Instantiation

In this subsection, we provide a key-openable HPS instantiation with openable$_k$ and efficient randomness resampling properties based on the MDDH assumption. This HPS can be seen as a generalization of the DDH-based HPS in [CS02]. Inspired by the technique in [HKM+18, HJR16], we are able to make the hash value space of our HPS to be compact and efficient randomness resamplable. Meanwhile, this does not affect the openability of our HPS.

More precisely, fixing some group generation algorithm $\mathsf{GGen}$, some positive integers $d, k$, some matrix distribution $\mathcal{D}_{d+k,d}$ and some polynomial $l = l(\lambda)$ (which can be set as the desired message length of the PKE scheme), consider $\mathsf{HPS} = (\mathsf{MPar}, \mathsf{Par}, \mathsf{Pub}, \mathsf{Priv}, \mathsf{HOpen}_k)$ in the following.

- $\mathsf{MPar}(1^\lambda)$. The master parameter generation algorithm runs $\mathcal{G} = (\mathbb{G}, q, P) \leftarrow_{\$} \mathsf{GGen}(1^\lambda)$. Let $\mathcal{H}_{\mathsf{u}} = \{\mathsf{H}_{\mathsf{u}} : \mathbb{G} \to \{0,1\}\}$ be a family of universal hash functions based on group $\mathbb{G}$. The algorithm selects $\mathsf{H}_{\mathsf{u}} \leftarrow_{\$} \mathcal{H}_{\mathsf{u}}$ and returns $\mathsf{mpar} := (\mathcal{G}, d, k, l, \mathcal{D}_{d+k,d}, \mathsf{H}_{\mathsf{u}})$ which implicitly defines the instance space $\mathcal{X} := \mathbb{G}^{d+k}$ with randomness space $R_{\mathcal{X}} := \mathbb{Z}_q^{d+k}$ and the hash value space $\Pi := \{0,1\}^l$ with randomness space $R_\Pi := \mathbb{Z}_q^l$. Given $\mathsf{mpar}$, one can efficiently sample a uniform element $x$ from $\mathcal{X}$ by selecting $r_x = \mathbf{x} \leftarrow_{\$} R_{\mathcal{X}}$ and setting $x := [r_x] = [\mathbf{x}]$. For simplicity, we define an efficiently computable function $\mathsf{H}_{\mathsf{u},l} : \mathbb{G}^l \to \{0,1\}^l$ where $\mathsf{H}_{\mathsf{u},l}([\mathbf{a}]) := (\mathsf{H}_{\mathsf{u}}([a_1]), \cdots, \mathsf{H}_{\mathsf{u}}([a_l]))$ for all $[\mathbf{a}] = [a_1, \cdots, a_l] \in \mathbb{G}^l$. Then, one can also efficiently sample a uniform element $\pi$ from $\Pi$ by selecting $r_\pi = \boldsymbol{\pi} \leftarrow_{\$} R_\Pi$ and setting $\pi := \mathsf{H}_{\mathsf{u},l}([\boldsymbol{\pi}]) \in \Pi$. (Actually, $\pi$ is only statistical close to uniform. According to the leftover hash lemma (Lemma 6 in Appendix A.3) together with the union bound, the statistically distance between $\pi$ and uniform distribution over $\Pi$ is bounded by

$\frac{l}{2}\sqrt{\frac{2}{q}}$, which is exponentially small for polynomially bounded $l$. Therefore, we omit this statistical distance here.)

- $\mathsf{Par}(\mathsf{mpar})$. The (ordinary) parameter generation algorithm selects matrix $\mathbf{A} \in \mathbb{Z}_q^{(d+k)\times d} \leftarrow_{\$} \mathcal{D}_{d+k,d}$, then it returns $\mathsf{par} := [\mathbf{A}]$ and $\mathsf{td} := \mathbf{A}$.
  The public parameter $\mathsf{par}$ (together with $\mathsf{mpar}$) implicitly defines the language as $\mathcal{L} := [\mathsf{span}(\mathbf{A})] = \{[\mathbf{Aw}] \mid \mathbf{w} \in \mathbb{Z}_q^d\}$. The hashing key space $\mathcal{SK} := \mathbb{Z}_q^{(d+k)\times l}$ and the projection key space $\mathcal{PK} := \mathbb{G}^{d\times l}$. The projection function $\alpha$ maps $\mathsf{sk} = \mathbf{S} \in \mathcal{SK}$ to $\mathsf{pk} = [\mathbf{P}] \in \mathcal{PK}$ where $[\mathbf{P}] = [\mathbf{A}^\top]\mathbf{S}$ and $\alpha$ is efficiently computable given $\mathsf{par}$ and $\mathsf{sk}$. For $\mathsf{sk} = \mathbf{S} \in \mathcal{SK}$, the hash function $\Lambda_{\mathsf{sk}}(\cdot)$ maps an element $x = [\mathbf{x}] \in \mathcal{X}$ to $\mathsf{H}_{\mathsf{u},l}(\mathbf{S}^\top[\mathbf{x}]) \in \Pi$ and it is efficiently computable given $\mathsf{sk}$ and $x$.
  Given $\mathsf{par}$, one can efficiently sample a uniform element $x$ from language $\mathcal{L}$ together with a witness $w$ by choosing $w = \mathbf{w} \leftarrow_{\$} \mathbb{Z}_q^d$ and computing $x = [\mathbf{x}] = [\mathbf{A}]\mathbf{w}$.
- $\mathsf{Pub}(\mathsf{pk}, x, w)$. Given public key $\mathsf{pk} = [\mathbf{P}] \in \mathcal{PK}$, an instance $x = [\mathbf{x}] = [\mathbf{Aw}] \in \mathcal{L}$, and its witness $w = \mathbf{w}$, the public evaluation algorithm outputs $\pi = \mathsf{H}_{\mathsf{u},l}([\mathbf{P}^\top]\mathbf{w}) \in \Pi$.
- $\mathsf{Priv}(\mathsf{sk}, x)$. Given secret key $\mathsf{sk} = \mathbf{S} \in \mathcal{SK}$ and $x = [\mathbf{x}] \in \mathcal{X}$, the private evaluation algorithm outputs $\pi = \mathsf{H}_{\mathsf{u},l}(\mathbf{S}^\top[\mathbf{x}]) \in \Pi$.
- $\mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x_\gamma, r_{x_\gamma}, \pi_\gamma, r_{\pi_\gamma})_{\gamma\in\{1,\cdots,k\}})$. Given $\mathsf{td} = \mathbf{A}, \mathsf{pk} = [\mathbf{P}], \mathsf{sk} = \mathbf{S}, x_\gamma = [\mathbf{x}_\gamma], r_{x_\gamma} = \mathbf{x}_\gamma, \pi_\gamma = \mathsf{H}_{\mathsf{u},l}([\boldsymbol{\pi}_\gamma])$ and $r_{\pi_\gamma} = \boldsymbol{\pi}_\gamma \in \mathbb{Z}_q^l$ for all $\gamma \in \{1, \cdots, k\}$, the open algorithm computes $\mathsf{sk}' = \mathbf{S}' \in \mathbb{Z}_q^{(d+k)\times l}$ by solving the following system of linear equations,

$$\mathbf{S}'^\top (\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k) = (\mathbf{S}^\top\mathbf{A} \mid \boldsymbol{\pi}_1 \mid \cdots \mid \boldsymbol{\pi}_k) \bmod q. \tag{3}$$

  Note that, given $\mathsf{td} = \mathbf{A}$ and the randomnesses $(r_{x_\gamma} = \mathbf{x}_\gamma)_{\gamma\in\{1,\cdots,k\}}$, one can easily compute the square matrix $\mathbf{M} = (\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k) \in \mathbb{Z}_q^{(d+k)\times(d+k)}$. If $\mathbf{M}$ is invertible, one can compute and output

$$\mathbf{S}'^\top = (\mathbf{S}^\top\mathbf{A} \mid \boldsymbol{\pi}_1 \mid \cdots \mid \boldsymbol{\pi}_k) \cdot \mathbf{M}^{-1} \bmod q.$$

  If $\mathbf{M}$ is not invertible, algorithm $\mathsf{HOpen}_k$ outputs $\perp$.

Note that the hash value space $\Pi = \{0,1\}^l$ is an additive group with group operation $\oplus$ (string xor). We define its randomness resample algorithm $\mathsf{ReSmp}_\Pi$ in Fig. 9.

**Theorem 4.** *The above instantiation* HPS *(1) is a key-openable HPS; (2) has a hard multi-fold SMP under the multi-fold* $\mathcal{D}_{d+k,d}$-MDDH *assumption (i.e., for any PPT adversary* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that* $\mathbf{Adv}^{k\text{-msmp}}_{\mathsf{HPS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{B}}(\lambda)$*); (3) is openable*$_k$ *and (4) supports efficient randomness resampling on* $\Pi$ *with algorithm* $\mathsf{ReSmp}_\Pi$*.*

We put the proof of Theorem 4 in Appendix C.6.

| $\mathsf{ReSmp}_{\Pi}(\mathbf{b} = (b_1, \cdots, b_l) \in \{0,1\}^l)$: | $\mathsf{OnebitReSmp}(\mathsf{H}_u, b_i \in \{0,1\})$: |
|---|---|
| //Implicit input: $\mathsf{H}_u \in \mathsf{mpar}$ | For $j \in \{1, \cdots, \lambda\}$: |
| For $i \in \{1, \cdots, l\}$: | $r_j \leftarrow\$ \mathbb{Z}_q$ |
| $r_i \leftarrow\$ \mathsf{OnebitReSmp}(\mathsf{H}_u, b_i)$ | If $\mathsf{H}_u([r_j]) = b_i$: Return $r_j$ |
| Return $\mathbf{r} := (r_1, \cdots, r_l)$ | Return $\perp$ |

**Fig. 9** Randomness resample algorithm $\mathsf{ReSmp}_{\Pi}$ for hash value space $\Pi = \{0,1\}^l$ of the hash proof system $\mathsf{HPS}$. The algorithm $\mathsf{OnebitReSmp}$ will return $\perp$ and terminate after $\lambda$ iterations, which makes it a polynomial-time algorithm.

### 6.3 Openable$_k$ and Universal$_{k+1}$ Tag-based HPS Instantiation

In this subsection, we provide a tag-based key-openable HPS instantiation with both openable$_k$ and universal$_{k+1}$ properties based on the MDDH assumption. This tag-based HPS can be seen as a generalization of the tag-based HPS from the DDH assumption in [CS02]. More precisely, fixing some group generation algorithm $\mathsf{GGen}$, some positive integers $d, k$ and some matrix distribution $\mathcal{D}_{d+k,d}$, consider instantiation $\widetilde{\mathsf{HPS}} = (\widetilde{\mathsf{MPar}}, \widetilde{\mathsf{Par}}, \widetilde{\mathsf{Pub}}, \widetilde{\mathsf{Priv}}, \widetilde{\mathsf{HOpen}}_k)$ in the following.

- $\widetilde{\mathsf{MPar}}(1^\lambda)$. The master parameter generation algorithm runs $\mathcal{G} = (\mathbb{G}, q, P) \leftarrow\$ \mathsf{GGen}(1^\lambda)$ and returns $\widetilde{\mathsf{mpar}} := (\mathcal{G}, d, k, \mathcal{D}_{d+k,d})$ which implicitly defines the instance space $\mathcal{X} := \mathbb{G}^{d+k}$ with randomness space $R_{\mathcal{X}} := \mathbb{Z}_q^{d+k}$ and the hash value space $\widetilde{\Pi} := \mathbb{G}$ with randomness space $R_{\widetilde{\Pi}} := \mathbb{Z}_q$.[6] Given $\mathsf{mpar}$, one can efficiently sample a uniform element $x$ from $\mathcal{X}$ by selecting $r_x = \mathbf{x} \leftarrow\$ R_{\mathcal{X}}$ and set $x = [r_x] = [\mathbf{x}]$. One can also efficiently sample a uniform element $\widetilde{\pi}$ from $\widetilde{\Pi}$ by selecting $r_{\widetilde{\pi}} \leftarrow\$ R_{\widetilde{\Pi}}$ and set $\widetilde{\pi} = [r_{\widetilde{\pi}}]$.
- $\widetilde{\mathsf{Par}}(\widetilde{\mathsf{mpar}})$. The (ordinary) parameter generation algorithm selects matrix $\mathbf{A} \in \mathbb{Z}_q^{(d+k)\times d} \leftarrow\$ \mathcal{D}_{d+k,d}$, then it returns $\widetilde{\mathsf{par}} := [\mathbf{A}]$ and $\widetilde{\mathsf{td}} := \mathbf{A}$.
  The public parameter $\widetilde{\mathsf{par}}$ (together with $\widetilde{\mathsf{mpar}}$) implicitly defines the language as $\mathcal{L} := [\mathsf{span}(\mathbf{A})] = \{[\mathbf{Aw}] \mid \mathbf{w} \in \mathbb{Z}_q^d\}$.[7] The hashing key space $\widetilde{\mathcal{SK}} := \mathbb{Z}_q^{2d+2k}$ and the projection key space $\widetilde{\mathcal{PK}} := \mathbb{G}^{2d}$. The projection function $\widetilde{\alpha}$ maps $\widetilde{\mathsf{sk}} = \mathbf{s} = \begin{pmatrix}\mathbf{s}_1\\ \mathbf{s}_2\end{pmatrix} \in \widetilde{\mathcal{SK}}$ (where $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}_q^{d+k}$) to $\widetilde{\mathsf{pk}} = [\mathbf{p}] = \begin{bmatrix}\mathbf{p}_1\\ \mathbf{p}_2\end{bmatrix} = \begin{bmatrix}\mathbf{A}^\top & \\ & \mathbf{A}^\top\end{bmatrix}\mathbf{s} \in \widetilde{\mathcal{PK}}$ (where $[\mathbf{p}_i] = \left[\mathbf{A}^\top \mathbf{s}_i\right] \in \mathbb{G}^d$ for $i \in \{1,2\}$) and $\widetilde{\alpha}$ is efficiently computable given $\widetilde{\mathsf{par}}$ and $\widetilde{\mathsf{sk}}$. The tag space is $\mathcal{T} := \mathbb{Z}_q$. For $\widetilde{\mathsf{sk}} = \mathbf{s} \in \widetilde{\mathcal{SK}}$, the hash function $\widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(\cdot,\cdot)$ maps an element $x = [\mathbf{x}] \in \mathcal{X}$ together with a tag $\tau \in \mathcal{T}$ to $\widetilde{\pi} = \mathbf{s}^\top \begin{bmatrix}\mathbf{x}\\ \tau\mathbf{x}\end{bmatrix} = \left[\mathbf{s}_1^\top \mathbf{x} + \tau \mathbf{s}_2^\top \mathbf{x}\right] \in \widetilde{\Pi}$ and it is efficiently computable given $\widetilde{\mathsf{sk}}, x$ and $\tau$.

[6] To get an instantiation $\widetilde{\mathsf{HPS}}$ which satisfies the conditions of Theorem 2, $\widetilde{\mathsf{HPS}}$ needs to share the same universe set $\mathcal{X}$ with $\mathsf{HPS}$. In that way, we can set $(\mathcal{G}, d, k, \mathcal{D}_{d+k,d})$ in $\widetilde{\mathsf{mpar}}$ to be exactly the same with the ones in $\mathsf{mpar}$.

[7] Similarly, we set $\widetilde{\mathsf{par}} := \mathsf{par}$ and $\widetilde{\mathsf{td}} := \mathsf{td}$ to make sure $\widetilde{\mathsf{HPS}}$ shares the same language $\mathcal{L}$ with $\mathsf{HPS}$.

Given $\widetilde{\mathsf{par}}$, one can efficiently sample a uniform element $x$ from language $\mathcal{L}$ together with a witness $w$ by choosing $w = \mathbf{w} \leftarrow_{\$} \mathbb{Z}_q^d$ and computing $x = [\mathbf{x}] = [\mathbf{A}\mathbf{w}]$.

- $\widetilde{\mathsf{Pub}}(\widetilde{\mathsf{pk}}, x, w, \tau)$. Given public key $\widetilde{\mathsf{pk}} = [\mathbf{p}]$, witness $w = \mathbf{w}$ of instance $x = [\mathbf{A}\mathbf{w}]$ and tag $\tau$, the public evaluation algorithm outputs the hash value $\widetilde{\pi} = [\mathbf{p}^\top] \begin{pmatrix} \mathbf{w} \\ \tau\mathbf{w} \end{pmatrix}$.
- $\widetilde{\mathsf{Priv}}(\widetilde{\mathsf{sk}}, x, \tau)$. Given secret key $\widetilde{\mathsf{sk}} = \mathbf{s}$, $x = [\mathbf{x}]$ and tag $\tau$, the private evaluation algorithm outputs $\widetilde{\pi} = \mathbf{s}^\top \begin{bmatrix} \mathbf{x} \\ \tau\mathbf{x} \end{bmatrix}$.
- $\widetilde{\mathsf{HOpen}}_k(\widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, (x_\gamma, r_{x_\gamma}, \widetilde{\pi}_\gamma, r_{\widetilde{\pi}_\gamma}, \tau_\gamma)_{\gamma\in\{1,\cdots,k\}})$. Given trapdoor $\widetilde{\mathsf{td}} = \mathbf{A}$, public key $\widetilde{\mathsf{pk}} = [\mathbf{p}]$, secret key $\widetilde{\mathsf{sk}} = \mathbf{s}$, instance $x_\gamma = [\mathbf{x}_\gamma]$ with randomness $r_{x_\gamma} = \mathbf{x}_\gamma$, hash value $\widetilde{\pi}_\gamma = [r_{\widetilde{\pi}_\gamma}]$ with randomness $r_{\widetilde{\pi}_\gamma}$ and tag $\tau_\gamma$ for all $\gamma \in \{1, \cdots, k\}$, the open algorithm computes $\widetilde{\mathsf{sk}}' = \mathbf{s}' \in \mathbb{Z}_q^{2d+2k}$ by solving the following system of linear equations.

$$\mathbf{s}'^\top \mathbf{E} = (\mathbf{s}_1^\top \mathbf{A},\ \mathbf{s}_2^\top \mathbf{A},\ r_{\widetilde{\pi}_1}, \cdots,\ r_{\widetilde{\pi}_k}) \bmod q,\ \mathbf{E} = \begin{pmatrix} \mathbf{A} & \mathbf{x}_1 & \cdots & \mathbf{x}_k \\ \mathbf{A} & \tau_1\mathbf{x}_1 & \cdots & \tau_k\mathbf{x}_k \end{pmatrix}. \tag{4}$$

Matrix $\mathbf{E}$ has $2d + 2k$ rows and $2d + k$ columns.

- If matrix $(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k)$ has full column rank $d + k$, then matrix $\mathbf{E}$ has full column rank $2d + k$ and there are $q^k$ possible solutions for $\mathbf{s}'$ to make Equation (4) hold. Algorithm $\widetilde{\mathsf{HOpen}}_k$ selects and outputs a uniformly random solution.
- Otherwise, algorithm $\widetilde{\mathsf{HOpen}}_k$ outputs $\perp$.

Note that given $\widetilde{\mathsf{td}} = \mathbf{A}$, tags $(\tau_\gamma)_{\gamma\in\{1,\cdots,k\}}$ and the randomnesses $(r_{x_\gamma} = \mathbf{x}_\gamma)_{\gamma\in\{1,\cdots,k\}}$, one can easily compute the matrix $\mathbf{E}$. The right hand side of Equation (4) is also efficiently computable given $\widetilde{\mathsf{sk}} = \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix}$ and randomnesses $(r_{\widetilde{\pi}_\gamma})_{\gamma\in\{1,\cdots,k\}}$.

**Theorem 5.** *The above instantiation $\widetilde{\mathsf{HPS}}$ (1) is a tag-based key-openable HPS; (2) is universal$_{k+1}$ and (3) is openable$_k$.*

We put the proof of Theorem 5 in Appendix C.7.

### 6.4 Concrete AC-RSO$_k$&C-CCA secure PKE Instantiation

We instantiate our PKE scheme by plugging the instantiations, $\mathsf{HPS}$ in Section 6.2 and $\widetilde{\mathsf{HPS}}$ in Section 6.3, into the generic KM-NCE construction in Fig. 8. By Theorem 1, we immediately get a PKE instantiation that can achieve AC-RSO$_k$&C-CCA security in the standard model with *compact* ciphertexts. If we set the matrix distribution $\mathcal{D}_{d+k,d}$ (i.e., the matrix distribution used to sample matrix $\mathbf{A}$ by the key generation algorithm $\mathsf{Gen}$) to be uniform matrix distribution $\mathcal{U}_{d+k,d}$, the resulting PKE can achieve *tight* AC-RSO$_k$&C-CCA security.

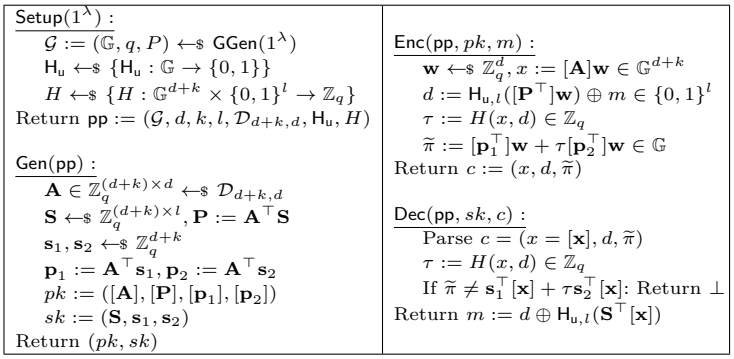


**Fig. 10** Concrete AC-RSO$_k$&C-CCA secure PKE instantiation.

Fixing some group generation algorithm $\mathsf{GGen}$, some positive integers $d, k$, some matrix distribution $\mathcal{D}_{d+k,d}$ and some polynomial $l = l(\lambda)$, the instantiation $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\{0,1\}^l$ is shown in Fig. 10. This scheme can be viewed as a generalization of the DDH-based scheme in [HKM+18, Fig. 3] and both schemes are variants of the Cramer-Shoup encryption scheme [CS02].

We can see that, for the PKE scheme in Fig. 10, the ciphertext length is $(d + k + 1) \times |\mathbb{G}| + l$ for messages of length $l$ and the ciphertext overhead is the size of a constant number of group elements (since $d$ and $k$ are both fixed constants), which is also independent of the message length. This suggests that the PKE instantiation in Fig. 10 has compact ciphertexts [HJR16, HKM+18].

We note that our PKE can achieve *tight* AC-RSO$_k$&C-CCA security for certain instantiation. Taking a closer look at the AC-RSO$_k$&C-CCA security of our MDDH-based PKE instantiation, we obtain the following inequality by combining equation (1) in Theorem 1, equation (2) in Theorem 2 and Theorem 4 together.

$$\begin{aligned}
&\mathbf{Adv}^{\text{ac-rso\&c-cca}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D},n,t,k}(\lambda) \le n \cdot \mathbf{Adv}^{\text{kmnc-cca}}_{\mathsf{KM\text{-}NCE},\mathcal{B}',k}(\lambda) + n \cdot t \cdot k \cdot \boldsymbol{\epsilon}^{\text{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda) \\
&\le n \cdot \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{B}_1}(\lambda) + 2n \cdot \mathbf{Adv}^{\text{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda) + 2Q_d n \cdot \boldsymbol{\epsilon}^{\text{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda) + \ 2n \cdot \boldsymbol{\epsilon}^{\text{open}_k}_{\mathsf{HPS},\mathcal{B}_4}(\lambda) \\
&\quad + 2n \cdot \boldsymbol{\epsilon}^{\text{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{B}_5}(\lambda) + 2kn \cdot \boldsymbol{\epsilon}^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda) + n \cdot t \cdot k \cdot \boldsymbol{\epsilon}^{\text{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda). \qquad (5)
\end{aligned}$$

The $2Q_d n \cdot \boldsymbol{\epsilon}^{\text{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda) + \ 2n \cdot \boldsymbol{\epsilon}^{\text{open}_k}_{\mathsf{HPS},\mathcal{B}_4}(\lambda) + 2n \cdot \boldsymbol{\epsilon}^{\text{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{B}_5}(\lambda) + 2kn \cdot \boldsymbol{\epsilon}^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda) + n \cdot t \cdot k \cdot \boldsymbol{\epsilon}^{\text{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda)$ part in equation (5) does not affect tightness of the reduction since it is statistically small. Only reductions to computational properties matter to tightness of the reduction, i.e., the term $n \cdot \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{B}_1}(\lambda) + 2n \cdot \mathbf{Adv}^{\text{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$. This security loss $n$ and $2n$ are introduced because 1) in the proof of Theorem 1 (KMNC$_k$-CCA + robustness $\Rightarrow$ AC-RSO$_k$&C-CCA), we handle one user at a time with $n$ game transitions (cf. Lemma 1), and in each transition, a term $\mathbf{Adv}^{\text{kmnc-cca}}_{\mathsf{KM\text{-}NCE},\mathcal{B}'_1,k}(\lambda)$ is incurred; 2) according to Theorem 2, the term

$\mathbf{Adv}^{\text{kmnc-cca}}_{\mathsf{KM\text{-}NCE},\mathcal{B}'_1,k}(\lambda)$ contains $\mathbf{Adv}^{k\text{-msmp}}_{\mathsf{HPS},\mathcal{B}''_1}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$; and 3) according to Theorem 4, $\mathbf{Adv}^{k\text{-msmp}}_{\mathsf{HPS},\mathcal{B}''_1}(\lambda) \leq \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{B}_1}(\lambda)$.

Alternatively, if we set the matrix distribution to be uniform matrix distribution (i.e., $\mathcal{D}_{d+k,d} := \mathcal{U}_{d+k,d}$), we can avoid such security loss by integrating the proofs of Theorem 1, Theorem 2 and Theorem 4. We can handle the $n$ reductions to the $k$-fold $\mathcal{U}_{d+k,d}$-$\mathsf{MDDH}$ assumption (i.e., $n \cdot \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}_1}(\lambda)$) and the $2n$ reductions to the collision-resistance of $\mathcal{H}$ (i.e., $2n \cdot \mathbf{Adv}^{\text{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$) *for all $n$ users at one time* (while keeping the reductions to other statistical properties unchanged, namely one user at a time). Specifically,

- we can change all the $kn$ ciphertexts (of all $n$ users) at one time, corresponding to the game transition $\mathsf{G}_1$ to $\mathsf{G}_2$ in the proof of Theorem 2, and the indistinguishability can be reduced to the $\mathcal{U}_{d+k,d}$-$\mathsf{MDDH}$ assumption using Lemma 5 in below;
- we can handle collisions of all users at one time, corresponding to the game transitions $\mathsf{G}_2$ to $\mathsf{G}_3$ and $\mathsf{G}_7$ to $\mathsf{G}_8$ in the proof of Theorem 2.

With this strategy, we obtain a tight reduction with $\mathbf{Adv}^{\text{mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}_1}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$, instead of $n \cdot \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}_1}(\lambda) + 2n \cdot \mathbf{Adv}^{\text{cr}}_{\mathcal{H},\mathcal{B}_2}(\lambda)$, to the computational properties. Thus, the PKE scheme enjoys tight security reduction.

**Lemma 5.** *For any adversary $\mathcal{A}$, any positive integer $d, k, n$, any matrix distribution $\mathcal{D}_{d+k,d}$ and any group generation algorithm $\mathsf{GGen}$, we define the advantage* $\mathbf{Adv}^{(n,k)\text{-mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{A}}(\lambda) :=$

$$\left|\Pr[\mathcal{A}(\mathcal{G}, ([\mathbf{A}_i], [\mathbf{X}_i])_{i=1}^n) = 1] - \Pr[\mathcal{A}(\mathcal{G}, ([\mathbf{A}_i], [\mathbf{X}'_i])_{i=1}^n) = 1]\right|$$

*where $\mathcal{G} \leftarrow_{\$} \mathsf{GGen}(1^\lambda), \mathbf{A}_i \leftarrow_{\$} \mathcal{D}_{d+k,d}, \mathbf{W}_i \leftarrow_{\$} \mathbb{Z}_q^{d\times k}, \mathbf{X}_i := \mathbf{A}_i\mathbf{W}_i$ and $\mathbf{X}'_i \leftarrow_{\$} \mathbb{Z}_q^{(d+k)\times k}$ for all $i \in \{1, \cdots, n\}$. Then, for any PPT adversary $\mathcal{A}$ and uniform matrix distribution $\mathcal{U}_{d+k,d}$, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{(n,k)\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}}(\lambda) + \frac{k+1}{q-1}. \tag{6}$$

We put the proof of Lemma 5 in Appendix C.8.

**Acknowledgment.** We appreciate the anonymous reviewers for their valuable comments.

## References

ABN10. Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *TCC 2010*, pages 480–497. Springer, 2010.

BBDP01. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001*, pages 566–582. Springer, 2001.

BDWY12. Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In *EUROCRYPT 2012*, pages 645–662. Springer, 2012.

BGG+20. Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In *TCC 2020*, pages 260–290. Springer, 2020.

BHK12. Florian Böhl, Dennis Hofheinz, and Daniel Kraschewski. On definitions of selective opening security. In *PKC 2012*, pages 522–539. Springer, 2012.

BS20. Mihir Bellare and Igors Stepanovs. Security under message-derived keys: Signcryption in imessage. In *EUROCRYPT 2020*, pages 507–537, Cham, 2020. Springer.

CFGN96. Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC 1996*, pages 639–648, 1996.

CHK05. Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In *TCC 2005*, pages 150–168. Springer, 2005.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, pages 45–64, 2002.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In *CRYPTO 2013*, pages 129–147. Springer, 2013.

GHKW16. Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In *EUROCRYPT 2016*, pages 1–27. Springer, 2016.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

HJR16. Dennis Hofheinz, Tibor Jager, and Andy Rupp. Public-key encryption with simulation-based selective-opening security and compact ciphertexts. In *TCC 2016*, pages 146–168. Springer, 2016.

HKM+18. 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 2018*, pages 140–159. Springer, 2018.

HLC+19. 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.

HLLG19. Shuai Han, Shengli Liu, Lin Lyu, and Dawu Gu. Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In *CRYPTO 2019*, pages 417–447. Springer, 2019.

HPW15. Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In *ASIACRYPT 2015*, pages 443–469. Springer, 2015.

HT05. Ryotaro Hayashi and Keisuke Tanaka. The sampling twice technique for the RSA-based cryptosystems with anonymity. In *PKC 2005*, pages 216–233. Springer, 2005.

JLL16. Dingding Jia, Xianhui Lu, and Bao Li. Receiver selective opening security from indistinguishability obfuscation. In *INDOCRYPT 2016*, pages 393–410. Springer, 2016.

JLL17. Dingding Jia, Xianhui Lu, and Bao Li. Constructions secure against receiver selective opening and chosen ciphertext attacks. In *CT-RSA 2017*, pages 417–431. Springer, 2017.

LLP20. Youngkyung Lee, Dong Hoon Lee, and Jong Hwan Park. Tightly CCA-secure encryption scheme in a multi-user setting with corruptions. *Designs, Codes and Cryptography*, 88(11):2433–2452, 2020.

Moh10. Payman Mohassel. A closer look at anonymity and robustness in encryption schemes. In *ASIACRYPT 2010*, pages 501–518. Springer, 2010.

QLC15. Baodong Qin, Shengli Liu, and Kefei Chen. Efficient chosen-ciphertext secure public-key encryption scheme with high leakage-resilience. *IET Information Security*, 9(1):32–42, 2015.

WC81. Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

YLH+20. 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 2020*, pages 191–220. Springer, 2020.

# Appendix

## A More Definitions

### A.1 Public Key Encryption Scheme

We recall the formal definition of public key encryption scheme here.

**Definition 15.** *A* public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *consists of four algorithms.*

- $\mathsf{Setup}$. *The PPT setup algorithm on input the security parameter* $1^\lambda$ *outputs a public parameter* $\mathsf{pp}$*, which can be used for multiple users.*
- $\mathsf{Gen}$. *The PPT key generation algorithm on input the public parameter* $\mathsf{pp}$ *outputs a pair of public key and secret key* $(pk, sk)$.
- $\mathsf{Enc}$. *The polynomial time encryption algorithm on input* $\mathsf{pp}$*, a public key* $pk$ *and a message* $m \in \mathcal{M}$*, outputs a ciphertext* $c$.
- $\mathsf{Dec}$. *The deterministic polynomial time decryption algorithm on input* $\mathsf{pp}$*, a secret key* $sk$ *and a ciphertext* $c$*, either outputs a message* $m \in \mathcal{M}$ *or outputs a failure symbol* $\perp$.

$\mathsf{PKE}$ *has* perfect correctness *if* $\mathsf{Dec}(\mathsf{pp}, sk, c) = m$ *for all* $\mathsf{pp} \leftarrow_\$ \mathsf{Setup}(1^\lambda)$*, all* $(pk, sk) \leftarrow_\$ \mathsf{Gen}(\mathsf{pp})$*, all* $m \in \mathcal{M}$ *and all* $c \leftarrow_\$ \mathsf{Enc}(\mathsf{pp}, pk, m)$.

### A.2 Collision-Resistant Hash Function

**Definition 16 (Collision-Resistant Hash Function).** *A family of hash functions* $\mathcal{H}$ *is collision-resistant, if for any PPT adversary* $\mathcal{A}$*, the following advantage is negligible in* $\lambda$

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathcal{H},\mathcal{A}}(\lambda) := \Pr[x_1 \neq x_2 \wedge H(x_1) = H(x_2) | H \leftarrow_\$ \mathcal{H}(1^\lambda), (x_1, x_2) \leftarrow_\$ \mathcal{A}(H)].$$

### A.3 Universal Hash Function and Leftover Hash Lemma

**Definition 17 (Universal Hash Function [WC81]).** *A family of hash functions $\mathcal{H} = \{\mathsf{H} : \mathcal{X} \to \mathcal{Y}\}$ is universal, if for all $x, x' \in \mathcal{X}$ with $x \neq x'$, it holds that*

$$\Pr[\mathsf{H}(x) = \mathsf{H}(x') \mid \mathsf{H} \leftarrow_{\$} \mathcal{H}] \leq \frac{1}{|\mathcal{Y}|}.$$

**Lemma 6 (Leftover Hash Lemma [HILL99]).** *Let $\mathcal{H} = \{\mathsf{H} : \mathcal{X} \to \mathcal{Y}\}$ be a family of universal hash functions. Then it holds that*

$$\Delta((\mathsf{H}, \mathsf{H}(X)), (\mathsf{H}, Y)) \leq \frac{1}{2} \cdot \sqrt{\frac{|\mathcal{Y}|}{|\mathcal{X}|}}$$

*where $\mathsf{H} \leftarrow_{\$} \mathcal{H}$, $X \leftarrow_{\$} \mathcal{X}$, $Y \leftarrow_{\$} \mathcal{Y}$ and $\Delta(\cdot)$ denotes the statistical distance.*

## B Some security notions

We recall the notion of simulation-based receiver selective opening security in the multi-challenge setting [YLH+20] as follows. Here we recall the version that the adversary is granted multiple opening queries. Without loss of generality, we implicitly assume that all the messages are the same size.

**Definition 18. (SIM-RSO$_k$-CPA/CCA).** *A PKE scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* SIM-RSO$_k$-ATK secure *(where* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$*), if for any polynomially bounded $n > 0$, and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there is a PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, such that for any PPT distinguisher $\mathcal{D}$, the advantage* $\mathbf{Adv}^{\text{sim-rso-atk}}_{\mathsf{PKE},\mathcal{A},\mathcal{S},\mathcal{D},n,k}(\lambda) :=$

$$|\Pr[\mathcal{D}(\mathbf{Exp}^{\text{sim-rso-atk-real}}_{\mathsf{PKE},\mathcal{A},n,k}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathbf{Exp}^{\text{sim-rso-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,k}(\lambda)) = 1]|$$

*is negligible, where* $\mathbf{Exp}^{\text{sim-rso-atk-real}}_{\mathsf{PKE},\mathcal{A},n,k}(\lambda)$ *and* $\mathbf{Exp}^{\text{sim-rso-atk-ideal}}_{\mathsf{PKE},\mathcal{S},n,k}(\lambda)$ *are defined in Fig. 11, and* $\mathrm{atk} \in \{\mathrm{cpa}, \mathrm{cca}\}$*. In both of the experiments, we require that for all* $\mathsf{Dist}_{\mathrm{m}}$ *output by* $\mathcal{A}_1$ *and* $\mathcal{S}_1$*,* $\mathsf{Dist}_{\mathrm{m}}$ *is efficiently samplable.*

When $k = 1$, SIM-RSO$_k$-CPA/CCA security is standard simulation-based receiver selective opening (SIM-RSO-CPA/CCA) security [HPW15, HKM+18, HLC+19].

Next, we recall the ANON-COR security [BGG+20], as follows.

**Definition 19. (ANON-COR).** *A PKE scheme* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* anonymous under selective opening attacks (ANON-COR secure)*, if for any $\epsilon > 0$, any $\lambda < \{t, Q\} < n(1-\epsilon)$ and any PPT adversary $\mathcal{A}$, $\mathcal{A}$ wins the following game with only negligible probability:*

*(1) The challenger runs* $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$*, generates* $((pk_i, sk_i) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp}))_{i\in[n]}$*, and sends* $(\mathsf{pp}, (pk_i)_{i\in[n]})$ *to* $\mathcal{A}$*.*
*(2)* $\mathcal{A}$ *sends* $t$ *messages* $(m_1^*, \cdots, m_t^*)$ *to the challenger.*

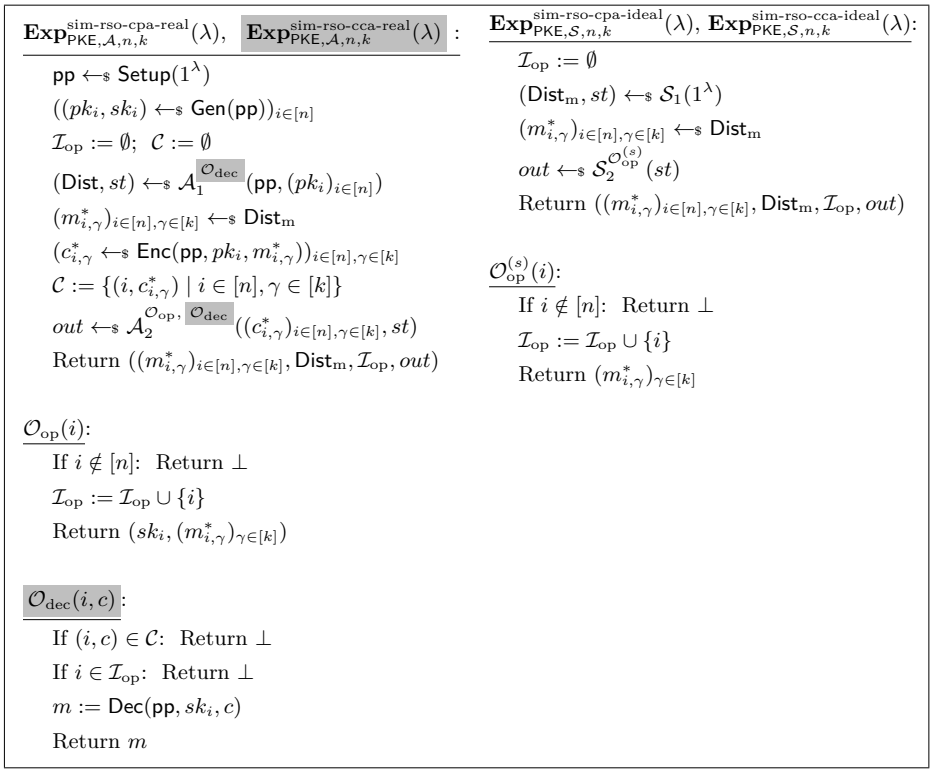


**Fig. 11** Experiments for defining SIM-RSO$_k$-CPA/CCA security of a PKE scheme $\mathsf{PKE}$.

*(3) The challenger randomly samples $t$ distinct indexes $\{i_1, \cdots, i_t\} \subset [n]$, computes $(c^*_j \leftarrow \mathsf{Enc}(\mathsf{pp}, pk_{i_j}, m^*_j))_{j\in[t]}$, and sends $(c^*_j)_{j\in[t]}$ to $\mathcal{A}$.*
*(4) $\mathcal{A}$ adaptively chooses indexes $i'_1, \cdots, i'_Q \in [n]$ one at a times, and receives $sk_{i'_\ell}$ for each $i'_\ell$ ($\ell \in [Q]$).*
*(5) $\mathcal{A}$ wins the game if he opens more than $\frac{Q}{n} + \epsilon$ fraction of the ciphertext-encrypting keys indexed by $\mathcal{A}$.*

# C Omitted Proofs

## C.1 Proof of Lemma 1

*Proof of Lemma 1.* We construct a KMNC$_k$-CCA adversary $\mathcal{B}$ against scheme $\mathsf{KM\text{-}NCE}$ as follows.

On input of $(\mathsf{pp}, pk)$, $\mathcal{B}$ sets $pk_{\widehat{i}} := pk$ and maintains five sets $\mathcal{I}_{\text{op}}$, $\mathcal{I}_{\text{cor}}$, $\mathcal{I}_{\text{op-sk}}$, $\mathcal{I}_{\text{cor-sk}}$, $\mathcal{C}$, which are initially empty. For all indices $i \in \{1, \ldots, n\} \setminus \{\widehat{i}\}$, $\mathcal{B}$ generates $(pk_i, sk_i, tk_i) \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp})$. It finally sends $(\mathsf{pp}, (pk_i)_{i\in[n]})$ to $\mathcal{A}_1$. $\mathcal{A}_1$ adaptively issues $\mathcal{O}_{\text{cor},1}, \mathcal{O}_{\text{dec}}$ oracle queries and $\mathcal{B}$ answers the queries as follows.

- On query $\mathcal{O}_{\text{cor},1}(i)$ where $i \in [n]$, if $i = \widehat{i}$, $\mathcal{B}$ aborts and outputs a random bit as $\mathcal{B}$'s final output; otherwise $\mathcal{B}$ returns $sk_i$ and sets $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}$, $\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk_i)\}$.
- On query $\mathcal{O}_{\text{dec}}(i, c)$, if $i \in \mathcal{I}_{\text{cor}}$, $\mathcal{B}$ returns $\perp$; else if $i = \widehat{i}$, $\mathcal{B}$ invokes its own decryption oracle on $c$; else (i.e., $i \neq \widehat{i}$), $\mathcal{B}$ returns $\mathsf{Dec}(\mathsf{pp}, sk_i, c)$.

In the challenge phase, $\mathcal{A}_1$ submits a distribution $\mathsf{Dist}$. $\mathcal{B}$ first samples $(i_j, (m^*_{j,\gamma})_{\gamma\in[k]})_{j\in[t]} \leftarrow_{\$} \mathsf{Dist}$. If there does not exist any $j' \in [t]$ such that $i_{j'} = \widehat{i}$, $\mathcal{B}$ aborts and outputs a random bit; else (i.e., there is some $j' \in [t]$ such that $i_{j'} = \widehat{i}$), $\mathcal{B}$ returns $(m^*_\gamma)_{\gamma\in[k]} := (m^*_{j',\gamma})_{\gamma\in[k]}$ to its KMNC$_k$-CCA challenger.

Receiving the challenge ciphertexts $(c^*_\gamma)_{\gamma\in[k]}$ from the KMNC$_k$-CCA challenger, $\mathcal{B}$ generates the challenge ciphertexts for $\mathcal{A}$ as follows. For each $j \in [t]$, if $(i_j \in \mathcal{I}_{\text{cor}}) \vee (n \geq i_j > \widehat{i})$, $\mathcal{B}$ computes $(c^*_{j,\gamma} \leftarrow_{\$} \mathsf{Enc}(\mathsf{pp}, pk_{i_j}, m^*_{j,\gamma}))_{\gamma\in[k]}$; else if $i_j < \widehat{i}$, $\mathcal{B}$ computes $((c^*_{j,\gamma}, td^*_{j,\gamma}) \leftarrow_{\$} \mathsf{Fake}(\mathsf{pp}))_{\gamma\in[k]}$; else (i.e., $i_j = \widehat{i}$), $\mathcal{B}$ sets $(c^*_{j,\gamma})_{\gamma\in[k]} := (c^*_\gamma)_{\gamma\in[k]}$. Finally, $\mathcal{B}$ sends $(c^*_{j,\gamma})_{j\in[t],\gamma\in[k]}$ to $\mathcal{A}_2$ and sets $\mathcal{C} = \{(i_j, c^*_{j,\gamma}) \mid j \in [t], \gamma \in [k]\}$.

After receiving the challenge ciphertexts $(c^*_{j,\gamma})_{j\in[t],\gamma\in[k]}$, $\mathcal{A}_2$ adaptively issues $\mathcal{O}_{\text{cor},2}, \mathcal{O}_{\text{op}}, \mathcal{O}_{\text{dec}}$ oracle queries and $\mathcal{B}$ answers the queries as follows.

- On query $\mathcal{O}_{\text{cor},2}(i)$ where $i \in [n]$, if there is some $j' \in [t]$ such that $i_{j'} = i$, $\mathcal{B}$ proceeds as follows:
  1. If $j' \in \mathcal{I}_{\text{op}}$, then there must be some tuple $(j', i_{j'} = i, \overline{sk}_i) \in \mathcal{I}_{\text{op-sk}}$, and $\mathcal{B}$ returns $\overline{sk}_i$ to $\mathcal{A}_2$, and sets $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}, \mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, \overline{sk}_i)\}$.
  2. Else if $(j' \notin \mathcal{I}_{\text{op}}) \wedge (i = \widehat{i})$, $\mathcal{B}$ sets $\overline{sk}_i = sk'$ and $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}$, $\mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, \overline{sk}_i)\}$, where $sk'$ is $\mathcal{B}$'s challenge secret key obtained from its KMNC$_k$-CCA challenger, and returns $\overline{sk}_i$ to $\mathcal{A}_2$.
  3. Else if $(j' \notin \mathcal{I}_{\text{op}}) \wedge (i < \widehat{i})$, $\mathcal{B}$ returns $\overline{sk}_i \leftarrow_{\$} \mathsf{Open}_k(\mathsf{pp}, tk_i, pk_i, sk_i, (c^*_{j',\gamma}, td^*_{j',\gamma}, m^*_{j',\gamma})_{\gamma\in[k]})$ to $\mathcal{A}_2$, and sets $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}, \mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, \overline{sk}_i)\}$.
  4. Else (i.e., $(j' \notin \mathcal{I}_{\text{op}}) \wedge (i > \widehat{i})$), $\mathcal{B}$ returns $sk_i$ to $\mathcal{A}_2$, and sets $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}, \mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk_i)\}$.

  Otherwise (i.e., there does not exist any $j' \in [t]$ satisfying $i_{j'} = i$), $\mathcal{B}$ proceeds as follows:
  1. If $i = \widehat{i}$, $\mathcal{B}$ aborts and outputs a random bit.
  2. Else (i.e., $i \neq \widehat{i}$), $\mathcal{B}$ returns $sk_i$ and sets $\mathcal{I}_{\text{cor}} := \mathcal{I}_{\text{cor}} \cup \{i\}, \mathcal{I}_{\text{cor-sk}} := \mathcal{I}_{\text{cor-sk}} \cup \{(i, sk_i)\}$.
- On query $\mathcal{O}_{\text{op}}(j)$ where $j \in [t]$, $\mathcal{B}$ firstly sets $\mathcal{I}_{\text{op}} := \mathcal{I}_{\text{op}} \cup \{j\}$. If $i_j \in \mathcal{I}_{\text{cor}}$, there must be some tuple $(i_j, \overline{sk}_{i_j}) \in \mathcal{I}_{\text{cor-sk}}$, and $\mathcal{B}$ returns $\overline{sk}_{i_j}$ to $\mathcal{A}_2$; otherwise (i.e., $i_j \notin \mathcal{I}_{\text{cor}}$), $\mathcal{B}$ proceeds as follows:
  1. If $i_j = \widehat{i}$, $\mathcal{B}$ sets $\overline{sk}_{i_j} := sk'$ and $\mathcal{I}_{\text{op-sk}} := \mathcal{I}_{\text{op-sk}} \cup \{(j, i_j, \overline{sk}_{i_j})\}$, where $sk'$ is $\mathcal{B}$'s challenge secret key obtained from its KMNC$_k$-CCA challenger, and returns $\overline{sk}_{i_j}$ to $\mathcal{A}_2$.
  2. Else if $i_j < \widehat{i}$, $\mathcal{B}$ returns $\overline{sk}_{i_j} \leftarrow_{\$} \mathsf{Open}_k(\mathsf{pp}, tk_{i_j}, pk_{i_j}, sk_{i_j}, (c^*_{j,\gamma}, td^*_{j,\gamma}, m^*_{j,\gamma})_{\gamma\in[k]})$ to $\mathcal{A}_2$, and sets $\mathcal{I}_{\text{op-sk}} := \mathcal{I}_{\text{op-sk}} \cup \{(j, i_j, \overline{sk}_{i_j})\}$.

3. Else (i.e., $i_j > \widehat{i}$), $\mathcal{B}$ returns $sk_{i_j}$ to $\mathcal{A}_2$, and sets $\mathcal{I}_{\text{op-sk}} := \mathcal{I}_{\text{op-sk}} \cup \{(j, i_j, sk_{i_j})\}$.

– On query $\mathcal{O}_{\text{dec}}(i, c)$, if $(i \in \mathcal{I}_{\text{cor}}) \vee (\exists j' \in \mathcal{I}_{\text{op}}$ s.t. $i = i_{j'}) \vee ((i, c) \in \mathcal{C})$, it returns $\perp$; else if $i = \widehat{i}$, $\mathcal{B}$ invokes its own decryption oracle on $c$; else (i.e., $i \neq \widehat{i}$), $\mathcal{B}$ returns $\mathsf{Dec}(\mathsf{pp}, sk_i, c)$.

At last, receiving $\mathcal{A}_2$'s final output $out$, $\mathcal{B}$ computes $b' \leftarrow\!\!\$\ \mathcal{D}((i_j, (m^*_{j,\gamma})_{\gamma\in[k]})_{j\in[t]}, \mathsf{Dist}, \mathcal{I}_{\text{op}}, \mathcal{I}_{\text{cor}}, out)$, and returns $b'$ as its own final output.

That's the construction of the KMNC$_k$-CCA adversary $\mathcal{B}$. Now we compute $\mathcal{B}$'s advantage.

We say that an event $\mathsf{Evt}_{\widehat{i}}$ occurs, if (i) for $\mathcal{A}_1$'s each $\mathcal{O}_{\text{cor},1}$ oracle query $i \in [n]$, $i \neq \widehat{i}$, and (ii) for the sampled tuple $(i_j, (m^*_{j,\gamma})_{\gamma\in[k]})_{j\in[t]}$, there is some $j \in [t]$ such that $i_j = \widehat{i}$. Note that if $\mathsf{Evt}_{\widehat{i}}$ does not occur, $\mathcal{B}$ will return a random bit as its final output.

For simplicity, we use $\mathbf{E}^{\text{real}}_{\mathcal{B}}$ (resp. $\mathbf{E}^{\text{sim}}_{\mathcal{B}}$) to denote $\mathbf{Exp}^{\text{kmnc-cca-real}}_{\mathsf{KM\text{-}NCE},\mathcal{B},k}(\lambda)$ (resp. $\mathbf{Exp}^{\text{kmnc-cca-sim}}_{\mathsf{KM\text{-}NCE},\mathcal{B},k}(\lambda)$). Thus, we obtain that

$$
\begin{aligned}
&\mathbf{Adv}^{\text{kmnc-cca}}_{\mathsf{KM\text{-}NCE},\mathcal{B},k}(\lambda) \\
&= |\Pr[\mathbf{E}^{\text{real}}_{\mathcal{B}} = 1] - \Pr[\mathbf{E}^{\text{sim}}_{\mathcal{B}} = 1]| && (7)\\
&= |(\Pr[\mathbf{E}^{\text{real}}_{\mathcal{B}} = 1|\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\mathsf{Evt}_{\widehat{i}}] + \Pr[\mathbf{E}^{\text{real}}_{\mathcal{B}} = 1|\neg\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\neg\mathsf{Evt}_{\widehat{i}}]) \\
&\quad -(\Pr[\mathbf{E}^{\text{sim}}_{\mathcal{B}} = 1|\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\mathsf{Evt}_{\widehat{i}}] + \Pr[\mathbf{E}^{\text{sim}}_{\mathcal{B}} = 1|\neg\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\neg\mathsf{Evt}_{\widehat{i}}])| && (8)\\
&= |\Pr[\mathbf{E}^{\text{real}}_{\mathcal{B}} = 1|\mathsf{Evt}_{\widehat{i}}] - \Pr[\mathbf{E}^{\text{sim}}_{\mathcal{B}} = 1|\mathsf{Evt}_{\widehat{i}}]| \cdot \Pr[\mathsf{Evt}_{\widehat{i}}] \ . && (9)
\end{aligned}
$$

Eq. (9) is derived because $\Pr[\mathbf{E}^{\text{real}}_{\mathcal{B}} = 1|\neg\mathsf{Evt}_{\widehat{i}}] = \Pr[\mathbf{E}^{\text{sim}}_{\mathcal{B}} = 1|\neg\mathsf{Evt}_{\widehat{i}}] = \frac{1}{2}$.

On the other hand, recalling the descriptions of **Game** $\mathsf{G}_{\widehat{i}}$ and **Game** $\mathsf{G}_{\widehat{i}-1}$, we notice that when $\mathsf{Evt}_{\widehat{i}}$ does not occur, $\mathsf{G}_{\widehat{i}}$ and $\mathsf{G}_{\widehat{i}-1}$ are identical from $\mathcal{A}$'s point of view. As a result,

$$\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}}) = 1|\neg\mathsf{Evt}_{\widehat{i}}] = \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1}) = 1|\neg\mathsf{Evt}_{\widehat{i}}] \ .$$

Hence, we obtain that

$$
\begin{aligned}
&|\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}}) = 1] - \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1}) = 1]| \\
&= |(\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}}) = 1|\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\mathsf{Evt}_{\widehat{i}}] + \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}}) = 1|\neg\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\neg\mathsf{Evt}_{\widehat{i}}]) \\
&\quad -(\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1}) = 1|\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\mathsf{Evt}_{\widehat{i}}] + \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1}) = 1|\neg\mathsf{Evt}_{\widehat{i}}] \cdot \Pr[\neg\mathsf{Evt}_{\widehat{i}}])| \\
&= |\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}}) = 1|\mathsf{Evt}_{\widehat{i}}] - \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1}) = 1|\mathsf{Evt}_{\widehat{i}}]| \cdot \Pr[\mathsf{Evt}_{\widehat{i}}] \ . && (10)
\end{aligned}
$$

Observe that, $\mathcal{B}$ does not abort (and return a random bit) if and only if $\mathsf{Evt}_{\widehat{i}}$ occurs. If $\mathcal{B}$ does not abort, it perfectly simulates game $\mathsf{G}_{\widehat{i}}$ or $\mathsf{G}_{\widehat{i}-1}$ for $\mathcal{A}$ and $\mathcal{D}$, and $\mathcal{B}$'s final output is $\mathcal{D}$'s output, i.e.,

$$
\begin{aligned}
\Pr[\mathbf{E}^{\text{real}}_{\mathcal{B}} = 1|\mathsf{Evt}_{\widehat{i}}] &= \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1}) = 1|\mathsf{Evt}_{\widehat{i}}] \ , && (11)\\
\Pr[\mathbf{E}^{\text{sim}}_{\mathcal{B}} = 1|\mathsf{Evt}_{\widehat{i}}] &= \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}}) = 1|\mathsf{Evt}_{\widehat{i}}] \ . && (12)
\end{aligned}
$$

Combining Eqs. (10)-(12), we obtain that

$$\begin{aligned} &|\Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}}) = 1] - \Pr[\mathcal{D}(\mathsf{G}_{\widehat{i}-1}) = 1]| \\ =\ &|\Pr[\mathbf{E}_{\mathcal{B}}^{\mathrm{sim}} = 1|\mathsf{Evt}_{\widehat{i}}] - \Pr[\mathbf{E}_{\mathcal{B}}^{\mathrm{real}} = 1|\mathsf{Evt}_{\widehat{i}}]| \cdot \Pr[\mathsf{Evt}_{\widehat{i}}]. \end{aligned} \tag{13}$$

Hence, combining Eqs. (9) and (13), we finish the proof of Lemma 1. □

### C.2 Proof of Lemma 2

*Proof of Lemma 2.* Observe that, the only differences between $\mathsf{G}_{n+\widehat{i}}$ and $\mathsf{G}_{n+\widehat{i}-1}$ is that when $\mathcal{A}_2$ issues an $\mathcal{O}_{\mathrm{dec}}$ oracle query $(i, c)$ satisfying $(\exists(i_j, c^*_{j,\gamma}) \in \mathcal{C}$ s.t. $i_j = \widehat{i} \wedge c^*_{j,\gamma} = c \wedge i \neq \widehat{i}) \wedge (i \notin \mathcal{I}_{\mathrm{cor}})$, the challenger returns $\perp$ to $\mathcal{A}_2$ in game $\mathsf{G}_{n+\widehat{i}}$, and the challenger returns $\mathsf{Dec}(\mathsf{pp}, sk_i, c := c^*_{j,\gamma})$ to $\mathcal{A}_2$ in game $\mathsf{G}_{n+\widehat{i}-1}$. Since $\mathsf{KM\text{-}NCE}$ is robust and all the challenge ciphertexts are generated with $\mathsf{Fake}$ in $\mathsf{G}_{n+\widehat{i}-1}$, for any $j' \in [t]$ and any $\gamma' \in [k]$, $\mathsf{Dec}(\mathsf{pp}, sk_i, c^*_{j',\gamma'})$ returns $\perp$ with overwhelming probability. So we obtain that $|\Pr[\mathcal{D}(\mathsf{G}_{n+\widehat{i}}) = 1] - \Pr[\mathcal{D}(\mathsf{G}_{n+\widehat{i}-1}) = 1]| \leq t \cdot k \cdot \epsilon^{\mathrm{rob}}_{\mathsf{KM\text{-}NCE}}(\lambda)$. □

### C.3 Full description of $\mathcal{B}_3$ in the Proof of Theorem 2

Let $\mathsf{Forge}$ denote the event that $\mathcal{A}$ ever queries $\mathcal{O}_{\mathrm{dec}}(c)$ with $c = (x, d, \widetilde{\pi})$, such that $\tau \notin \{\tau^*_\gamma\}_{\gamma\in[k]}$, $\widetilde{\pi} = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$ but $x \in \mathcal{X} \setminus \mathcal{L}$. Clearly, $\mathsf{G}_3$ and $\mathsf{G}_4$ are identical unless $\mathsf{Forge}$ occurs, thus by the difference lemma, $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \Pr_4[\mathsf{Forge}]$.

To bound $\Pr_4[\mathsf{Forge}]$, we construct an unbounded adversary $\mathcal{B}_3$ against the universal$_{k+1}$ property of tag-based $\widetilde{\mathsf{HPS}}$ as follows. $\mathcal{B}_3$ will simulate $\mathsf{G}_4$ for $\mathcal{A}$.

- In the beginning, $\mathcal{B}_3$ is given $(\widetilde{\mathsf{mpar}}, \widetilde{\mathsf{par}}, \widetilde{\mathsf{pk}}, (x^*_\gamma)_{\gamma\in[k]})$, where $\widetilde{\mathsf{mpar}} \leftarrow_{\$} \widetilde{\mathsf{MPar}}(1^\lambda)$, $(\widetilde{\mathsf{par}}, \widetilde{\mathsf{td}}) \leftarrow_{\$} \widetilde{\mathsf{Par}}(\widetilde{\mathsf{mpar}})$, $\widetilde{\mathsf{sk}} \leftarrow_{\$} \widetilde{\mathcal{SK}}$, $\widetilde{\mathsf{pk}} := \widetilde{\alpha}(\widetilde{\mathsf{sk}})$, and $x^*_\gamma \leftarrow_{\$} \mathcal{X}$ for each $\gamma \in [k]$.
- $\mathcal{B}_3$ invokes $\mathsf{mpar} \leftarrow_{\$} \mathsf{MPar}(1^\lambda)$, samples $H \leftarrow_{\$} \mathcal{H}$, and sets $\mathsf{pp} := (\mathsf{mpar}, \widetilde{\mathsf{mpar}}, H)$. It then invokes $(\mathsf{par}, \mathsf{td}) \leftarrow_{\$} \mathsf{Par}(\mathsf{mpar})$, samples $\mathsf{sk} \leftarrow_{\$} \mathcal{SK}$, computes $\mathsf{pk} := \alpha(\mathsf{sk})$, and sets $pk := (\mathsf{par}, \widetilde{\mathsf{par}}, \mathsf{pk}, \widetilde{\mathsf{pk}})$. $\mathcal{B}_3$ also computes an $\widetilde{\mathsf{sk}}^* \in \widetilde{\mathcal{SK}}$ such that $\widetilde{\mathsf{pk}} = \widetilde{\alpha}(\widetilde{\mathsf{sk}}^*)$.
- $\mathcal{B}_3$ forwards $(\mathsf{pp}, pk)$ to $\mathcal{A}$, and simulates $\mathcal{O}_{\mathrm{dec}}$ oracle for $\mathcal{A}$ as follows. For $\mathcal{O}_{\mathrm{dec}}(c)$ query with $c = (x, d, \widetilde{\pi})$, $\mathcal{B}_3$ computes $\tau := H(x, d)$, and outputs $\perp$ immediately if $c \in \{c^*_\gamma\}_{\gamma\in[k]}$ or $\tau \in \{\tau^*_\gamma\}_{\gamma\in[k]}$ or $x \in \mathcal{X} \setminus \mathcal{L}$ or $\widetilde{\pi} \neq \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}^*}(x, \tau)$ using $\widetilde{\mathsf{sk}}^*$. Otherwise, it computes $m := d - \Lambda_{\mathsf{sk}}(x) \in \Pi$, and returns $m$ to $\mathcal{A}$.
- $\mathcal{B}_3$ receives $(m^*_\gamma)_{\gamma\in[k]}$ from $\mathcal{A}$, and prepares $(c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma))_{\gamma\in[k]}$ as follows. For each $\gamma \in [k]$, $\mathcal{B}_3$ sets $x^*_\gamma$ as the one in its own challenge, and computes $d^*_\gamma := \Lambda_{\mathsf{sk}}(x^*_\gamma) + m^*_\gamma$ and $\tau^*_\gamma := H(x^*_\gamma, d^*_\gamma)$. Then $\mathcal{B}_3$ returns $(\tau^*_\gamma)_{\gamma\in[k]}$ to its own challenger, and receives $(\widetilde{\pi}^*_\gamma)_{\gamma\in[k]}$ from its challenger, where $\widetilde{\pi}^*_\gamma := \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x^*_\gamma, \tau^*_\gamma)$. For each $\gamma \in [k]$, $\mathcal{B}_3$ sets $\widetilde{\pi}^*_\gamma$ as the one it received from its own challenge.

- $\mathcal{B}_3$ sends $(c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma))_{\gamma\in[k]}$ to $\mathcal{A}$, and continues the simulation of $\mathcal{O}_{\mathrm{dec}}$ oracle for $\mathcal{A}$ as described above.
- Finally, when $\mathcal{A}$ finishes all $\mathcal{O}_{\mathrm{dec}}$ queries and asks for a secret key $sk = (\mathsf{sk}, \widetilde{\mathsf{sk}})$, $\mathcal{B}_3$ terminates the interaction with $\mathcal{A}$. Then $\mathcal{B}_3$ randomly picks one $\mathcal{O}_{\mathrm{dec}}$ query $c = (x, d, \widetilde{\pi})$ made by $\mathcal{A}$ (from all the $Q_d$ queries), computes $\tau := H(x, d)$, and outputs $(x, \tau, \widetilde{\pi})$ to its own challenger.

Firstly, we claim that $\mathcal{B}_3$ perfectly simulates $\mathsf{G}_4$ for $\mathcal{A}$ except the very last secret key query.

- Note that $\mathcal{B}_3$'s simulation of oracle $\mathcal{O}_{\mathrm{dec}}$ (using $\widetilde{\mathsf{sk}}^*$) is identical to the $\mathcal{O}_{\mathrm{dec}}$ in $\mathsf{G}_4$ (using $\widetilde{\mathsf{sk}}$). The reason is, due to the new rejection rule added in $\mathsf{G}_4$, $\mathcal{O}_{\mathrm{dec}}$ outputs $\perp$ immediately if $x \in \mathcal{X} \setminus \mathcal{L}$, and for $x \in \mathcal{L}$, $\widetilde{\Lambda}_{\widetilde{\mathsf{sk}}^*}(x, \tau) = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$ always holds by the perfect correctness of $\widetilde{\mathsf{HPS}}$.
- $\mathcal{B}_3$'s computation of $(c^*_\gamma = (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma))_{\gamma\in[k]}$ is identical to that in $\mathsf{G}_4$.

Then we analyze the advantage of $\mathcal{B}_3$. It is clear to see that $\mathcal{B}_3$'s output $(x, \tau, \widetilde{\pi})$ succeeds, i.e., satisfying $x \in \mathcal{X} \setminus \mathcal{L} \wedge \tau \notin \{\tau^*_\gamma\}_{\gamma\in[k]} \wedge \widetilde{\pi} = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$, as long as $\mathsf{Forge}$ occurs and exactly occurs in the $\mathcal{O}_{\mathrm{dec}}$ query chosen by $\mathcal{B}_3$. Consequently, we get that $\boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda) \geq \frac{1}{Q_d} \cdot \Pr_4[\mathsf{Forge}]$.

Overall, $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \Pr_4[\mathsf{Forge}] \leq Q_d \cdot \boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{B}_3}(\lambda)$.

### C.4 Proof of Lemma 3

**Lemma 3** ($\mathsf{G}_4 \to \mathsf{G}_5$) *There exists an unbounded $\mathcal{B}_4$ against the openable$_k$ property of $\mathsf{HPS}$, such that $|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq 2 \cdot \boldsymbol{\epsilon}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{B}_4}(\lambda) + 2k \cdot \boldsymbol{\epsilon}^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda)$.*

*Proof.* We construct an unbounded $\mathcal{B}_4$ by invoking $\mathcal{A}$, as shown below.

- In the beginning, $\mathcal{B}_4$ is given $(\mathsf{mpar}, \mathsf{par}, \mathsf{td}, \mathsf{pk}, (x^*_\gamma, r_{x^*_\gamma}, \pi^{*(b)}_\gamma)_{\gamma\in[k]}, \mathsf{sk}^{(b)})$, where $\mathsf{mpar} \leftarrow_{\$} \mathsf{MPar}(1^\lambda)$, $(\mathsf{par}, \mathsf{td}) \leftarrow_{\$} \mathsf{Par}(\mathsf{mpar})$, $\mathsf{sk} \leftarrow_{\$} \mathcal{SK}$, $\mathsf{pk} := \alpha(\mathsf{sk})$, $x^*_\gamma \leftarrow_{\$} \mathcal{X}$ with randomness $r_{x^*_\gamma}$, $\pi^{*(0)}_\gamma := \Lambda_{\mathsf{sk}}(x^*_\gamma)$, $\mathsf{sk}^{(0)} := \mathsf{sk}$, $\pi^{*(1)}_\gamma \leftarrow_{\$} \Pi$ with randomness $r_{\pi^{*(1)}_\gamma}$, $\mathsf{sk}^{(1)} \leftarrow_{\$} \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x^*_\gamma, r_{x^*_\gamma}, \pi^{*(1)}_\gamma, r_{\pi^{*(1)}_\gamma})_{\gamma\in[k]})$, and $b \leftarrow_{\$} \{0, 1\}$. $\mathcal{B}_4$ is asked to guess the value of $b$.
- $\mathcal{B}_4$ invokes $\widetilde{\mathsf{mpar}} \leftarrow_{\$} \widetilde{\mathsf{MPar}}(1^\lambda)$, samples $H \leftarrow_{\$} \mathcal{H}$, and sets $\mathsf{pp} := (\mathsf{mpar}, \widetilde{\mathsf{mpar}}, H)$. It then invokes $(\widetilde{\mathsf{par}}, \widetilde{\mathsf{td}}) \leftarrow_{\$} \widetilde{\mathsf{Par}}(\widetilde{\mathsf{mpar}})$, samples $\widetilde{\mathsf{sk}} \leftarrow_{\$} \widetilde{\mathcal{SK}}$, computes $\widetilde{\mathsf{pk}} := \widetilde{\alpha}(\widetilde{\mathsf{sk}})$, and sets $pk := (\mathsf{par}, \widetilde{\mathsf{par}}, \mathsf{pk}, \widetilde{\mathsf{pk}})$. $\mathcal{B}_4$ also computes an $\mathsf{sk}^* \in \mathcal{SK}$ such that $\mathsf{pk} = \alpha(\mathsf{sk}^*)$.
- $\mathcal{B}_4$ forwards $(\mathsf{pp}, pk)$ to $\mathcal{A}$, and simulates $\mathcal{O}_{\mathrm{dec}}$ oracle for $\mathcal{A}$ as follows. For $\mathcal{O}_{\mathrm{dec}}(c)$ query with $c = (x, d, \widetilde{\pi})$, $\mathcal{B}_4$ computes $\tau := H(x, d)$, and outputs $\perp$ immediately if $c \in \{c^*_\gamma\}_{\gamma\in[k]}$ or $\tau \in \{\tau^*_\gamma\}_{\gamma\in[k]}$ or $x \in \mathcal{X} \setminus \mathcal{L}$ or $\widetilde{\pi} \neq \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$. Otherwise, it computes $m := d - \Lambda_{\mathsf{sk}^*}(x) \in \Pi$ using $\mathsf{sk}^*$, and returns $m$ to $\mathcal{A}$.

- $\mathcal{B}_4$ receives $(m^*_\gamma)_{\gamma\in[k]}$ from $\mathcal{A}$, and prepares $(c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma))_{\gamma\in[k]}$ as follows. For each $\gamma \in [k]$, $\mathcal{B}_4$ sets $x^*_\gamma$ as the one in its own challenge, computes $d^*_\gamma := \pi^{*(b)}_\gamma + m^*_\gamma$ using its own challenge $\pi^{*(b)}_\gamma$, $\tau^*_\gamma := H(x^*_\gamma, d^*_\gamma)$, and $\widetilde{\pi}^*_\gamma := \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x^*_\gamma, \tau^*_\gamma)$.
- $\mathcal{B}_4$ sends $(c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma))_{\gamma\in[k]}$ to $\mathcal{A}$, and continues the simulation of $\mathcal{O}_{\mathrm{dec}}$ oracle for $\mathcal{A}$ as described above.
- Finally, $\mathcal{B}_4$ sends $(\mathsf{sk}^{(b)}, \widetilde{\mathsf{sk}})$ to $\mathcal{A}$, and receives a bit $b'$ from $\mathcal{A}$. $\mathcal{B}_4$ sends $b'$ to its own challenger.

We analyze the advantage of $\mathcal{B}_4$. Firstly, we note that $\mathcal{B}_4$'s simulation of oracle $\mathcal{O}_{\mathrm{dec}}$ (using $\mathsf{sk}^*$) is identical to the $\mathcal{O}_{\mathrm{dec}}$ in $\mathsf{G}_4$ and $\mathsf{G}_5$ (using $\mathsf{sk}$). The reason is, due to the new rejection rule added in $\mathsf{G}_4$, $\mathcal{O}_{\mathrm{dec}}$ will not output $m$ unless $x \in \mathcal{L}$, and for $x \in \mathcal{L}$, $\Lambda_{\mathsf{sk}^*}(x) = \Lambda_{\mathsf{sk}}(x)$ always holds by the perfect correctness of $\mathsf{HPS}$. Next, we claim that $\mathcal{B}_4$ perfectly simulates $\mathsf{G}_4$ when $b = 0$, and simulates $\mathsf{G}_5$ with a statistical distance $k \cdot \epsilon^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda)$ when $b = 1$.

- In the case $b = 0$, for each $\gamma \in [k]$, $d^*_\gamma := \pi^{*(0)}_\gamma + m^*_\gamma = \Lambda_{\mathsf{sk}}(x^*_\gamma) + m^*_\gamma$, and in the last step, $\mathcal{B}_4$ sends $(\mathsf{sk}^{(0)} = \mathsf{sk}, \widetilde{\mathsf{sk}})$. This is the same as $\mathsf{G}_4$.
- In the case $b = 1$, for each $\gamma \in [k]$, $d^*_\gamma := \pi^{*(1)}_\gamma + m^*_\gamma$ where $\pi^{*(1)}_\gamma \leftarrow_{\$} \Pi$ with randomness $r_{\pi^{*(1)}_\gamma}$, so $d^*_\gamma$ is uniformly distributed over $\Pi$, the same as $\mathsf{G}_5$. Moreover, in the last step, $\mathcal{B}_4$ sends $(\mathsf{sk}^{(1)}, \widetilde{\mathsf{sk}})$ to $\mathcal{A}$, where $\mathsf{sk}^{(1)} \leftarrow_{\$} \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x^*_\gamma, r_{x^*_\gamma}, \pi^{*(1)}_\gamma, r_{\pi^{*(1)}_\gamma})_{\gamma\in[k]})$.

  Note that in $\mathsf{G}_5$, the challenger would compute $e^*_\gamma := d^*_\gamma - m^*_\gamma = \pi^{*(1)}_\gamma$ and resample $r_{e^*_\gamma} \leftarrow_{\$} \mathsf{ReSmp}_\Pi(e^*_\gamma)$ for each $\gamma \in [k]$, invoke $\mathsf{sk}' \leftarrow_{\$} \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x^*_\gamma, r_{x^*_\gamma}, e^*_\gamma, r_{e^*_\gamma})_{\gamma\in[k]})$, and send $(\mathsf{sk}', \widetilde{\mathsf{sk}})$ to $\mathcal{A}$.

  By the efficient randomness resampling property on $\Pi$, the sampling randomness $r_{\pi^{*(1)}_\gamma}$ is statistically close to the resampled randomness $r_{e^*_\gamma}$ in $\mathsf{G}_5$ with statistical distance $\epsilon^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda)$. Therefore, $\mathsf{sk}^{(1)}$ (computed using $(r_{\pi^{*(1)}_\gamma})_{\gamma\in[k]}$) is statistically close to $\mathsf{sk}'$ (computed using $(r_{e^*_\gamma})_{\gamma\in[k]}$) in $\mathsf{G}_5$ with statistical distance $k \cdot \epsilon^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda)$.

  Overall, in the case $b = 1$, $\mathcal{B}_4$'s simulation is statistical close to $\mathsf{G}_5$ with statistical distance $k \cdot \epsilon^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda)$.

Consequently, we have

$$\begin{aligned}
&\epsilon^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{B}_4}(\lambda) = |\Pr[b' = b] - \tfrac{1}{2}| \\
&= \tfrac{1}{2} \cdot |\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| \\
&\geq \tfrac{1}{2} \cdot \big(|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_4 = 1]| - k \cdot \epsilon^{\Pi\text{-resmp}}_{\mathsf{HPS}}(\lambda)\big).
\end{aligned}$$

This completes the proof of Lemma 3. □

### C.5 Proof of Lemma 4

**Lemma 4** ($\mathsf{G}_5 \to \mathsf{G}_6$) *There exists an unbounded $\mathcal{B}_5$ against the openable$_k$ property of tag-based $\widetilde{\mathsf{HPS}}$, s.t. $|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq 2 \cdot \epsilon^{\mathrm{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{B}_5}(\lambda)$.*

*Proof.* This proof is similar to the proof of Lemma 3. We construct an unbounded $\mathcal{B}_5$ by invoking $\mathcal{A}$, as shown below.

- In the beginning, $\mathcal{B}_5$ is given $(\widetilde{\mathsf{mpar}}, \widetilde{\mathsf{par}}, \widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, (x^*_\gamma, r_{x^*_\gamma})_{\gamma\in[k]})$, where $\widetilde{\mathsf{mpar}} \leftarrow_{\$} \widetilde{\mathsf{MPar}}(1^\lambda)$, $(\widetilde{\mathsf{par}}, \widetilde{\mathsf{td}}) \leftarrow_{\$} \widetilde{\mathsf{Par}}(\widetilde{\mathsf{mpar}})$, $\widetilde{\mathsf{sk}} \leftarrow_{\$} \widetilde{\mathcal{SK}}$, $\widetilde{\mathsf{pk}} := \widetilde{\alpha}(\widetilde{\mathsf{sk}})$, and $x^*_\gamma \leftarrow_{\$} \mathcal{X}$ with randomness $r_{x^*_\gamma}$.
- $\mathcal{B}_5$ invokes $\mathsf{mpar} \leftarrow_{\$} \mathsf{MPar}(1^\lambda)$, samples $H \leftarrow_{\$} \mathcal{H}$, and sets $\mathsf{pp} := (\mathsf{mpar}, \widetilde{\mathsf{mpar}}, H)$. It then invokes $(\mathsf{par}, \mathsf{td}) \leftarrow_{\$} \mathsf{Par}(\mathsf{mpar})$, samples $\mathsf{sk} \leftarrow_{\$} \mathcal{SK}$, computes $\mathsf{pk} := \alpha(\mathsf{sk})$, and sets $pk := (\mathsf{par}, \widetilde{\mathsf{par}}, \mathsf{pk}, \widetilde{\mathsf{pk}})$. $\mathcal{B}_5$ also computes an $\widetilde{\mathsf{sk}}^* \in \widetilde{\mathcal{SK}}$ such that $\widetilde{\mathsf{pk}} = \widetilde{\alpha}(\widetilde{\mathsf{sk}}^*)$.
- $\mathcal{B}_5$ forwards $(\mathsf{pp}, pk)$ to $\mathcal{A}$, and simulates $\mathcal{O}_{\mathrm{dec}}$ oracle for $\mathcal{A}$ as follows. For $\mathcal{O}_{\mathrm{dec}}(c)$ query with $c = (x, d, \widetilde{\pi})$, $\mathcal{B}_5$ computes $\tau := H(x, d)$, and outputs $\perp$ immediately if $c \in \{c^*_\gamma\}_{\gamma\in[k]}$ or $\tau \in \{\tau^*_\gamma\}_{\gamma\in[k]}$ or $x \in \mathcal{X} \setminus \mathcal{L}$ or $\widetilde{\pi} \neq \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}^*}(x, \tau)$ using $\widetilde{\mathsf{sk}}^*$. Otherwise, it computes $m := d - \Lambda_{\mathsf{sk}}(x) \in \Pi$, and returns $m$ to $\mathcal{A}$.
- $\mathcal{B}_5$ receives $(m^*_\gamma)_{\gamma\in[k]}$ from $\mathcal{A}$, and prepares $(c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma))_{\gamma\in[k]}$ as follows. For each $\gamma \in [k]$, $\mathcal{B}_5$ sets $x^*_\gamma$ as the one in its own challenge, samples $d^*_\gamma \leftarrow_{\$} \Pi$, and computes $\tau^*_\gamma := H(x^*_\gamma, d^*_\gamma)$. Then $\mathcal{B}_5$ returns $(\tau^*_\gamma)_{\gamma\in[k]}$ to its own challenger, and receives $((\widetilde{\pi}^{*(b)}_\gamma)_{\gamma\in[k]}, \widetilde{\mathsf{sk}}^{(b)})$ from its challenger, where $\widetilde{\pi}^{*(0)}_\gamma := \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x^*_\gamma, \tau^*_\gamma)$, $\widetilde{\mathsf{sk}}^{(0)} := \widetilde{\mathsf{sk}}$, $\widetilde{\pi}^{*(1)}_\gamma \leftarrow_{\$} \widetilde{\Pi}$ with randomness $r_{\widetilde{\pi}^{*(1)}_\gamma}$, $\widetilde{\mathsf{sk}}^{(1)} \leftarrow_{\$} \widetilde{\mathsf{HOpen}}_k(\widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, (x^*_\gamma, r_{x^*_\gamma}, \widetilde{\pi}^{*(1)}_\gamma, r_{\widetilde{\pi}^{*(1)}_\gamma}, \tau^*_\gamma)_{\gamma\in[k]})$, and $b \leftarrow_{\$} \{0, 1\}$. $\mathcal{B}_5$ is asked to guess the value of $b$. For each $\gamma \in [k]$, $\mathcal{B}_5$ sets $\widetilde{\pi}^*_\gamma := \widetilde{\pi}^{*(b)}_\gamma$ contained in its own challenge.
- $\mathcal{B}_5$ sends $(c^*_\gamma := (x^*_\gamma, d^*_\gamma, \widetilde{\pi}^*_\gamma))_{\gamma\in[k]}$ to $\mathcal{A}$, and continues the simulation of $\mathcal{O}_{\mathrm{dec}}$ oracle for $\mathcal{A}$ as described above.
- Finally, $\mathcal{B}_5$ computes $e^*_\gamma := d^*_\gamma - m^*_\gamma \in \Pi$ and resamples $r_{e^*_\gamma} \leftarrow_{\$} \mathsf{ReSmp}_\Pi(e^*_\gamma)$ for each $\gamma \in [k]$, invokes $\mathsf{sk}' \leftarrow_{\$} \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x^*_\gamma, r_{x^*_\gamma}, e^*_\gamma, r_{e^*_\gamma})_{\gamma\in[k]})$, sends $(\mathsf{sk}', \widetilde{\mathsf{sk}}^{(b)})$ to $\mathcal{A}$, and receives a bit $b'$ from $\mathcal{A}$. $\mathcal{B}_5$ sends $b'$ to its own challenger.

We analyze the advantage of $\mathcal{B}_5$. Firstly, we note that $\mathcal{B}_5$'s simulation of oracle $\mathcal{O}_{\mathrm{dec}}$ (using $\widetilde{\mathsf{sk}}^*$) is identical to the $\mathcal{O}_{\mathrm{dec}}$ in $\mathsf{G}_5$ and $\mathsf{G}_6$ (using $\widetilde{\mathsf{sk}}$). The reason is, due to the new rejection rule added in $\mathsf{G}_4$, $\mathcal{O}_{\mathrm{dec}}$ outputs $\perp$ immediately if $x \in \mathcal{X} \setminus \mathcal{L}$, and for $x \in \mathcal{L}$, $\widetilde{\Lambda}_{\widetilde{\mathsf{sk}}^*}(x, \tau) = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$ always holds by the perfect correctness of $\widetilde{\mathsf{HPS}}$. Next, we claim that $\mathcal{B}_5$ perfectly simulates $\mathsf{G}_5$ when $b = 0$, and perfectly simulates $\mathsf{G}_6$ when $b = 1$.

- In the case $b = 0$, for each $\gamma \in [k]$, $\widetilde{\pi}^*_\gamma := \widetilde{\pi}^{*(0)}_\gamma = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x^*_\gamma, \tau^*_\gamma)$, and in the last step, $\mathcal{B}_5$ sends $(\mathsf{sk}', \widetilde{\mathsf{sk}}^{0)} = \widetilde{\mathsf{sk}})$ to $\mathcal{A}$. This is the same as $\mathsf{G}_5$.
- In the case $b = 1$, for each $\gamma \in [k]$, $\widetilde{\pi}^*_\gamma := \widetilde{\pi}^{*(1)}_\gamma \leftarrow_{\$} \widetilde{\Pi}$ with randomness $r_{\widetilde{\pi}^{*(1)}_\gamma}$, the same as $\mathsf{G}_6$. Moreover, in the last step, $\mathcal{B}_5$ sends $(\mathsf{sk}', \widetilde{\mathsf{sk}}^{(1)})$, where

$\widetilde{\mathsf{sk}}^{(1)} \leftarrow_{\$} \widetilde{\mathsf{HOpen}}_k(\widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, (x^*_\gamma, r_{x^*_\gamma}, \widetilde{\pi}^{*(1)}_\gamma, r_{\widetilde{\pi}^{*(1)}_\gamma}, \tau^*_\gamma)_{\gamma\in[k]})$, also the same as $\mathsf{G}_6$.

Consequently, we have

$$\begin{aligned}&\boldsymbol{\epsilon}^{\mathrm{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{B}_5}(\lambda) = |\Pr[b'=b] - \tfrac{1}{2}|\\ =\ &\tfrac{1}{2}\cdot|\Pr[b'=1 \mid b=1] - \Pr[b'=1 \mid b=0]|\\ =\ &\tfrac{1}{2}\cdot|\Pr[\mathsf{G}_6=1] - \Pr[\mathsf{G}_5=1]|.\end{aligned}$$

This completes the proof of Lemma 4. □

### C.6 Proof of Theorem 4

**Proof of Theorem 4.** First of all, for any $\mathsf{mpar} = (\mathcal{G}, d, k, l, \mathcal{D}_{d+k,d}, \mathsf{H}_\mathsf{u})$, any $\mathsf{par} = [\mathbf{A}]$, any $\mathsf{sk} = \mathbf{S} \in \mathcal{SK}$ and $\mathsf{pk} = [\mathbf{P}] \in \mathcal{PK}$ where $[\mathbf{P}] = \alpha(\mathsf{sk}) = [\mathbf{A}^\top\mathbf{S}]$, if $x = [\mathbf{x}] = [\mathbf{Aw}] \in \mathcal{L}$ with witness $w = \mathbf{w}$, then

$$\mathsf{Pub}(\mathsf{pk}, x, w) = \mathsf{H}_{\mathsf{u},l}([\mathbf{P}^\top]\mathbf{w}) = \mathsf{H}_{\mathsf{u},l}([\mathbf{S}^\top\mathbf{Aw}]) = \mathsf{H}_{\mathsf{u},l}(\mathbf{S}^\top[\mathbf{x}]) = \mathsf{Priv}(\mathsf{sk}, x).$$

Thus, the instantiation HPS in Section 6.2 is a valid hash proof system with perfect correctness.

Next, we prove the following claim which states that the instantiation HPS has a hard $k$-fold SMP under the $k$-fold $\mathcal{D}_{d+k,d}$-MDDH assumption.

*Claim.* For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{k\text{-msmp}}_{\mathsf{HPS},\mathcal{A}}(\lambda) \le \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{D}_{d+k,d},\mathsf{GGen},\mathcal{B}}(\lambda).$$

*Proof of Claim.* The construction of $\mathcal{B}$ is straightforward. After getting inputs $(\mathcal{G}, [\mathbf{A}], [\mathbf{X}])$ where $\mathcal{G} = (\mathbb{G}, q, P) \leftarrow_{\$} \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow_{\$} \mathcal{D}_{d+k,d}$ and $[\mathbf{X}] \in \mathbb{G}^{(d+k)\times k}$, $\mathcal{B}$ selects $\mathsf{H}_\mathsf{u} \leftarrow_{\$} \mathcal{H}_\mathsf{u}$ and sets $\mathsf{mpar} = (\mathcal{G}, d, k, l, \mathcal{D}_{d+k,d}, \mathsf{H}_\mathsf{u})$, $\mathsf{par} = [\mathbf{A}]$ and returns $\mathcal{A}(\mathsf{mpar}, \mathsf{par}, [\mathbf{X}])$.

If every column of $\mathbf{X}$ is uniformly chosen from $\mathsf{span}(\mathbf{A})$, then $\mathcal{A}$ gets properly distributed public parameters together with $k$ uniform elements from language $\mathcal{L}$. If every column of $\mathbf{X}$ is uniformly chosen from $\mathbb{Z}_q^{d+k}$, then $[\mathbf{X}]$ is uniformly distributed over $\mathcal{X}^k$. Thus, the claim follows. □

Next, we show that if algorithm $\mathsf{HOpen}_k$ does not output $\perp$, then the outputted key $\mathsf{sk}'$ is correct. More precisely, we prove the following claim.

*Claim.* For any $\mathsf{mpar} \leftarrow_{\$} \mathsf{MPar}(1^\lambda)$, any $(\mathsf{par}, \mathsf{td}) \leftarrow_{\$} \mathsf{Par}(\mathsf{mpar})$, any secret key $\mathsf{sk} \in \mathcal{SK}$ with its public key $\mathsf{pk} := \alpha(\mathsf{sk})$, any instances $(x_\gamma)_{\gamma\in\{1,\cdots,k\}} \in \mathcal{X}^k$ with their sampling randomnesses $(r_{x_\gamma})_{\gamma\in\{1,\cdots,k\}} \in (R_\mathcal{X})^k$, any hash values $(\pi_\gamma)_{\gamma\in\{1,\cdots,k\}} \in \Pi^k$ with their sampling randomness $(r_{\pi_\gamma})_{\gamma\in\{1,\cdots,k\}} \in (R_\Pi)^k$ and any $\mathsf{sk}' \leftarrow_{\$} \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x_\gamma, r_{x_\gamma}, \pi_\gamma, r_{\pi_\gamma})_{\gamma\in\{1,\cdots,k\}})$, if $\mathsf{sk}' \neq \perp$, then it holds that

$$\mathsf{pk} = \alpha(\mathsf{sk}') \ \text{ and } \ \pi_\gamma = \Lambda_{\mathsf{sk}'}(x_\gamma) \text{ for all } \gamma \in \{1, \cdots, k\}.$$

*Proof of Claim.* For any $\mathsf{par} = [\mathbf{A}]$ with trapdoor $\mathsf{td} = \mathbf{A}$, any $\mathsf{sk} = \mathbf{S} \in \mathcal{SK}$, $\mathsf{pk} = [\mathbf{P}] = [\mathbf{A}^\top \mathbf{S}]$, any $x_\gamma = [\mathbf{x}_\gamma] \in \mathcal{X}$ with sampling randomness $r_{x_\gamma} = \mathbf{x}_\gamma$, any $\pi_\gamma = \mathsf{H}_{\mathsf{u},l}([\boldsymbol{\pi}_\gamma]) \in \Pi$ with randomness $r_{\pi_\gamma} = \boldsymbol{\pi}_\gamma \in R_\Pi$ for all $\gamma \in \{1, \cdots, k\}$, and any $\mathsf{sk}' \leftarrow_\$ \mathsf{HOpen}_k(\mathsf{td}, \mathsf{pk}, \mathsf{sk}, (x_\gamma, r_{x_\gamma}, \pi_\gamma, r_{\pi_\gamma})_{\gamma \in \{1,\cdots,k\}})$, if $\mathsf{sk}' = \mathbf{S}' \in \mathcal{SK}$ is not $\perp$, $\mathbf{S}'$ is the (only) solution to the linear equation system (3). So it holds that $\mathbf{S}'^\top \mathbf{A} = \mathbf{S}^\top \mathbf{A} = \mathbf{P}^\top \bmod q$ and $\mathbf{S}'^\top \mathbf{x}_\gamma = \boldsymbol{\pi}_\gamma \bmod q$. This implies $\mathsf{pk} = [\mathbf{P}] = [\mathbf{A}^\top]\mathbf{S}' = \alpha(\mathsf{sk}')$ and $\pi_\gamma = \mathsf{H}_{\mathsf{u},l}([\boldsymbol{\pi}_\gamma]) = \mathsf{H}_{\mathsf{u},l}(\mathbf{S}'^\top[\mathbf{x}_\gamma]) = \Lambda_{\mathsf{sk}'}(x_\gamma)$ for all $\gamma \in \{1, \cdots, k\}$. Thus, instantiation $\mathsf{HPS}$ is a valid key-openable hash proof system and the above claim follows. □

Next we prove the following claim which shows that $\mathsf{HPS}$ is openable$_k$.

*Claim.* For any (unbounded) adversary $\mathcal{A}$, $\boldsymbol{\epsilon}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{A}}(\lambda) \leq \dfrac{k}{q}$.

*Proof of Claim.* We prove the claim by showing that $(\mathsf{mpar}, \mathsf{par}, \mathsf{td}, \mathsf{pk}, (x_\gamma, r_{x_\gamma}, \pi^{(0)}_\gamma)_{\gamma\in\{1,\cdots,k\}}, \mathsf{sk}^{(0)})$ is statistically close to $(\mathsf{mpar}, \mathsf{par}, \mathsf{td}, \mathsf{pk}, (x_\gamma, r_{x_\gamma}, \pi^{(1)}_\gamma)_{\gamma\in\{1,\cdots,k\}}, \mathsf{sk}^{(1)})$ in game $\mathbf{Exp}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{A}}(\lambda)$ in Figure 6.

For $\mathsf{mpar} = (\mathcal{G}, d, k, l, \mathcal{D}_{d+k,d}, \mathsf{H}_\mathsf{u})$, $\mathsf{par} = [\mathbf{A}]$, $\mathsf{td} = \mathbf{A}$, $\mathsf{pk} = [\mathbf{P}]$, $x_\gamma = [\mathbf{x}_\gamma]$ and $r_{x_\gamma} = \mathbf{x}_\gamma$ for $\gamma \in \{1, \cdots, k\}$, let $\mathbf{M} = (\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k) \in \mathbb{Z}_q^{(d+k)\times(d+k)}$, then two subcases are possible.

- $\mathbf{M}$ is invertible. In this case, the secret key $\mathsf{sk}^{(0)} = \mathbf{S}$ follows a uniform distribution conditioned on $\mathbf{P} = \mathbf{A}^\top \mathbf{S}$. Furthermore, $\pi^{(0)}_\gamma = \mathsf{H}_{\mathsf{u},l}([\boldsymbol{\pi}_\gamma])$ equals $\mathsf{H}_{\mathsf{u},l}([\mathbf{S}^\top \mathbf{x}_\gamma])$ for all $\gamma \in \{1, \cdots, k\}$. We see that the following equation holds.

  $$\mathbf{S}^\top(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k) = (\mathbf{P}^\top \mid \boldsymbol{\pi}_1 \mid \cdots \mid \boldsymbol{\pi}_k).$$

  Since $\mathbf{M}$ is invertible, $(\boldsymbol{\pi}_1 \mid \cdots \mid \boldsymbol{\pi}_k)$ follows a uniform distribution over $\mathbb{Z}_q^{l\times k}$ due to the conditional uniform distribution of $\mathbf{S}$. Moreover, the conditional distribution of $(\mathbf{S}, \boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_k)$ can be generated as follows. We first select uniform $r_{\pi_\gamma} = \boldsymbol{\pi}_\gamma$ from $\mathbb{Z}_q^l$ for all $\gamma \in \{1, \cdots, k\}$ and calculate

  $$\mathbf{S}^\top = (\mathbf{P}^\top \mid \boldsymbol{\pi}_1 \mid \cdots \mid \boldsymbol{\pi}_k) \cdot \mathbf{M}^{-1}.$$

  And this calculation is exactly what the algorithm $\mathsf{HOpen}_k$ does when $\mathbf{M}$ is invertible. Thus, the conditional distribution of $((\pi^{(0)}_\gamma)_{\gamma\in\{1,\cdots,k\}}, \mathsf{sk}^{(0)})$ is identical to the conditional distribution of $((\pi^{(1)}_\gamma)_{\gamma\in\{1,\cdots,k\}}, \mathsf{sk}^{(1)})$ in this subcase.
- $\mathbf{M}$ is not invertible. We show that this subcase happens with probability at most $\frac{k}{q}$ in game $\mathbf{Exp}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{A}}(\lambda)$.
  Since $\mathbf{A}$ has full column rank, the event that $\mathbf{M}$ is not invertible is equivalent to the event that there exists some $\gamma \in \{1, \cdots, k\}$ such that

  $$\mathbf{x}_\gamma \in \mathsf{span}(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_{\gamma-1}).$$

  In game $\mathbf{Exp}^{\mathrm{open}_k}_{\mathsf{HPS},\mathcal{A}}(\lambda)$, $\mathbf{x}_\gamma$ is uniformly selected from $\mathbb{Z}_q^{d+k}$ for every $\gamma \in \{1, \cdots, k\}$. Thus, for any fixed $\gamma \in \{1, \cdots, k\}$, the probability that $\mathbf{x}_\gamma \in$

$\mathsf{span}(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_{\gamma-1})$ is upper bounded by $\frac{q^{d+\gamma-1}}{q^{d+k}} \leq \frac{1}{q}$. Thus, by union bound, the probability that $\mathbf{M}$ is not invertible is upper bounded by $\frac{k}{q}$.

The above claim follows after combining the two subcases. □

Finally, we prove the following claim showing that $\mathsf{HPS}$ supports efficient randomness resampling on $\Pi$ with algorithm $\mathsf{ReSmp}_{\Pi}$ in Figure 9.

*Claim.* $\boldsymbol{\epsilon}_{\mathsf{HPS}}^{\Pi\text{-resmp}}(\lambda) \leq \frac{l}{2^\lambda} + \frac{l(\lambda+1)}{\sqrt{2q}}$.

*Proof of Claim.* We prove this claim by showing that

$$\Delta_l := \Delta((\mathsf{H}_\mathsf{u}, \pi, r_\pi), (\mathsf{H}_\mathsf{u}, \pi', r'_{\pi'})) \leq \frac{l}{2^\lambda} + \frac{l(\lambda+1)}{\sqrt{2q}}$$

for $\mathsf{H}_\mathsf{u} \leftarrow_\$ \mathcal{H}_\mathsf{u}$, $\pi = \mathsf{H}_{\mathsf{u},l}([\mathbf{s}]) = (\mathsf{H}_\mathsf{u}([s_1]), \cdots, \mathsf{H}_\mathsf{u}([s_l]))$ with $r_\pi = \mathbf{s} = (s_1, \cdots, s_l) \leftarrow_\$ \mathbb{Z}_q^l$, $\pi' = (b_1, \cdots, b_l) \leftarrow_\$ \{0,1\}^l$ and $r'_{\pi'} = (r'_1, \cdots, r'_l) \leftarrow_\$ \mathsf{ReSmp}_{\Pi}(b_1, \cdots, b_l)$ where $r'_i \leftarrow_\$ \mathsf{OnebitReSmp}(\mathsf{H}_\mathsf{u}, b_i)$ for all $i \in \{1, \cdots, l\}$ according to algorithm $\mathsf{ReSmp}_{\Pi}$ in Figure 9. More precisely speaking, we have that

$$\Delta_l = \Delta\left(\left(\mathsf{H}_\mathsf{u}, \begin{pmatrix} \mathsf{H}_\mathsf{u}([s_1]) & s_1 \\ \vdots & \vdots \\ \mathsf{H}_\mathsf{u}([s_l]) & s_l \end{pmatrix}\right), \left(\mathsf{H}_\mathsf{u}, \begin{pmatrix} b_1 & \mathsf{OnebitReSmp}(\mathsf{H}_\mathsf{u}, b_1) \\ \vdots & \vdots \\ b_l & \mathsf{OnebitReSmp}(\mathsf{H}_\mathsf{u}, b_l) \end{pmatrix}\right)\right).$$

By a hybrid argument we get that $\Delta_l \leq l \cdot \Delta_1$ where

$$\Delta_1 := \Delta((\mathsf{H}_\mathsf{u}, \mathsf{H}_\mathsf{u}([s]), s), (\mathsf{H}_\mathsf{u}, b, \mathsf{OnebitReSmp}(\mathsf{H}_\mathsf{u}, b)))$$

where $\mathsf{H}_\mathsf{u} \leftarrow_\$ \mathcal{H}_\mathsf{u}$, $s \leftarrow_\$ \mathbb{Z}_q$ and $b \leftarrow_\$ \{0,1\}$.

To show that $\Delta_1$ is statistically small, we first show that

$$p_\perp := \Pr_{\substack{\mathsf{H}_\mathsf{u} \leftarrow_\$ \mathcal{H}_\mathsf{u} \\ b \leftarrow_\$ \{0,1\}}} [\mathsf{OnebitReSmp}(\mathsf{H}_\mathsf{u}, b) = \perp]$$

is small. According to Figure 9, algorithm $\mathsf{OnebitReSmp}$ tries $\lambda$ times. For the $j$-th attempt, the algorithm randomly selects $r_j \leftarrow_\$ \mathbb{Z}_q$ and tests whether $\mathsf{H}_\mathsf{u}([r_j])$ equals $b$. If all the $\lambda$ attempts fail, algorithm $\mathsf{OnebitReSmp}$ outputs $\perp$. So we have that

$$\begin{aligned} p_\perp = \Pr_{\substack{\mathsf{H}_\mathsf{u} \leftarrow_\$ \mathcal{H}_\mathsf{u} \\ b \leftarrow_\$ \{0,1\} \\ r_j \leftarrow_\$ \mathbb{Z}_q}} \left[\bigwedge_{j=1}^{\lambda} (\mathsf{H}_\mathsf{u}([r_j]) \neq b)\right] &\leq \Pr_{\substack{h_j \leftarrow_\$ \{0,1\} \\ b \leftarrow_\$ \{0,1\}}} \left[\bigwedge_{j=1}^{\lambda} (h_j \neq b)\right] + \Delta_{\mathsf{u},\lambda} \\ &= \frac{1}{2^\lambda} + \Delta_{\mathsf{u},\lambda}, \end{aligned}$$

where $\Delta_{\mathsf{u},\lambda} := \Delta((\mathsf{H}_\mathsf{u}, \mathsf{H}_\mathsf{u}([r_1]), \cdots, \mathsf{H}_\mathsf{u}([r_\lambda])), (\mathsf{H}_\mathsf{u}, h_1, \cdots, h_\lambda))$ for $\mathsf{H}_\mathsf{u} \leftarrow_\$ \mathcal{H}_\mathsf{u}$, $r_j \leftarrow_\$ \mathbb{Z}_q$ and $h_j \leftarrow_\$ \{0,1\}$. By a hybrid argument, we have that $\Delta_{\mathsf{u},\lambda} \leq \lambda \cdot \Delta_{\mathsf{u},1}$

where $\Delta_{\mathsf{u},1} := \Delta((\mathsf{H_u}, \mathsf{H_u}([r])), (\mathsf{H_u}, h))$ for $\mathsf{H_u} \leftarrow_\$ \mathcal{H}_\mathsf{u}, r \leftarrow_\$ \mathbb{Z}_q$ and $h \leftarrow_\$ \{0, 1\}$. We have that $\Delta_{\mathsf{u},1} \leq \frac{1}{\sqrt{2q}}$ due to the leftover hash lemma (c.f., Lemma 6) together with the fact that $\mathcal{H}_\mathsf{u}$ is a family of universal hash function. Thus, we have that

$$p_\perp \leq \frac{1}{2^\lambda} + \frac{\lambda}{\sqrt{2q}}. \tag{14}$$

Next, we define a perfect one-bit resample algorithm $\mathsf{PerfectOnebitReSmp}$ as follows.

$\mathsf{PerfectOnebitReSmp}(\mathsf{H_u}, b \in \{0,1\})$:
Define set $R(\mathsf{H_u}, b) := \{s \in \mathbb{Z}_q \mid \mathsf{H_u}(s) = b\}$
  If $R(\mathsf{H_u}, b) \neq \emptyset$:
    $s' \leftarrow_\$ R(\mathsf{H_u}, b)$
  Else:
    $s' := \perp$
Return $s'$

**Fig. 12** Perfect one-bit randomness resample algorithm $\mathsf{PerfectOnebitReSmp}$.

Note that this algorithm is only conceptual and its running time is not a concern here. Intuitively, if for function $\mathsf{H_u}$ there exists preimage of $b$, algorithm $\mathsf{PerfectOnebitReSmp}$ always outputs a uniform preimage. Otherwise it outputs $\perp$. Thus, for $\mathsf{H_u} \leftarrow_\$ \mathcal{H}_\mathsf{u}$ and $b \leftarrow_\$ \{0,1\}$, we have that

$$\Delta_p := \Delta((\mathsf{H_u}, b, \mathsf{OnebitReSmp}(\mathsf{H_u}, b)), (\mathsf{H_u}, b, \mathsf{PerfectOnebitReSmp}(\mathsf{H_u}, b))) \leq p_\perp.$$

The reason is that when algorithm $\mathsf{PerfectOnebitReSmp}$ does not output $\perp$, algorithm $\mathsf{OnebitReSmp}$ can still output $\perp$ with probability no larger than $p_\perp$. Then

$$\begin{aligned}\Delta_1 &= \Delta((\mathsf{H_u}, \mathsf{H_u}([s]), s), (\mathsf{H_u}, b, \mathsf{OnebitReSmp}(\mathsf{H_u}, b))) \\ &\leq \underbrace{\Delta((\mathsf{H_u}, \mathsf{H_u}([s]), s), (\mathsf{H_u}, b, \mathsf{PerfectOnebitReSmp}(\mathsf{H_u}, b)))}_{\text{Denoted as } \Delta_1'} + \Delta_p.\end{aligned}$$

We will show that $\Delta_1'$ is small. Before that, we first focus on the following statistical distance. Namely, for $\mathsf{H_u} \leftarrow_\$ \mathcal{H}_\mathsf{u}, s \leftarrow_\$ \mathbb{Z}_q$ and $b \leftarrow_\$ \{0,1\}$, consider

$$\begin{aligned}&\Delta((\mathsf{H_u}, \mathsf{H_u}([s])), (\mathsf{H_u}, b)) \\ &= \frac{1}{2} \sum_{\mathsf{h} \in \mathcal{H}_\mathsf{u}} \sum_{\beta \in \{0,1\}} |\Pr[(\mathsf{H_u}, \mathsf{H_u}([s])) = (\mathsf{h}, \beta)] - \Pr[(\mathsf{H_u}, b) = (\mathsf{h}, \beta)]| \\ &= \sum_{\mathsf{h} \in \mathcal{H}_\mathsf{u}} \Pr[\mathsf{H_u} = \mathsf{h}] \cdot \frac{1}{2} \sum_{\beta \in \{0,1\}} |\Pr[\mathsf{h}([s]) = \beta \mid \mathsf{H_u} = \mathsf{h}] - \Pr[b = \beta \mid \mathsf{H_u} = \mathsf{h}]| \\ &= \sum_{\mathsf{h} \in \mathcal{H}_\mathsf{u}} \Pr[\mathsf{H_u} = \mathsf{h}] \cdot \frac{1}{2} \sum_{\beta \in \{0,1\}} \left| \frac{|R(\mathsf{h}, \beta)|}{q} - \frac{1}{2} \right| \end{aligned} \tag{15}$$

where the set $R(\mathsf{h}, \beta)$ is the preimage set defined in Figure 12.

Finally, for $\mathsf{H_u} \leftarrow\!\!\$\ \mathcal{H}_\mathsf{u}, s \leftarrow\!\!\$\ \mathbb{Z}_q$ and $b \leftarrow\!\!\$\ \{0,1\}$, we consider

$$\begin{aligned}
\Delta_1' &= \Delta((\mathsf{H_u}, \mathsf{H_u}([s]), s), (\mathsf{H_u}, b, \mathsf{PerfectOnebitReSmp}(\mathsf{H_u}, b))) \\
&= \frac{1}{2} \sum_{\mathsf{h} \in \mathcal{H}_\mathsf{u}} \sum_{\beta \in \{0,1\}} \sum_{\mu \in \mathbb{Z}_q \cup \{\perp\}} \left| \begin{array}{c} \Pr[(\mathsf{H_u}, \mathsf{H_u}([s]), s) = (\mathsf{h}, \beta, \mu)] \\ - \\ \Pr[(\mathsf{H_u}, b, \mathsf{PerfectOnebitReSmp}(\mathsf{H_u}, b)) = (\mathsf{h}, \beta, \mu)] \end{array} \right| \\
&= \sum_{\mathsf{h} \in \mathcal{H}_\mathsf{u}} \Pr[\mathsf{H_u} = \mathsf{h}] \\
&\quad \cdot \frac{1}{2} \sum_{\beta \in \{0,1\}} \underbrace{\sum_{\mu \in \mathbb{Z}_q \cup \{\perp\}} \left| \begin{array}{c} \Pr[(\mathsf{h}([s]), s) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] \\ - \\ \Pr[(b, \mathsf{PerfectOnebitReSmp}(\mathsf{h}, b)) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] \end{array} \right|}_{\text{Denoted as } \delta(\mathsf{h}, \beta)}
\end{aligned}$$

We claim that for any $\mathsf{h} \in \mathcal{H}_\mathsf{u}$ and any $\beta \in \{0,1\}$,

$$\delta(\mathsf{h}, \beta) = \left| \frac{|R(\mathsf{h}, \beta)|}{q} - \frac{1}{2} \right|.$$

The reason is as follows.

- If $|R(\mathsf{h}, \beta)| = 0$, then $\Pr[(\mathsf{h}([s]), s) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] = 0$ for any $\mu$ because it is not possible that $\beta$ has preimage under function $\mathsf{h}$. Similarly we have that $\Pr[(b, \mathsf{PerfectOnebitReSmp}(\mathsf{h}, b)) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] = 0$ for any $\mu \in \mathbb{Z}_q$ and $\Pr[(b, \mathsf{PerfectOnebitReSmp}(\mathsf{h}, b)) = (\beta, \perp) \mid \mathsf{H_u} = \mathsf{h}] = \frac{1}{2}$ due to the random choice of $b$. Thus,
$$\delta(\mathsf{h}, \beta) = \frac{1}{2} = \left| \frac{|R(\mathsf{h}, \beta)|}{q} - \frac{1}{2} \right|.$$
- If $|R(\mathsf{h}, \beta)| > 0$.
  - If $\mu \in R(\mathsf{h}, \beta)$, then $\Pr[(\mathsf{h}([s]), s) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] = \frac{1}{q}$ due to the random choice of $s$ and $\Pr[(b, \mathsf{PerfectOnebitReSmp}(\mathsf{h}, b)) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] = \frac{1}{2|R(\mathsf{h}, \beta)|}$ due to the random choice of $b$ together with the randomness in algorithm $\mathsf{PerfectOnebitReSmp}$.
  - Otherwise if $\mu \notin R(\mathsf{h}, \beta)$, then $\Pr[(\mathsf{h}([s]), s) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] = 0$ and $\Pr[(b, \mathsf{PerfectOnebitReSmp}(\mathsf{h}, b)) = (\beta, \mu) \mid \mathsf{H_u} = \mathsf{h}] = 0$.

  Then
  $$\delta(\mathsf{h}, \beta) = |R(\mathsf{h}, \beta)| \cdot \left| \frac{1}{q} - \frac{1}{2|R(\mathsf{h}, \beta)|} \right| = \left| \frac{|R(\mathsf{h}, \beta)|}{q} - \frac{1}{2} \right|.$$

  Thus,

$$\begin{aligned}
\Delta_1' &= \sum_{\mathsf{h} \in \mathcal{H}_\mathsf{u}} \Pr[\mathsf{H_u} = \mathsf{h}] \cdot \frac{1}{2} \sum_{\beta \in \{0,1\}} \left| \frac{|R(\mathsf{h}, \beta)|}{q} - \frac{1}{2} \right| \\
&= \Delta((\mathsf{H_u}, \mathsf{H_u}([s])), (\mathsf{H_u}, b)),
\end{aligned}$$

where the last equality holds because of Equation (15). According to the leftover hash lemma (c.f., Lemma 6), we have that $\Delta_1' = \Delta((\mathsf{H}_\mathsf{u}, \mathsf{H}_\mathsf{u}([s])), (\mathsf{H}_\mathsf{u}, b)) \leq \frac{1}{\sqrt{2q}}$.

Putting everything together, we have that

$$\begin{aligned}\Delta_l &\leq l \cdot \Delta_1 \\ &\leq l \cdot (\Delta_1' + \Delta_p) \\ &\leq l \cdot (\Delta_1' + p_\perp) \\ &\leq l \cdot \left(\frac{1}{\sqrt{2q}} + \frac{1}{2^\lambda} + \frac{\lambda}{\sqrt{2q}}\right) = \frac{l}{2^\lambda} + \frac{l(\lambda+1)}{\sqrt{2q}}.\end{aligned}$$

Thus, the above claim follows. □

This completes the proof of Theorem 4. ■

### C.7 Proof of Theorem 5

**Proof of Theorem 5.** First of all, for any $\widetilde{\mathsf{mpar}} := (\mathcal{G}, d, k, \mathcal{D}_{d+k,d})$, any $\widetilde{\mathsf{par}} = [\mathbf{A}]$, any $\widetilde{\mathsf{sk}} = \mathbf{s} \in \widetilde{\mathcal{SK}}$ and any $\widetilde{\mathsf{pk}} = [\mathbf{p}] = \begin{bmatrix}\mathbf{A}^\top & \\ & \mathbf{A}^\top\end{bmatrix}\mathbf{s} \in \widetilde{\mathcal{PK}}$ and any $\tau \in \mathcal{T}$, if $x = [\mathbf{x}] = [\mathbf{A}\mathbf{w}] \in \mathcal{L}$ with witness $w = \mathbf{w}$, then

$$\widetilde{\mathsf{Pub}}(\widetilde{\mathsf{pk}}, x, w, \tau) = [\mathbf{p}^\top]\begin{pmatrix}\mathbf{w} \\ \tau\mathbf{w}\end{pmatrix} = \left[\mathbf{s}^\top\begin{pmatrix}\mathbf{A} & \\ & \mathbf{A}\end{pmatrix}\begin{pmatrix}\mathbf{w} \\ \tau\mathbf{w}\end{pmatrix}\right] = \left[\mathbf{s}^\top\begin{pmatrix}\mathbf{x} \\ \tau\mathbf{x}\end{pmatrix}\right]$$

and it equals to $\widetilde{\mathsf{Priv}}(\widetilde{\mathsf{sk}}, x, \tau)$. Thus, the instantiation $\widetilde{\mathsf{HPS}}$ in Section 6.3 is a valid tag-based hash proof system with perfect correctness.

Next, we show that if algorithm $\widetilde{\mathsf{HOpen}}_k$ does not output $\perp$, then the outputted key $\widetilde{\mathsf{sk}}'$ is correct. More precisely, we prove the following claim.

*Claim.* For any $\widetilde{\mathsf{mpar}} \leftarrow_\$ \widetilde{\mathsf{MPar}}(1^\lambda)$, any $(\widetilde{\mathsf{par}}, \widetilde{\mathsf{td}}) \leftarrow_\$ \widetilde{\mathsf{Par}}(\widetilde{\mathsf{mpar}})$, any secret key $\widetilde{\mathsf{sk}} \in \widetilde{\mathcal{SK}}$ with its public key $\widetilde{\mathsf{pk}} := \widetilde{\alpha}(\widetilde{\mathsf{sk}})$, any instances $(x_\gamma)_{\gamma\in[k]} \in \mathcal{X}^k$ with their sampling randomnesses $(r_{x_\gamma})_{\gamma\in[k]} \in (R_\mathcal{X})^k$, any hash values $(\widetilde{\pi}_\gamma)_{\gamma\in[k]} \in \widetilde{\Pi}^k$ with their sampling randomnesses $(r_{\widetilde{\pi}_\gamma})_{\gamma\in[k]} \in (R_{\widetilde{\Pi}})^k$, any tags $(\tau_\gamma)_{\gamma\in[k]} \in \mathcal{T}^k$ and any opened secret key $\widetilde{\mathsf{sk}}' \leftarrow_\$ \widetilde{\mathsf{HOpen}}_k(\widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, (x_\gamma, r_{x_\gamma}, \widetilde{\pi}_\gamma, r_{\widetilde{\pi}_\gamma}, \tau_\gamma)_{\gamma\in[k]})$, if $\widetilde{\mathsf{sk}}' \neq \perp$, then it holds that

$$\widetilde{\mathsf{pk}} = \widetilde{\alpha}\left(\widetilde{\mathsf{sk}}'\right) \quad \text{and} \quad \widetilde{\pi}_\gamma = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}'}(x_\gamma, \tau_\gamma) \text{ for all } \gamma \in [k].$$

*Proof of Claim.* For any $\widetilde{\mathsf{mpar}} = (\mathcal{G}, d, k, \mathcal{D}_{d+k,d})$, any $\widetilde{\mathsf{par}} = [\mathbf{A}]$, any $\widetilde{\mathsf{td}} = \mathbf{A}$, any $\widetilde{\mathsf{sk}} = \mathbf{s} = \begin{pmatrix}\mathbf{s}_1 \\ \mathbf{s}_2\end{pmatrix}$, any $\widetilde{\mathsf{pk}} = [\mathbf{p}] = \begin{bmatrix}\mathbf{A}^\top & \\ & \mathbf{A}^\top\end{bmatrix}\mathbf{s} = [\mathbf{s}_1^\top\mathbf{A}, \mathbf{s}_2^\top\mathbf{A}]^\top$, any $x_\gamma = [\mathbf{x}_\gamma]$ from $\mathcal{X}$ with sampling randomness $r_{x_\gamma} = \mathbf{x}_\gamma$, any $\widetilde{\pi}_\gamma = [r_{\widetilde{\pi}_\gamma}] \in \widetilde{\Pi}$ with randomness

$r_{\widetilde{\pi}_\gamma} \in R_{\widetilde{\Pi}}$, any tag $\tau_\gamma \in \mathcal{T}$ for $\gamma \in [k]$ and any $\widetilde{\mathsf{sk}}' \leftarrow_{\$} \widetilde{\mathsf{HOpen}}_k(\widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, (x_\gamma, r_{x_\gamma}, \widetilde{\pi}_\gamma, r_{\widetilde{\pi}_\gamma}, \tau_\gamma)_{\gamma\in[k]})$, if $\widetilde{\mathsf{sk}}' \neq \perp$, then $\widetilde{\mathsf{sk}}' = \mathbf{s}'$ is a solution to the linear equation system (4). So it holds that

$$\mathbf{s}'^{\top} \begin{pmatrix} \mathbf{A} & \mathbf{x}_1 & \cdots & \mathbf{x}_k \\ & \mathbf{A} & \tau_1\mathbf{x}_1 & \cdots \tau_k\mathbf{x}_k \end{pmatrix} = (\underbrace{\mathbf{s}_1^{\top}\mathbf{A}, \ \mathbf{s}_2^{\top}\mathbf{A}}_{\mathbf{p}^{\top}}, \ r_{\widetilde{\pi}_1}, \cdots, \ r_{\widetilde{\pi}_k}).$$

We have $\widetilde{\mathsf{pk}} = \widetilde{\alpha}\left(\widetilde{\mathsf{sk}}'\right)$ and $\widetilde{\pi}_\gamma = \widetilde{\Lambda}_{\widetilde{\mathsf{sk}}'}(x_\gamma, \tau_\gamma)$ for all $\gamma \in [k]$.

Thus, the claim follows and the instantiation $\widetilde{\mathsf{HPS}}$ is a valid tag-based key-openable hash proof system. □

Next, we show that $\widetilde{\mathsf{HPS}}$ is universal$_{k+1}$ by proving the following claim.

*Claim.* For any (unbounded) adversary $\mathcal{A}$, $\boldsymbol{\epsilon}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) \leq \dfrac{k+1}{q}$.

*Proof of Claim.* In game $\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda)$, $k$ instances are uniformly selected from $\mathcal{X}$, i.e., $x_\gamma = [\mathbf{x}_\gamma] \leftarrow_{\$} \mathbb{G}^{d+k}$ for $\gamma \in [k]$. We use $\mathsf{E}$ to denote the event that matrix $(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k)$ is invertible. Then, we have that

$$\begin{aligned}
&\Pr[\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) = 1] \\
&= \Pr[\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) = 1 \wedge \mathsf{E}] + \Pr[\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) = 1 \wedge \neg\mathsf{E}] \\
&\leq \Pr[\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) = 1 \mid \mathsf{E}] + \Pr[\neg\mathsf{E}] \\
&\leq \Pr[\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) = 1 \mid \mathsf{E}] + \frac{k}{q} \qquad (16)
\end{aligned}$$

The proof of $\Pr[\neg\mathsf{E}] \leq \frac{k}{q}$ is exactly the same as the proof in Theorem 4.

Next, we will prove that

$$\Pr[\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) = 1 \mid \mathsf{E}] = \frac{1}{q}. \qquad (17)$$

We prove that $\widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau)$ is uniformly distributed over $\widetilde{\Pi}$ from the point view of $\mathcal{A}$. First, $\mathcal{A}$ knows nothing about the uniform secret key $\widetilde{\mathsf{sk}} = \mathbf{s}$ except for the information of the public key $\widetilde{\mathsf{pk}}$ and the $k$ hash values $(\widetilde{\pi}_\gamma)_{\gamma\in[k]}$ where

$$(\widetilde{\mathsf{pk}}, \widetilde{\pi}_1, \cdots, \widetilde{\pi}_k) = \mathbf{s}^{\top} \begin{bmatrix} \mathbf{A} & \mathbf{x}_1 & \cdots & \mathbf{x}_k \\ & \mathbf{A} & \tau_1\mathbf{x}_1 & \cdots \tau_k\mathbf{x}_k \end{bmatrix}.$$

The hash value that adversary $\mathcal{A}$ wants to predict is $\widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x, \tau) = \mathbf{s}^{\top} \begin{bmatrix} \mathbf{x} \\ \tau\mathbf{x} \end{bmatrix}$. So if we can prove that matrix

$$\begin{pmatrix} \mathbf{A} & \mathbf{x}_1 & \cdots & \mathbf{x}_k & \mathbf{x} \\ & \mathbf{A} & \tau_1\mathbf{x}_1 & \cdots & \tau_k\mathbf{x}_k & \tau\mathbf{x} \end{pmatrix} \qquad (18)$$

has full column rank, then the hash value $\widetilde{\Lambda}_{\widetilde{\mathsf{sk}}}(x,\tau)$ is uniformly distributed from the view of $\mathcal{A}$. Suppose that there exists $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{Z}_q^d, s_1, \cdots, s_k, s \in \mathbb{Z}_q$ such that

$$\begin{pmatrix} \mathbf{A} & \mathbf{x}_1 & \cdots & \mathbf{x}_k & \mathbf{x} \\ & \mathbf{A} & \tau_1\mathbf{x}_1 & \cdots & \tau_k\mathbf{x}_k & \tau\mathbf{x} \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ s_1 \\ \vdots \\ s_k \\ s \end{pmatrix} = \mathbf{0} \quad \text{and} \quad \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ s_1 \\ \vdots \\ s_k \\ s \end{pmatrix} \neq \mathbf{0}.$$

Then we have that

$$\mathbf{A}\mathbf{r}_1 + s_1\mathbf{x}_1 + \cdots + s_k\mathbf{x}_k + s\mathbf{x} = \mathbf{0} \tag{19}$$

$$\mathbf{A}\mathbf{r}_2 + s_1\tau_1\mathbf{x}_1 + \cdots + s_k\tau_k\mathbf{x}_k + s\tau\mathbf{x} = \mathbf{0} \tag{20}$$

Combining the above two equations, we get

$$\mathbf{A}(\tau\mathbf{r}_1 - \mathbf{r}_2) + s_1(\tau - \tau_1)\mathbf{x}_1 + \cdots + s_k(\tau - \tau_k)\mathbf{x}_k = \mathbf{0}$$

Under the condition that matrix $(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k)$ is invertible, we know that $s_\gamma(\tau - \tau_\gamma) = 0$ for all $\gamma \in [k]$. Since $\tau \notin \{\tau_\gamma\}_{\gamma\in[k]}$, we must have $s_\gamma = 0$ for all $\gamma \in [k]$. Combining equation (19) and (20), we get that

$$\mathbf{A}\mathbf{r}_1 + s\mathbf{x} = \mathbf{0}$$
$$\mathbf{A}\mathbf{r}_2 + s\tau\mathbf{x} = \mathbf{0}$$

We know $s \neq 0$ otherwise $\mathbf{r}_1 = \mathbf{r}_2 = \mathbf{0}$ (since $\mathbf{A}$ has full column rank) and contradicts the assumption that $(\mathbf{r}_1^\top, \mathbf{r}_2^\top, s_1, \cdots, s_k, s) \neq \mathbf{0}^\top$. Then we have that $\mathbf{x} = s^{-1}\mathbf{A}\mathbf{r}_1 \in \mathsf{span}(\mathbf{A})$ and it contradicts to the event that $x \notin \mathcal{L}$ since $\mathbf{Exp}^{\mathrm{univ}_{k+1}}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) = 1$.

Thus, matrix (18) has full column rank and equation (17) holds.

The claim follows combining inequality (16) and equation (17). □

Next, we prove the following claim which shows that $\widetilde{\mathsf{HPS}}$ is openable$_k$.

*Claim.* For any (unbounded) adversary $\mathcal{A}$, $\boldsymbol{\epsilon}^{\mathrm{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda) \leq \dfrac{k}{q}$.

*Proof of Claim.* We prove the claim by showing that $(\widetilde{\mathsf{mpar}}, \widetilde{\mathsf{par}}, \widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, (x_\gamma, r_{x_\gamma}, \widetilde{\pi}^{(0)}_\gamma)_{\gamma\in[k]}, \widetilde{\mathsf{sk}}^{(0)})$ distributes identically to $(\widetilde{\mathsf{mpar}}, \widetilde{\mathsf{par}}, \widetilde{\mathsf{td}}, \widetilde{\mathsf{pk}}, (x_\gamma, r_{x_\gamma}, \widetilde{\pi}^{(1)}_\gamma)_{\gamma\in[k]}, \widetilde{\mathsf{sk}}^{(1)})$ in $\mathbf{Exp}^{\mathrm{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda)$.

For $\widetilde{\mathsf{mpar}} = (\mathcal{G}, d, k, \mathcal{D}_{d+k,d}), \widetilde{\mathsf{par}} = [\mathbf{A}]$, $\widetilde{\mathsf{td}} = \mathbf{A}$, $\widetilde{\mathsf{pk}} = [\mathbf{p}]$, $x_\gamma = [\mathbf{x}_\gamma]$ and $r_{x_\gamma} = \mathbf{x}_\gamma$ for $\gamma \in [k]$, two subcases are possible

– Matrix $(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k)$ is invertible. In this case, the secret key $\widetilde{\mathsf{sk}}^{(0)} = \mathbf{s}$ follows a uniform distribution conditioned on $\mathbf{p} = \begin{pmatrix} \mathbf{A}^\top & \\ & \mathbf{A}^\top \end{pmatrix} \mathbf{s}$. Furthermore, $\widetilde{\pi}_\gamma^{(0)} = [r_{\widetilde{\pi}_\gamma}]$ equals the hash value $\left[\mathbf{s}^\top \begin{pmatrix} \mathbf{x}_\gamma \\ \tau_\gamma \mathbf{x}_\gamma \end{pmatrix}\right]$ for all $\gamma \in [k]$. We see that the following equation holds.

$$\mathbf{s}^\top \begin{pmatrix} \mathbf{A} & \mathbf{x}_1 & \cdots & \mathbf{x}_k \\ \mathbf{A} & \tau_1 \mathbf{x}_1 & \cdots & \tau_k \mathbf{x}_k \end{pmatrix} = (\mathbf{p}^\top, r_{\widetilde{\pi}_1}, \cdots, r_{\widetilde{\pi}_k}). \tag{21}$$

Since we have matrix $(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k)$ is invertible, then matrix

$$\begin{pmatrix} \mathbf{A} & \mathbf{x}_1 & \cdots & \mathbf{x}_k \\ \mathbf{A} & \tau_1 \mathbf{x}_1 & \cdots & \tau_k \mathbf{x}_k \end{pmatrix}$$

has full column rank. Thus, $(r_{\widetilde{\pi}_1}, \cdots, r_{\widetilde{\pi}_k})$ follows a uniform distribution over $\mathbb{Z}_q^k$ due to the conditional uniform distribution of $\mathbf{s}$.
Moreover, the condition distribution of $(\mathbf{s}, [r_{\widetilde{\pi}_1}], \cdots, [r_{\widetilde{\pi}_k}])$ can be generated by selecting uniform $r_{\widetilde{\pi}_\gamma} \leftarrow_\$ \mathbb{Z}_q$ for all $\gamma \in [k]$, treating $\mathbf{s}$ as a unknown variable and randomly select a solution to the linear system in Equation (21). And this calculation is exactly what the algorithm $\widetilde{\mathsf{HOpen}}_k$ does when matrix $(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k)$ is invertible. Thus, the conditional distribution of $((\widetilde{\pi}_\gamma^{(0)})_{\gamma \in [k]}, \widetilde{\mathsf{sk}}^{(0)})$ is identical to the conditional distribution of $((\widetilde{\pi}_\gamma^{(1)})_{\gamma \in [k]}, \widetilde{\mathsf{sk}}^{(1)})$ in this subcase.
– Matrix $(\mathbf{A} \mid \mathbf{x}_1 \mid \cdots \mid \mathbf{x}_k)$ is not invertible. This subcase happens with probability at most $\frac{k}{q}$ in game $\mathbf{Exp}^{\mathrm{open}_k}_{\widetilde{\mathsf{HPS}},\mathcal{A}}(\lambda)$ and the proof is exactly the same with the one in Appendix C.6.

The above claim follows after combining the two subcases. □

This completes the proof of Theorem 5. ■

### C.8 Proof of Lemma 5

**Proof of Lemma 5.** For any PPT adversary $\mathcal{A}$, we first construct a PPT $k$-fold $\mathcal{U}_{d+k,d}$-MDDH adversary $\mathcal{B}'$ and prove that

$$\mathbf{Adv}^{(n,k)\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{k\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}'}(\lambda) + \frac{k}{q}. \tag{22}$$

On getting a $k$-fold $\mathcal{U}_{d+k,d}$-MDDH challeng $(\mathcal{G}, [\mathbf{A}], [\mathbf{X}])$ where $\mathcal{G} \leftarrow_\$ \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow_\$ \mathcal{U}_{d+k,d}$ and $\mathbf{X} \in \mathbb{Z}_q^{(d+k)\times k}$, $\mathcal{B}'$ selects uniform $\mathbf{R}_i \leftarrow_\$ \mathbb{Z}_q^{(d+k)\times(d+k)}$ and uniform $\mathbf{W}_i \leftarrow_\$ \mathbb{Z}_q^{d\times k}$ for all $i \in \{1, \cdots, n\}$. Then $\mathcal{B}'$ sets $[\mathbf{A}_i] := \mathbf{R}_i[\mathbf{A}]$ and $[\mathbf{X}_i] := \mathbf{R}_i[\mathbf{X}] + [\mathbf{A}_i]\mathbf{W}_i$. Finally, $\mathcal{B}'$ invokes $\mathcal{A}(\mathcal{G}, ([\mathbf{A}_i], [\mathbf{X}_i])_{i=1}^n)$ and outputs whatever $\mathcal{A}$ outputs.

We analyse the input distribution of $\mathcal{A}$ as follows. First, if we denote $\mathbf{R}_i^{(L)} \in \mathbb{Z}_q^{(d+k)\times d}$ as the left $d$ columns of matrix $\mathbf{R}_i$ and denote $\mathbf{R}_i^{(R)} \in \mathbb{Z}_q^{(d+k)\times k}$ as the right $k$ columns of matrix $\mathbf{R}_i$, then $\mathbf{R}_i = (\mathbf{R}_i^{(L)} \mid \mathbf{R}_i^{(R)})$ and $\mathbf{A}_i = \mathbf{R}_i\mathbf{A} = \mathbf{R}_i^{(L)}\overline{\mathbf{A}} + \mathbf{R}_i^{(R)}\underline{\mathbf{A}}$ where $\overline{\mathbf{A}}$ is the upper $d$ rows of $\mathbf{A}$ and $\underline{\mathbf{A}}$ is the lower $k$ rows of $\mathbf{A}$. Since $\mathbf{A}$ is chosen from a uniform matrix distribution, we know that $\overline{\mathbf{A}}$ is a full rank square matrix. Due to the random choice of $\mathbf{R}_i^{(L)}$, we know that $\mathbf{A}_i$ distributes uniformly over $\mathbb{Z}_q^{(d+k)\times d}$. Next, we analyse the distribution of $\mathbf{X}_i$.

- If $\mathbf{X}$ are $k$ random vectors in the column span of $\mathbf{A}$, then there exists $\mathbf{W} \in \mathbb{Z}_q^{d\times k}$ such that $\mathbf{X} = \mathbf{A}\mathbf{W}$. Then

$$\mathbf{X}_i = \mathbf{R}_i\mathbf{X} + \mathbf{A}_i\mathbf{W}_i = \mathbf{R}_i\mathbf{A}\mathbf{W} + \mathbf{A}_i\mathbf{W}_i = \mathbf{A}_i(\mathbf{W} + \mathbf{W}_i).$$

  That means $\mathbf{X}_i$ are $k$ independent and uniform vectors in the span of $\mathbf{A}_i$ due to the random choice of $\mathbf{W}_i$.
- If $\mathbf{X}$ are $k$ random vectors from $\mathbb{Z}_q^{d+k}$, then we have that matrix $(\mathbf{A} \mid \mathbf{X})$ is an invertible square matrix except with probability at most $\frac{k}{q}$. In this way, $\mathbf{R}_i(\mathbf{A} \mid \mathbf{X}) = (\mathbf{R}_i\mathbf{A} \mid \mathbf{R}_i\mathbf{X})$ is uniformly distributed due to the random choice of $\mathbf{R}_i$. Thus, $\mathbf{R}_i\mathbf{X}$ distributes uniformly over $\mathbb{Z}_q^{(d+k)\times k}$ and is independent of $\mathbf{A}_i = \mathbf{R}_i\mathbf{A}$. This implies that $\mathbf{X}_i = \mathbf{R}_i\mathbf{X} + \mathbf{A}_i\mathbf{W}_i$ is uniformly distributed and independent of $\mathbf{A}_i$.

In this way, $\mathcal{B}'$ perfectly simulates the input of $\mathcal{A}$ except with probability at most $\frac{k}{q}$. So we have that $\mathbf{Adv}^{k\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}'}(\lambda) \geq \mathbf{Adv}^{(n,k)\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{A}}(\lambda) - \frac{k}{q}$ and equation (22) holds.

Finally, for uniform matrix distribution, the $k$-fold $\mathcal{U}_{d+k,d}$-MDDH assumption can be tightly reduced to the $\mathcal{U}_{d+k,d}$-MDDH assumption due to the random self-reducibility of $\mathcal{U}_{d+k,d}$-MDDH [GHKW16, Lemma 3]. Thus, we can build a PPT adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{k\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}'}(\lambda) \leq \mathbf{Adv}^{\text{mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}}(\lambda) + \frac{1}{q-1}. \tag{23}$$

Combining equation (22) and (23), we have

$$\mathbf{Adv}^{(n,k)\text{-mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{mddh}}_{\mathcal{U}_{d+k,d},\mathsf{GGen},\mathcal{B}}(\lambda) + \frac{k+1}{q-1}.$$

Thus, Lemma 5 follows. ■