Possibility and Impossibility Results for Receiver Selective Opening Secure PKE in the Multi-Challenge Setting

Rupeng Yang[1]⋆, Junzuo Lai[2]⋆, Zhengan Huang[3]⋆, Man Ho Au[1], Qiuliang Xu[4], and Willy Susilo[5]

[1] Department of Computer Science, The University of Hong Kong, Hong Kong, China
orbbyrp@gmail.com, allenau@cs.hku.hk
[2] College of Information Science and Technology, Jinan University, Guangzhou, China
laijunzuo@gmail.com
[3] Peng Cheng Laboratory, Shenzhen, China
zhahuang.sjtu@gmail.com
[4] School of Software, Shandong University, Jinan, China
xql@sdu.edu.cn
[5] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong NSW, Australia
wsusilo@uow.edu.au

Abstract. Public key encryption (PKE) schemes are usually deployed in an open system with numerous users. In practice, it is common that some users are corrupted. A PKE scheme is said to be receiver selective opening (RSO) secure if it can still protect messages transmitted to uncorrupted receivers after the adversary corrupts some receivers and learns their secret keys. This is usually defined by requiring the existence of a simulator that can simulate the view of the adversary given only the opened messages. Existing works construct RSO secure PKE schemes in a single-challenge setting, where the adversary can only obtain one challenge ciphertext for each public key. However, in practice, it is preferable to have a PKE scheme with RSO security in the multi-challenge setting, where public keys can be used to encrypt multiple messages.

In this work, we explore the possibility of achieving PKE schemes with receiver selective opening security in the multi-challenge setting. Our contributions are threefold. First, we demonstrate that PKE schemes with RSO security in the single-challenge setting are not necessarily RSO secure in the multi-challenge setting. Then, we show that it is impossible to achieve RSO security for PKE schemes if the number of challenge ciphertexts under each public key is a priori unbounded. In particular, we prove that no PKE scheme can be RSO secure in the k-challenge setting (i.e., the adversary can obtain k challenge ciphertexts for each public key) if its secret key contains less than k bits. On the positive side, we give a concrete construction of PKE scheme with RSO security in the k-challenge setting, where the ratio of the secret key length to k approaches the lower bound 1.

⋆ Corresponding author.

1 Introduction

The standard notion of security for public key encryption (PKE) schemes is indistinguishability of 1-ciphertext (denoted as IND-CPA security). That is to say, given one challenge ciphertext to an adversary, which encrypts a message from a set of two messages chosen by the adversary, it could not distinguish which message is encrypted. Such a simple security notion in fact implies semantic security with multiple challenge ciphertexts, which prevents the adversary from learning any information about the encrypted messages after viewing a priori unbounded number of ciphertexts.

In many real world scenarios, the adversary may have the capability to learn internal states of partial users via corrupting their devices. Such attacks are called selective opening attacks [DNRS99]. A PKE scheme is said to be secure against selective opening attacks if it can still protect messages transmitted between uncorrupted users. Surprisingly, standard security does not imply security against selective opening attacks immediately [BDWY12, HR14, HRW16].

The formal study of selective opening secure PKE was initialized by Bellare et al. in [BHY09]. They consider two types of selective opening attacks, namely, sender selective opening (SSO) attacks, where the attacker corrupts senders and obtains the randomness used for encrypting messages, and receiver selective opening (RSO) attacks, where the attacker corrupts receivers and obtains their secret keys. Also, for each attack, security can be defined by either an indistinguishability-based definition, which extends the standard IND-CPA security to the selective opening setting, or a simulation-based definition, which defines semantic security against selective opening attackers. In all definitions, the adversary first gets some challenge ciphertexts, then it "opens" some of them via corrupting the related users. An indistinguishability-based definition ensures that the adversary is not able to distinguish encrypted messages in unopened ciphertexts, while in a simulation-based definition, there should exist a simulator that can simulate the view of the adversary given only the opened messages.

Since selective opening security can be defined in different manners, it is important to clarify relations between different definitions. As shown in [HPW15], indistinguishability-based selective opening security is not sufficient to imply simulation-based selective opening security in both the SSO setting and the RSO setting. Thus, for selective opening security, it is desirable to consider simulation-based definitions.[1]

It is also interesting to explore whether selective opening security in the single-challenge setting, i.e., each public key is only used once to produce a single challenge ciphertext, is enough for achieving selective opening security in the multi-challenge setting, where each public key can be reused to encrypt multiple

[1] In addition, we prefer simulation-based definitions because indistinguishability-based selective opening security are usually defined for *efficiently conditionally re-samplable* message distributions [BHY09] only. A definition without such restriction (called full IND-SO security [BHK12, ORV14]) needs an inefficient security experiment and seems not achievable.

challenge messages. This question is particularly important for the RSO setting, because all previous works in this area only consider how to construct encryption schemes secure in the single-challenge setting and it is unknown whether they are still secure in the more realistic multi-challenge setting.

1.1 Our Results

In this work, we initiate the study of RSO security in the multi-challenge setting. In particular, we consider an adversary that can obtain k challenge ciphertexts for each public key, and denote security in this setting as RSO_k security. [2] We focus on simulation-based definitions and define security against both the chosen-plaintext adversary (SIM-RSO_k-CPA) and the chosen-ciphertext adversary (SIM-RSO_k-CCA). In summary, our contributions are as follows:

- We show that RSO security in the single-challenge setting is not enough to guarantee RSO security in the multi-challenge setting. We demonstrate this by providing a PKE scheme that is SIM-RSO_k-CCA secure, but is not SIM-RSO_{k+1}-CPA secure for any polynomial k (recall that RSO security in the single-challenge setting can be denoted as RSO_1 security). The PKE schemes build on an IND-CPA secure PKE scheme and a simulation-sound non-interactive zero-knowledge (NIZK) proof, thus, this also provides the first positive result for achieving RSO security in the multi-challenge setting.
- We prove that it is impossible to achieve SIM-RSO security in the multi-challenge setting if we do not bound the number of challenge ciphertexts for each public key. In particular, we provide a lower bound on the secret key length for any PKE scheme with RSO_k security in the non-programmable random oracle model, which indicates that the size of the secret key must be as large as the total number of message bits ever encrypted. For example, for any PKE with RSO_k security, assuming its message space is $\{0,1\}^m$ and the secret key space is $\{0,1\}^l$, then we have $l \geq mk$.
- We construct a concrete SIM-RSO_k-CPA secure PKE scheme from the DDH assumption, where the message space is $\{0,1\}$, the public key is a group element and the secret key only contains a number in $\mathbb{Z}_q$ and k bits.[3] This is nearly optimal in an asymptotic sense as the ratio of secret key length to k is $1+\frac{\log q}{k}$, which approaches the lower bound 1 as the messages number k increases.
- We prove that the well-known Naor-Yung paradigm [NY90, Sah99] still works for SIM-RSO security and give a generic construction of SIM-RSO_k-CCA secure PKE scheme from a SIM-RSO_k-CPA secure PKE scheme, an IND-CPA secure PKE scheme, and a simulation-sound NIZK proof. The construction preserves the key length of the underlying SIM-RSO_k-CPA secure scheme. Thus, combining our (nearly) optimal SIM-RSO_k-CPA secure scheme with the generic construction, we obtain a (nearly) optimal SIM-RSO_k-CCA secure PKE scheme.

[2] Previous definitions in the single-challenge setting are specific cases of this new definition and can be denoted as RSO_1 security.

[3] Here, q is the group order and is fixed by the security parameter.

1.2 Technical Overview

In this section, we give a brief overview of how to achieve our negative and positive results. In a high-level, we first observe that a large enough secret key space (conditioned on some public information) is needed to achieve RSO_k security, and employ this observation to lower bound the secret key length for any RSO_k secure PKE scheme. Then we apply the observation to some concrete constructions and provide counterexamples separating RSO_k security and RSO_{k+1} security. Finally, we construct (nearly) optimal RSO_k secure PKE scheme, whose secret key length approaches the above lower bound in an asymptotic sense.

Next, we describe the ideas in more detail.

On Lower Bounding Key Length of $\mathbf{RSO}_k$ Secure PKE scheme. We start by showing that a RSO_k secure PKE scheme must have a long enough secret key. For simplicity of discussion, here we assume that the message space of the scheme is $\{0,1\}$ and explain why it cannot be RSO_k secure if its secret key length contains at most $k-1$ bits.

Intuitively, this is because the number of possible secret keys are not enough to explain k messages. In more detail, to simulate an adversary's output, a RSO_k simulator[4] should generate challenge ciphertexts and send them to the adversary first. Then in the opening phase, on input the opened messages, the simulator needs to generate secret keys that can map each ciphertext to corresponding message. Remember that it needs to map k fixed ciphertexts to a vector of k 1-bit messages using each secret key. Thus, the number of candidate secret keys should be at least 2^k to guarantee that the simulator is able to choose the correct secret key for every possible messages vector. However, if the secret key length of the scheme does not exceed $k-1$, then the number of possible secret keys will not exceed 2^{k-1}. That is to say, for at least half of the possible messages vectors, the simulator is not able to create a correct secret key to explain them. So, with probability $1/2$ (assuming messages are sampled uniformly), the simulation will fail.

To formalize this intuition, we use ideas in previous works [Nie02, BSW11, BDWY12, BO13] that argue impossibility to achieve simulation-based security against a key-revealing attacker.[5] In a nutshell, given a hash function, which is modeled as a non-programmable random oracle, we define a RSO_k adversary as follows. In the first phase, on receiving a set of n public keys $\boldsymbol{PK} = (pk_i)_{i\in[n]}$, it returns a uniform distribution; then in the second phase, on receiving a set of challenge ciphertexts $\boldsymbol{CT}$, it returns a set of indices $\mathcal{I} \subseteq [n]$, which is the hash of $(\boldsymbol{PK}, \boldsymbol{CT})$; finally, on receiving the opened secret keys $\boldsymbol{SK}_{\mathcal{I}}$ and messages $\boldsymbol{M}_{\mathcal{I}}$[6], it outputs $(\boldsymbol{PK}, \boldsymbol{CT}, \boldsymbol{SK}_{\mathcal{I}})$. Note that a simulator who would like to

[4] We refer the readers to Sec. 2.2 for the formal definition of a RSO_k simulator in either the CPA setting and the CCA setting.

[5] We remark that similar lower bounds on key length are achieved in these works, but these results do not imply lower bound for SIM-RSO secure PKE scheme directly.

[6] We use $\boldsymbol{SK}_{\mathcal{I}}$ and $\boldsymbol{M}_{\mathcal{I}}$ to denote the set of secret keys for $(pk_i)_{i\in\mathcal{I}}$ and messages encrypted under $(pk_i)_{i\in\mathcal{I}}$ respectively

simulate the adversary's view should generate $\boldsymbol{PK}$ and $\boldsymbol{CT}$ before viewing the opened messages, since otherwise, it has to invert the random oracle, which is infeasible. Thus, if we feed the simulator with different messages, it should create secret keys conditioned on fixed $\boldsymbol{PK}$ and $\boldsymbol{CT}$. As the number of possible messages is much larger than the number of possible secret keys[7], such simulator does not exist.

On Separating $\mathbf{RSO}_{k+1}$ Security and $\mathbf{RSO}_k$ Security. Next, we explain how to construct a scheme that is SIM-RSO$_k$-CCA secure, but is not even SIM-RSO$_{k+1}$-CPA secure. Our starting point is an encryption scheme Π_1 from the well-known Naor-Yung paradigm [NY90, Sah99], which is proved to be SIM-RSO$_1$-CCA secure for 1-bit message in [HKM$^+$18]. We first recall the scheme briefly and show that it is not SIM-RSO$_2$-CPA secure. Then we explain how to upgrade it to a scheme that is SIM-RSO$_k$-CCA secure, but is not SIM-RSO$_{k+1}$-CPA secure.

<u>A brief review of Π_1.</u> The scheme Π_1 relies on a normal PKE scheme E and a simulation-sound NIZK proof system. Its public key $PK = (pk_0, pk_1)$ is a pair of public keys of E and its secret key is $SK = (s, sk_s)$, where s is a bit and sk_s is the secret key corresponding to pk_s. The encryption of a bit m includes an encryption of m under pk_0, an encryption of m under pk_1 and a proof indicating that the two ciphertexts encrypt the same message. To decrypt a ciphertext, the decryption algorithm first checks the validity of the proof attached and decrypts the ciphertext under pk_s using sk_s.

The SIM-RSO$_1$-CCA security of Π_1 comes from the fact that given a malformed ciphertext, which encrypts a random bit b under pk_0 and encrypts $1-b$ under pk_1, one can open it to any message $m \in \{0,1\}$. In particular, if $m = b$, then the returned secret key is $(0, sk_0)$ and otherwise, the returned secret key is $(1, sk_1)$. In this way, to simulate the view of a SIM-RSO$_1$-CCA adversary the simulator can generate such malformed ciphertext in the beginning and answer the opening query according to the opened messages. Indistinguishability between malformed ciphertexts and well-formed ciphertexts comes from security of E and zero-knowledge property of the underlying NIZK. Also, determining the secret keys until the opening stage will not affect answers to decryption oracle queries since the adversary is only allowed to submit a well-formed ciphertext, which are identically decrypted under sk_0 and sk_1.

<u>Π_1 is not SIM-RSO$_2$-CPA secure.</u> Next, we show that if for each public key of E, there exists at most one valid secret key for it and it is easy to check if a public key/secret key pair is valid [8], Π_1 will not be SIM-RSO$_2$-CPA secure.

Our key observation is that in this case, while the number of possible secret keys is very large, the number of possible secret keys for a fixed public key is not enough to explain 2 messages. Recall that to prove SIM-RSO$_2$-CPA security of Π_1, we need a simulator that is forced to produce challenge ciphertexts before

[7] If $|\mathcal{I}| = 1$, then the number of possible messages is 2^k while the number of possible secret keys is no more than 2^{k-1}.

[8] Concretely, we may view E as ElGamal encryption scheme.

seeing the opened messages and is required to create the correct secret keys that maps the challenge ciphertexts to the opened messages. For a public key $PK = (pk_0, pk_1)$, the best possible strategy for the simulator to generate the challenge ciphertext seems to set the first ciphertext $CT_1 = (\mathsf{E}.\mathtt{Enc}(pk_0, b_1), \mathsf{E}.\mathtt{Enc}(pk_1, 1-b_1))$ and set the second ciphertext $CT_2 = (\mathsf{E}.\mathtt{Enc}(pk_0, b_2), \mathsf{E}.\mathtt{Enc}(pk_1, 1-b_2))$, where b_1 and b_2 are random bits. Then, in the opening phase, the simulator can return a secret key, which is either $(0, sk_0)$ or $(1, sk_1)$, to the adversary, where sk_0, sk_1 are the unique valid secret keys for pk_0 and pk_1 respectively. The secret key $(0, sk_0)$ can decrypt the challenger ciphertexts to (b_1, b_2) and the secret key $(1, sk_1)$ can decrypt the challenger ciphertexts to $(1-b_1, 1-b_2)$. But if the opened messages are $(b_1, 1-b_2)$ or $(1-b_1, b_2)$, no secret key can map challenge ciphertexts to them. So, with probability $1/2$ (assuming messages are sampled uniformly), the simulation will fail. Therefore, we can exploit the techniques for lower bounding secret key length of RSO_k secure PKE schemes to compromise the RSO_2 security of Π_1.

Upgrading Π_1. Next, we explain how to upgrade Π_1 to a RSO_k-secure but RSO_{k+1}-insecure scheme. Our main idea is to use k pairs of public keys of E to encrypt messages. More precisely, to encrypt a bit m under a public key $PK = (pk_{1,0}, pk_{1,1}, \ldots, pk_{k,0}, pk_{k,1})$, the encryption algorithm first samples a k-bit string $(p_1, \ldots, p_k)$ that $p_1 \oplus \ldots \oplus p_k = m$, and then encrypts p_i with $(pk_{i,0}, pk_{i,1})$. Then it generates a NIZK proof proving the correctness of all k pairs of ciphertexts. The final ciphertext includes all $2k$ ciphertexts of E and the proof.

Now, to simulate the view of an adversary in a SIM-RSO_k experiment, or alternatively, to generate k ciphertexts and open them to any k-bit string, the simulator generates the ciphertexts as follows:

$$\begin{array}{c|cccc}
(pk_{1,0}, pk_{1,1}) & \boxed{(p_{1,1}, 1-p_{1,1})} & (p_{2,1}, p_{2,1}) & \cdots & (p_{k,1}, p_{k,1}) \\
(pk_{2,0}, pk_{2,1}) & (p_{1,2}, p_{1,2}) & \boxed{(p_{2,2}, 1-p_{2,2})} & \cdots & (p_{k,2}, p_{k,2}) \\
\vdots & \vdots & \vdots & \boxed{\ddots} & \vdots \\
(pk_{k,0}, pk_{k,1}) & (p_{1,k}, p_{1,k}) & (p_{2,k}, p_{2,k}) & \cdots & \boxed{(p_{k,k}, 1-p_{k,k})} \\
\hline
 & CT_1 & CT_2 & \cdots & CT_k
\end{array}$$

where each $p_{i,j} \xleftarrow{\$} \{0,1\}$, and CT_i consists of encryption of $(p_{i,j}, p_{i,j})$ (or $(p_{i,i}, 1-p_{i,i})$) under public key $(pk_{j,0}, pk_{j,1})$ and a fake proof generated by the NIZK simulator.

Note that, for each public key pair $(pk_{i,0}, pk_{i,1})$, the simulator is only required to cheat on one ciphertext (the ones in a dashed box), thus it can succeed in finding the correct secret key.

The reason that the new scheme is not SIM-RSO_{k+1}-CPA secure is the same as that why Π_1 is not SIM-RSO_2-CPA secure. Note that in the new scheme, the number of valid secret keys for a public key $PK = (pk_{1,0}, pk_{1,1}, \ldots, pk_{k,0}, pk_{k,1})$ is 2^k, which is much less than the number of possible opening messages (2^{k+1}).

Thus, we can use a similar strategy to show that no simulator is able to simulate the adversary's view in a SIM-RSO$_{k+1}$-CPA experiment.

On Constructing RSO$_k$ Secure PKE Scheme with (Nearly) Optimal Secret Key Length. Now, we demonstrate how to achieve SIM-RSO$_k$-CCA secure PKE scheme with (nearly) optimal secret key length. Note that standard techniques for shortening secret keys of PKE schemes (e.g., deriving secret keys from a shorter seed via a pseudorandom generator) do not work here since in the receiver selective opening setting, the simulator needs to generate secret keys satisfying some conditions and using these techniques may lead to an inefficient simulator (e.g., the simulator may have to invert a pseudorandom generator).

Our starting point is the celebrated Cramer-Shoup encryption scheme [CS98], which was shown to be SIM-RSO$_1$-CCA secure in [HKM$^+$18, HLC$^+$19]. Here, we will use its variant with CPA security ($\Pi_{\text{CS-CPA}}$). We first reduce the key length of the scheme. Then, we upgrade it to be SIM-RSO$_k$-CPA secure via merely adding $k-1$ bits to the secret key. Finally, we transform the scheme into a SIM-RSO$_k$-CCA secure one by employing the well-known Naor-Yung double encryption paradigm [NY90, Sah99], where a normal IND-CPA secure PKE scheme and a simulation-sound NIZK proof is additionally used. In our construction, we fix the secret key of the new scheme to be the secret key of the underlying SIM-RSO$_k$-CPA secure scheme. Also, we need to tweak the security proof to fit the definition of SIM-RSO-CPA/CCA security.

Next, we first recall $\Pi_{\text{CS-CPA}}$ and explain why it is SIM-RSO$_1$-CPA secure. Then we provide a more detailed description on how to reduce its key length and how to upgrade the scheme to achieve SIM-RSO$_k$-CPA security.

A brief review of $\Pi_{\text{CS-CPA}}$. The scheme $\Pi_{\text{CS-CPA}}$ works in a cyclic group $\mathbb{G}$ of prime order q with generator g. Let $g_0 = g^{a_0}, g_1 = g^{a_1}, h = g^b$, then the secret key of the scheme is $(s_0, s_1) \in \mathbb{Z}_q^2$ and the public key is $pk = g_0^{s_0} g_1^{s_1}$. To encrypt a bit $m \in \{0,1\}$, the encryption algorithm samples $w \xleftarrow{\$} \mathbb{Z}_q$, and computes the ciphertext $CT = (x_0, x_1, C) = (g_0^w, g_1^w, pk^w \cdot h^m)$. The decryption algorithm tests if $x_0^{s_0} x_1^{s_1} = C$ and outputs 0 if this is the case.

To simulate the view of a SIM-RSO$_1$-CPA adversary, the simulator can first sample $(s_0', s_1') \xleftarrow{\$} \mathbb{Z}_q^2$, compute $pk = g_0^{s_0'} g_1^{s_1'}$ and generate a malformed ciphertext $CT = (x_0, x_1, C) = (g_0^{w_0}, g_1^{w_1}, x_0^{s_0'} x_1^{s_1'})$ for each receiver. Here, w_0, w_1 are distinct random elements in $\mathbb{Z}_q$ and the malformed ciphertext is indistinguishable from an honestly generated one due to the DDH assumption. Then, for each corrupted receiver, assuming the opened message is m, the simulator creates the secret key (s_0, s_1) compatible with the current view by solving the following equations:

$$\begin{cases} g_0^{s_0} g_1^{s_1} = g_0^{s_0'} g_1^{s_1'} \\ x_0^{s_0} x_1^{s_1} \cdot h^m = x_0^{s_0'} x_1^{s_1'} \end{cases} \tag{1}$$

which can be transformed into

$$\begin{cases} a_0 s_0 + a_1 s_1 = a_0 s_0' + a_1 s_1' \\ a_0 w_0 s_0 + a_1 w_1 s_1 + bm = a_0 w_0 s_0' + a_1 w_1 s_1' \end{cases}$$

The equation has a solution since $w_0 \neq w_1$. Thus, the simulator can succeed in simulating the view of a SIM-RSO$_1$-CPA adversary.

Reducing the Key Length. It is worth noting that in the scheme $\Pi_{\text{CS-CPA}}$, some bits of the secret key are wasted. In particular, the simulator is able to simulate the view of the adversary if Equation (1) has solutions in both the case $m = 0$ and that $m = 1$. Thus, it is appealing to see if the equations still always have solutions in some smaller solution space.

We observe that, if we change the strategy of the simulator, then it is possible to reduce the secret key space to $\mathbb{Z}_q \times \{0,1\}$. In more detail, for each receiver, the simulator samples $(s_0', s_1') \xleftarrow{\$} \mathbb{Z}_q \times \{0,1\}$, computes $pk = g_0^{s_0'} g_1^{s_1'}$ and changes the format of malformed ciphertext into $CT = (x_0, x_1, C) = (g_0^w, g_1^w \cdot h^\alpha, x_0^{s_0'} x_1^{s_1'})$. Here $\alpha = 1$ if $s_1' = 1$ and $\alpha = -1$ if $s_1' = 0$, and the malformed ciphertext is still indistinguishable from an honestly generated one due to the DDH assumption. Now, the secret key (s_0, s_1) needs to satisfy the following equation:

$$\begin{cases} a_0 s_0 + a_1 s_1 = a_0 s_0' + a_1 s_1' \\ a_0 w s_0 + a_1 w s_1 + b\alpha s_1 + bm = a_0 w s_0' + a_1 w s_1' + b\alpha s_1' \end{cases}$$

It is easy to see that if $m = 0$, then $s_1 = s_1'$ and thus $s_1 \in \{0,1\}$; if $m = 1$, then $1 = \alpha \cdot (s_1' - s_1)$, which implies that 1) if $s_1' = 1$, then $s_1 = 0$ and 2) if $s_1' = 0$, then $s_1 = 1$. Therefore, the scheme is still secure if we reduce the secret key length to $\lceil \log q \rceil + 1$.

Achieving RSO$_k$ Security. Next, we show how to upgrade the revised scheme to achieving SIM-RSO$_k$-CPA security. Our first attempt is to use the idea in upgrading the counterexample Π_1, i.e., secret sharing the message into k bits and using k independent instances of the scheme to encrypt each bit. However, this will lead to a scheme with key length $k \cdot (\lceil \log q \rceil + 1)$, which is far from optimal.

To solve this problem, our key observation is that, when generating the k public key/secret key pairs, s_0 and the public key can be reused. More precisely, let $g_0 = g^{a_0}, g_1 = g^{a_1}, \ldots, g_k = g^{a_k}, h = g^b$, then we set the secret key to be $(s_0, s_1, \ldots, s_k) \xleftarrow{\$} \mathbb{Z}_q \times \{0,1\}^k$ and set the public key to be $pk = g_0^{s_0} g_1^{s_1} \ldots g_k^{s_k}$. Note that the secret key only contains $\lceil \log q \rceil + k$ bits. Then, to encrypt a bit $m \in \{0,1\}$, the encryption algorithm samples $w \xleftarrow{\$} \mathbb{Z}_q$, and computes the ciphertext $CT = (x_0, x_1, \ldots, x_k, C) = (g_0^w, g_1^w, \ldots, g_k^w, pk^w \cdot h^m)$. The decryption algorithm tests if $x_0^{s_0} x_1^{s_1} \ldots x_k^{s_k} = C$ and outputs 0 if this is the case.

Next, we illustrate why the above scheme is SIM-RSO$_k$-CPA secure. For each receiver, the simulator samples $(s_0', s_1', \ldots, s_k') \xleftarrow{\$} \mathbb{Z}_q \times \{0,1\}^k$ and computes $pk = g_0^{s_0'} g_1^{s_1'} \ldots g_k^{s_k'}$. Also, it generates k malformed ciphertexts, where for the i-th ciphertext, x_i is dishonestly created. That is, $CT_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,i}, \ldots x_{i,k}, C) = (g_0^{w_i}, g_1^{w_i}, \ldots, g_i^{w_i} \cdot h^{\alpha_i}, \ldots, g_k^{w_i}, x_{i,0}^{s_0'} x_{i,1}^{s_1'} \ldots x_{i,k}^{s_k'})$. Here $\alpha_i = 1$ if $s_i' = 1$ and $\alpha_i = -1$ if $s_i' = 0$. Then, for each corrupted receiver, assuming the

k opened messages are $m_1, \ldots, m_k$, the simulator creates the secret key $(s_0, s_1, \ldots, s_k)$ compatible with the current view by solving the following equations:

$$\begin{cases} \prod_{j=0}^{k} g_j^{s_j} = \prod_{j=0}^{k} g_j^{s'_j} \\ (\prod_{j=0}^{k} x_{1,j}^{s_j}) \cdot h^{m_1} = \prod_{j=0}^{k} x_{1,j}^{s'_j} \\ \quad\vdots \\ (\prod_{j=0}^{k} x_{k,j}^{s_j}) \cdot h^{m_k} = \prod_{j=0}^{k} x_{k,j}^{s'_j} \end{cases}$$

This is equivalent to the following equation:

$$\begin{cases} \sum_{j=0}^{k} a_j s_j = \sum_{j=0}^{k} a_j s'_j \\ (\sum_{j=0}^{k} a_j w_1 s_j) + b\alpha_1 s_1 + b m_1 = (\sum_{j=0}^{k} a_j w_1 s'_j) + b\alpha_1 s'_1 \\ \quad\vdots \\ (\sum_{j=0}^{k} a_j w_k s_j) + b\alpha_k s_k + b m_k = (\sum_{j=0}^{k} a_j w_k s'_j) + b\alpha_k s'_k \end{cases}$$

which can be transformed into

$$\begin{cases} \sum_{j=0}^{k} a_j s_j = \sum_{j=0}^{k} a_j s'_j \\ m_1 = \alpha_1 (s'_1 - s_1) \\ \quad\vdots \\ m_k = \alpha_k (s'_k - s_k) \end{cases}$$

Note that, for $i \in [1, k]$, we can set $s_i = s'_i$ if $m_i = 0$ and set $s_i = 1 - s'_i$ if $m_i = 1$. Therefore, the simulator is able to produce a simulated secret key $(s_0, s_1, \ldots, s_k) \in \mathbb{Z}_q \times \{0,1\}^k$ and thus can simulate the view of the SIM-RSO$_k$-CPA adversary.

1.3 Related Works

Since first proposed in [BHY09], PKE with selective opening security has been extensively studied. Numerous constructions of SSO secure PKE have been proposed based on various assumptions in previous works (see [FHKW10, HLOV11, Hof12, HLQ13, LP15, HJKS15, HP16, LSSS17, BL17, LLHG18] and references therein for more details).

In contrast, the setting of RSO security is less studied. It is folklore that (receiver) non-committing encryption schemes [CFGN96, Nie02, DN00, CHK05, CDSMW09] imply RSO secure PKE schemes. Then, in [HPW15], Hazay et al. show that RSO security is achievable from a variety of well-established cryptographic primitives and construct RSO secure PKE schemes from various assumptions. In subsequent works [JLL16, JLL17, HKM$^+$18, HLC$^+$19], chosen-ciphertext attacks (CCA) are also considered in the RSO setting and PKE schemes with RSO-CCA security are provided. Moreover, in [KT18], RSO-secure identity-based encryption scheme is constructed. However, in all these works, the proposed encryption schemes are only proved to have RSO security in the single-challenge setting.

1.4 Roadmap

We recall some preliminaries and define RSO_k security in Sec. 2. Then in Sec. 3, we provide the lower bound for RSO_k secure PKE scheme. Next, we show our counterexamples separating RSO_k security and RSO_{k+1} security in Sec. 4. Then, we construct (nearly) optimal PKE schemes with SIM-RSO_k-CPA security and SIM-RSO_k-CCA security in Sec. 5. Finally, in Sec. 6, we conclude our work with a few possible future works.

2 Preliminaries

Notations. For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For positive integers n_1, n_2 s.t. $n_1 < n_2$, we use $[n_1, n_2]$ to denote the set $\{n_1, n_1 + 1, \cdots, n_2 - 1, n_2\}$. We use boldface to denote vectors, e.g., $\boldsymbol{x}$. We use $\boldsymbol{x}[i]$ to denote the i-th component of $\boldsymbol{x}$. Also, for a string $s \in \{0,1\}^*$, we use $s[i]$ to denote the i-th bit of s.

For a finite set $\mathcal{S}$, we use $|\mathcal{S}|$ to denote the size of $\mathcal{S}$ and use $s \xleftarrow{\$} \mathcal{S}$ to denote the process of sampling s uniformly from $\mathcal{S}$. For a distribution $\mathcal{D}$, we use $x \leftarrow \mathcal{D}$ to denote the process of sampling x from $\mathcal{D}$. For a positive integer n, we use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$.

For a probabilistic algorithm $\mathtt{A}$, we use $\mathtt{A}(x; r)$ to denote the process of running $\mathtt{A}$ on input x and inner randomness r. We write PPT for probabilistic polynomial-time. We use $\mathtt{negl}(\lambda)$ to denote a negligible function.

2.1 Assumptions and Cryptographic Primitives

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

Unbounded Simulation-Sound NIZK Proofs. The notion of NIZK proof was proposed by Blum et al. in [BFM88]. As shown in [Sah99], an unbounded simulation-sound NIZK proof for every language in NP exists assuming the existence of (doubly-enhanced) trapdoor permutations.

Let $\mathtt{R}$ be an efficiently computable binary relation. A NIZK proof for a language $\mathcal{L} = \{x : \exists w, (x, w) \in \mathtt{R}\}$ consists of three PPT algorithms:

- $\mathtt{Gen}$. On input the security parameter λ, the common reference string generation algorithm outputs a common reference string $\mathtt{crs}$.
- $\mathtt{Prove}$. On input a common reference string $\mathtt{crs}$, a statement $x \in \mathcal{L}$ and a witness w for x, the proving algorithm outputs a proof π.
- $\mathtt{Verify}$. On input a common reference string $\mathtt{crs}$, a statement x and a proof π, the verification algorithm outputs a bit indicating whether the proof is valid.

Also, it satisfies the following conditions:

- **Completeness.** For any $(x, w) \in \mathtt{R}$, let $\mathsf{crs} \leftarrow \mathtt{Gen}(1^\lambda)$ and $\pi \leftarrow \mathtt{Prove}(\mathsf{crs}, x, w)$, then we have $\mathtt{Verify}(\mathsf{crs}, x, \pi) = 1$.
- **Unbounded Zero-Knowledge.** There exists a PPT simulator $(\mathtt{S}_1, \mathtt{S}_2)$ that for any PPT adversary $\mathcal{A}$, we have

  $$\left| \Pr\begin{bmatrix} \mathsf{crs} \leftarrow \mathtt{Gen}(1^\lambda); \\ \mathcal{A}^{\mathtt{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 0 \end{bmatrix} - \Pr\begin{bmatrix} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathtt{S}_1(1^\lambda); \\ \mathcal{A}^{\mathtt{S}(\mathsf{crs},\mathsf{td},\cdot,\cdot)}(\mathsf{crs}) = 0 \end{bmatrix} \right| \le \mathsf{negl}(\lambda)$$

  where $\mathtt{P}(\mathsf{crs}, x, w)$ outputs $\mathtt{Prove}(\mathsf{crs}, x, w)$ if $(x, w) \in \mathtt{R}$ and outputs $\perp$ otherwise; $\mathtt{S}(\mathsf{crs}, \mathsf{td}, x, w)$ outputs $\mathtt{S}_2(\mathsf{crs}, \mathsf{td}, x)$ if $(x, w) \in \mathtt{R}$ and outputs $\perp$ otherwise.
- **Unbounded Simulation-Soundness.** Let $(\mathtt{S}_1, \mathtt{S}_2)$ be a PPT simulator for the zero-knowledge property of the NIZK proof. For any unbounded adversary $\mathcal{A}$, we have

  $$\Pr\begin{bmatrix} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathtt{S}_1(1^\lambda); & & \\ (x, \pi) \leftarrow \mathcal{A}^{\mathtt{S}(\mathsf{crs},\mathsf{td},\cdot)}(\mathsf{crs}); & : & (x, \pi) \notin Q \wedge x \notin \mathcal{L} \\ \text{Let } Q \text{ be list of input/output} & & \wedge\, \mathtt{Verify}(\mathsf{crs}, x, \pi) = 1 \\ \text{pairs for the oracle } \mathtt{S} & & \end{bmatrix} \le \mathsf{negl}(\lambda)$$

where $\mathtt{S}(\mathsf{crs}, \mathsf{td}, x)$ outputs $\mathtt{S}_2(\mathsf{crs}, \mathsf{td}, x)$.

2.2 PKE with $\mathbf{RSO}_k$ Security

A public key encryption scheme $\mathsf{PKE} = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ consists of four PPT algorithms:

- $\mathtt{Setup}$. On input the security parameter 1^λ, the setup algorithm outputs the public parameter pp.
- $\mathtt{Gen}$. On input the public parameter pp, the key generation algorithm outputs a public key pk and a secret key sk.
- $\mathtt{Enc}$. On input the public parameter pp, the public key pk and a message m, the encryption algorithm outputs a ciphertext ct.
- $\mathtt{Dec}$. On input the public parameter pp, the public key pk, the secret key sk and a ciphertext ct, the decryption algorithm outputs a message m.

Correctness of PKE requires that $\Pr[\mathtt{Dec}(\mathsf{pp}, pk, sk, ct) \neq m] \le \mathtt{negl}(\lambda)$ for any message m, where $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda), (pk, sk) \leftarrow \mathtt{Gen}(\mathsf{pp}), ct \leftarrow \mathtt{Enc}(\mathsf{pp}, pk, m)$.

The basic security requirement of PKE schemes is IND-CPA security:

Definition 2.1 (IND-CPA Security). *We say that a PKE scheme* $\mathsf{PKE} = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ *is IND-CPA secure if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\begin{aligned} &\Pr[\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda), (pk, sk) \leftarrow \mathtt{Gen}(\mathsf{pp}), (state, m_0^*, m_1^*) \leftarrow \mathcal{A}_1(pp, pk), \\ &\quad b \xleftarrow{\$} \{0, 1\}, ct^* \leftarrow \mathtt{Enc}(\mathsf{pp}, pk, m_b^*) : \quad \mathcal{A}_2(state, ct^*) = b] \le 1/2 + \mathtt{negl}(\lambda) \end{aligned}$$

In this work, we also consider the stronger receiver selective opening security for PKE schemes. Next, we provide definitions of RSO_k security, which are adapted from previous works [HPW15, HKM$^+$18, HLC$^+$19]. Our definitions consider chosen-plaintext attackers and chosen-ciphertext attackers respectively and in both cases, we will define security in a simulation-based sense.

Definition 2.2 (SIM-RSO$_k$-CPA Security). *We say that a PKE scheme* $\mathsf{PKE} = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ *is* SIM-RSO$_k$-CPA *secure, if for any polynomially bounded function* $n > 0$*, any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$*, such that for any PPT distinguisher* $\mathcal{D}$*,*

$$|\Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CPA-real}}_{\mathsf{PKE},\mathcal{A},n}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CPA-ideal}}_{\mathsf{PKE},\mathcal{S},n}(\lambda)) = 1]| \leq \mathtt{negl}(\lambda)$$

where $\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CPA-real}}_{\mathsf{PKE},\mathcal{A},n}$ *and* $\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CPA-ideal}}_{\mathsf{PKE},\mathcal{S},n}$ *are defined in Figure 1.*

Definition 2.3 (SIM-RSO$_k$-CCA Security). *We say that a PKE scheme* $\mathsf{PKE} = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ *is* SIM-RSO$_k$-CCA *secure, if for any polynomially bounded function* $n > 0$*, any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$*, such that for any PPT distinguisher* $\mathcal{D}$*,*

$$|\Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CCA-real}}_{\mathsf{PKE},\mathcal{A},n}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CCA-ideal}}_{\mathsf{PKE},\mathcal{S},n}(\lambda)) = 1]| \leq \mathtt{negl}(\lambda)$$

where $\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CCA-real}}_{\mathsf{PKE},\mathcal{A},n}$ *and* $\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CCA-ideal}}_{\mathsf{PKE},\mathcal{S},n}$ *are defined in Figure 1.*

3 Lower Bound for PKE with RSO$_k$ Security

In this section, we establish a lower bound on the secret key size of a PKE scheme with RSO_k security. Roughly, we show that a PKE scheme cannot be SIM-RSO$_k$-CPA secure (this also implies that it is not SIM-RSO$_k$-CCA secure) if the length of its secret key is not k times larger than the length of message. Formally, we have:

Theorem 3.1. *Let* $\Pi = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ *be a PKE scheme with secret key space* $\mathcal{SK}$ *and message space* $\mathcal{M}$ *(w.l.o.g, we assume* $\mathcal{SK} = \{0,1\}^l$ *and* $\mathcal{M} = \{0,1\}^m$*). If* $l \leq mk - 1$*, then* Π *is not* SIM-RSO$_k$-CPA *secure in the non-programmable random oracle model.*

Proof. Let $H : \{0,1\}^* \to \{0,1\}^h$ be a hash function, which is modeled as a non-programmable random oracle. Let $\mathcal{PP}$, $\mathcal{PK}$ and $\mathcal{C}$ be the public parameters set, the public key space and the ciphertext space of Π respectively. Also, let $a = \lceil \log |\mathcal{PP}| \rceil$, $b = \lceil \log |\mathcal{PK}| \rceil$, $c = \lceil \log |\mathcal{C}| \rceil$ and let $\kappa = a + b + ck + 2$. Let $n = h + 1$, $\epsilon = 1/(4\kappa)$.

Consider the concrete adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ and distinguisher $\mathcal{D}$ defined in Figure 2. Next, we show that for any PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$:

$$|\Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CPA-real}}_{\Pi,\mathcal{A},n}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k-}\mathtt{CPA-ideal}}_{\Pi,\mathcal{S},n}(\lambda)) = 1]| > \epsilon$$

```
$\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{real}}_{\mathsf{PKE},\mathcal{A},n}$:
  $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
  $(\boldsymbol{pk}, \boldsymbol{sk}) := (pk_i, sk_i)_{i\in[n]} \leftarrow (\mathtt{Gen}(\mathsf{pp}))^n$
  $(\mathcal{M}, s_1) \leftarrow \mathcal{A}_1(\mathsf{pp}, \boldsymbol{pk})$
  $\boldsymbol{M} := (m_{i,j})_{i\in[n],j\in[k]} \leftarrow \mathcal{M}$
  $(c_{i,j} \leftarrow \mathtt{Enc}(\mathsf{pp}, pk_i, m_{i,j}))_{i\in[n],j\in[k]}$
  $(\mathcal{I}, s_2) \leftarrow \mathcal{A}_2((c_{i,j})_{i\in[n],j\in[k]}, s_1)$
  $out \leftarrow \mathcal{A}_3((sk_i, m_{i,j})_{i\in\mathcal{I},j\in[k]}, s_2)$
  Return $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$
```

```
$\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{ideal}}_{\mathsf{PKE},\mathcal{S},n}$:
  $(\mathcal{M}, s_1) \leftarrow \mathcal{S}_1(1^\lambda)$
  $\boldsymbol{M} := (m_{i,j})_{i\in[n],j\in[k]} \leftarrow \mathcal{M}$
  $(\mathcal{I}, s_2) \leftarrow \mathcal{S}_2(s_1)$
  $out \leftarrow \mathcal{S}_3((m_{i,j})_{i\in\mathcal{I},j\in[k]}, s_2)$
  Return $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$
```

```
$\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CCA}-\mathtt{real}}_{\mathsf{PKE},\mathcal{A},n}$:
  $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$
  $(\boldsymbol{pk}, \boldsymbol{sk}) := (pk_i, sk_i)_{i\in[n]} \leftarrow (\mathtt{Gen}(\mathsf{pp}))^n$
  $\mathcal{C} = \emptyset$
  $(\mathcal{M}, s_1) \leftarrow \mathcal{A}_1^{\mathtt{Dec}}(\mathsf{pp}, \boldsymbol{pk})$
  $\boldsymbol{M} := (m_{i,j})_{i\in[n],j\in[k]} \leftarrow \mathcal{M}$
  $(c_{i,j} \leftarrow \mathtt{Enc}(\mathsf{pp}, pk_i, m_{i,j}))_{i\in[n],j\in[k]}$
  $\mathcal{C} := \{(i, c_{i,j}) \mid i \in [n], j \in [k]\}$
  $(\mathcal{I}, s_2) \leftarrow \mathcal{A}_2^{\mathtt{Dec}}((c_{i,j})_{i\in[n],j\in[k]}, s_1)$
  $out \leftarrow \mathcal{A}_3^{\mathtt{Dec}}((sk_i, m_{i,j})_{i\in\mathcal{I},j\in[k]}, s_2)$
  Return $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$
```

```
$\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CCA}-\mathtt{ideal}}_{\mathsf{PKE},\mathcal{S},n}$:
  $(\mathcal{M}, s_1) \leftarrow \mathcal{S}_1(1^\lambda)$
  $\boldsymbol{M} := (m_{i,j})_{i\in[n],j\in[k]} \leftarrow \mathcal{M}$
  $(\mathcal{I}, s_2) \leftarrow \mathcal{S}_2(s_1)$
  $out \leftarrow \mathcal{S}_3((m_{i,j})_{i\in\mathcal{I},j\in[k]}, s_2)$
  Return $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$
```

```
$\mathtt{Dec}(i, c)$:
  If $(i, c) \in \mathcal{C}$:
    Return $\perp$
  Return $\mathtt{Dec}(\mathsf{pp}, pk_i, sk_i, c)$
```

Fig. 1 Experiments for defining SIM-RSO$_k$-CPA security and SIM-RSO$_k$-CCA security. Let $\mathscr{M}$ be the message space of PKE, then in all experiments, $\mathcal{M}$ is a distribution over $\mathscr{M}^{n\times k}$ and $\mathcal{I} \subseteq [n]$.

First, by the correctness of Π, we have

$$\Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{real}}_{\Pi,\mathcal{A},n}(\lambda)) = 1] \leq \mathtt{negl}(\lambda)$$

Next, fixing any PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ [9], let

$$\delta = \Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{ideal}}_{\Pi,\mathcal{S},n}(\lambda)) = 1]$$

Then, it is sufficient to show that δ is notably larger than ϵ. Concretely, we will argue that $\delta \geq 1/(2\kappa)$ in the remaining part of the proof.

To lower bound δ, we consider an auxiliary experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$ defined in Figure 3 and analyze the distribution of its output. Here, we use $\mathcal{R}_D$ to denote the distribution of the randomness for the distinguisher $\mathcal{D}$ (the randomness is used in the decryption algorithm of Π) and use $\mathcal{D}(\cdot, \cdot, \cdot, \cdot; R)$ to denote running the distinguisher $\mathcal{D}$ with randomness R.

[9] Here, w.l.o.g., we assume that $\mathcal{S}_2$ and $\mathcal{S}_3$ are deterministic. This will not restrict the power of $\mathcal{S}$ since we can feed coins for $\mathcal{S}_2$ and $\mathcal{S}_3$ to $\mathcal{S}_1$ and require $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) to put coins for $\mathcal{S}_2$ and $\mathcal{S}_3$ (resp. $\mathcal{S}_3$) in its outputted state s_1 (resp. s_2).

```
$\mathcal{A}_1(\mathsf{PP}, (PK_i)_{i\in[n]})$:
    $\mathcal{M} = \mathcal{U}_{mnk}$
    $s_1 = (\mathsf{PP}, (PK_i)_{i\in[n]})$
    Output $(\mathcal{M}, s_1)$
$\mathcal{A}_2((C_{i,j})_{i\in[n],j\in[k]}, s_1)$:
    $t = H(\mathsf{PP}, (PK_i, C_{i,j})_{i\in[n],j\in[k]})$
    $\mathcal{I} = \{i \mid i \in [h] \wedge t[i] = 1\} \cup \{n\}$
    $s_2 = (s_1, (C_{i,j})_{i\in[n],j\in[k]})$
    Output $(\mathcal{I}, s_2)$
$\mathcal{A}_3((SK_i, m_{i,j})_{i\in\mathcal{I},j\in[k]}, s_2)$:
    $out = (s_2, (SK_i)_{i\in\mathcal{I}})$
    Output $out$

$\mathcal{D}((m_{i,j})_{i\in[n],j\in[k]}, \mathcal{M}, \mathcal{I}, out)$:
    If $\mathcal{M} \neq \mathcal{U}_{mnk}$ : Output 1
    $(\mathsf{PP}, (PK_i, C_{i,j})_{i\in[n],j\in[k]}, (SK_i)_{i\in\mathcal{I}}) = out$
    $t = H(\mathsf{PP}, (PK_i, C_{i,j})_{i\in[n],j\in[k]})$
    $\mathcal{I}' = \{i \mid i \in [h] \wedge t[i] = 1\} \cup \{n\}$
    If $\mathcal{I} \neq \mathcal{I}'$ : Output 1
    $(m'_{i,j} \leftarrow \mathtt{Dec}(\mathsf{PP}, PK_i, SK_i, C_{i,j}))_{i\in\mathcal{I},j\in[k]}$
    For $i \in \mathcal{I}, j \in [k]$ :
        If $m_{i,j} \neq m'_{i,j}$ : Output 1
    Output 0
```

Fig. 2 The adversary $\mathcal{A}$ and $\mathcal{D}$ in attacking SIM-RSO$_k$-CPA security of Π. Here, we abuse the notation of $\mathcal{U}_{mnk}$ to denote the description of an algorithm that outputs uniform mnk-bit string and assume that this description is hardwired in $\mathcal{A}$ and $\mathcal{D}$.

```
$\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$:
    $(\mathcal{M}, s_1) \leftarrow \mathcal{S}_1(1^\lambda)$; $(\mathcal{I}, s_2) = \mathcal{S}_2(s_1)$; $R \leftarrow \mathcal{R}_D$;
    For $\iota \in [1, \kappa]$:
        $(m^\iota_{i,j})_{i\in[n],j\in[k]} \leftarrow \mathcal{M}$; $out^\iota = \mathcal{S}_3((m^\iota_{i,j})_{i\in\mathcal{I},j\in[k]}, s_2)$
        If $\mathcal{D}((m^\iota_{i,j})_{i\in[n],j\in[k]}, \mathcal{M}, \mathcal{I}, out^\iota; R) = 1$ : Output 1
    For $\iota \in [2, \kappa]$:
        Parse $out^\iota = (\mathsf{PP}^\iota, (PK^\iota_i, C^\iota_{i,j})_{i\in[n],j\in[k]}, (SK^\iota_i)_{i\in\mathcal{I}})$
        If $(\mathsf{PP}^{\iota-1}, (PK^{\iota-1}_i, C^{\iota-1}_{i,j})_{i\in[n],j\in[k]}) \neq (\mathsf{PP}^\iota, (PK^\iota_i, C^\iota_{i,j})_{i\in[n],j\in[k]})$ :
            Output 2
    Output 0
```

Fig. 3 The auxiliary experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$.

Lemma 3.1. $\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 0] \leq 1/4$.

Proof. Assume the experiment outputs 0. First, we have $\mathcal{M} = \mathcal{U}_{mnk}$, thus, for each $\iota \in [\kappa], i \in [n], j \in [k]$, $m^\iota_{i,j}$ is sampled uniformly at random from $\{0,1\}^m$. Also, we know that $n \in \mathcal{I}$ and for $\iota \in [\kappa]$ and $j \in [k]$, we set $PK^\iota = PK^\iota_n$, $C^\iota_j = C^\iota_{n,j}$, $SK^\iota = SK^\iota_n$, and $m^\iota_j = m^\iota_{n,j}$. Moreover, we have $(\mathsf{PP}^{\iota-1}, (PK^{\iota-1}, C^{\iota-1}_j)_{j\in[k]}) = (\mathsf{PP}^\iota, (PK^\iota, C^\iota_j)_{j\in[k]})$ for all $\iota \in [n]$ and thus we can write PP^ι as PP, PK^ι as PK and C^ι_j as C_j. Finally, for all $\iota \in [\kappa]$ and $j \in [k]$, we have $m^\iota_j = \mathtt{Dec}(\mathsf{PP}, PK, SK^\iota, C_j; r^\iota_j)$, where r^ι_j is the randomness for $\mathtt{Dec}$ derived deterministically from R.

Next, for any randomness R (which determines $(r^\iota_j)_{\iota\in[\kappa],j\in[k]}$), we analyze the probability that all above requirements are satisfied.

First, fix any tuple $(\mathsf{PP}, PK, \boldsymbol{C} = (C_1, \dots, C_k), \boldsymbol{SK} = (SK^1, \dots, SK^\kappa))$ in $\{0,1\}^{a+b+ck+l\kappa}$, which is *not* necessary the output of the simulator, then we have

$$\Pr[\forall \iota \in [\kappa], j \in [k], m_j^\iota = \mathtt{Dec}(\mathsf{PP}, PK, SK^\iota, C_j; r_j^\iota)] = \frac{1}{2^{mk\kappa}}$$

where the probability is taken over the random choice of each m_j^ι.

As the total possible ways to choose PP, PK, $\boldsymbol{C} = (C_1, \dots, C_k)$, and $\boldsymbol{SK} = (SK^1, \dots, SK^\kappa)$ does not exceed $2^{a+b+ck+l\kappa} = 2^{(l+1)\kappa-2}$, we have

$$\Pr[\exists \mathsf{PP}, PK, \boldsymbol{C}, \boldsymbol{SK} : \forall \iota \in [\kappa], j \in [k], \\ m_j^\iota = \mathtt{Dec}(\mathsf{PP}, PK, SK^\iota, C_j; r_j^\iota)] \le \frac{2^{(l+1)\kappa-2}}{2^{mk\kappa}} \le \frac{2^{mk\kappa-2}}{2^{mk\kappa}} = \frac{1}{4}$$

Therefore, the probability that the auxiliary experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$ outputs 0 does not exceed $1/4$. □

Lemma 3.2. $\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 1] \le \kappa \cdot \delta$.

Proof. First, note that randomness of the experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$ comes from three parts, namely, R, randomness of the simulator $\mathcal{S}$ (denoted as ρ here) and randomness used in sampling $m_{i,j}^\iota$. Let $\mathcal{R}_S$ be the distribution of the randomness for the simulator $\mathcal{S}$. Let

$$f(R, \rho) = \Pr\left[\begin{array}{l}(\mathcal{M}, s_1) = \mathcal{S}_1(1^\lambda; \rho); \\ (\mathcal{I}, s_2) = \mathcal{S}_2(s_1); \\ \boldsymbol{M} := (m_{i,j})_{i\in[n], j\in[k]} \leftarrow \mathcal{M}; \\ out = \mathcal{S}_3((m_{i,j})_{i\in\mathcal{I}, j\in[k]}, s_2);\end{array} \quad : \quad \mathcal{D}(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out; R) = 1\right]$$

where the probability is taken over the random choice of each $\boldsymbol{M}$. Then, we have

$$\begin{aligned}&\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 1] \\ =&\mathbb{E}_{R\leftarrow\mathcal{R}_D, \rho\leftarrow\mathcal{R}_S}(1 - (1 - f(R,\rho))^\kappa) \\ \le&\mathbb{E}_{R\leftarrow\mathcal{R}_D, \rho\leftarrow\mathcal{R}_S}(\kappa \cdot f(R,\rho)) \\ =&\kappa \cdot \mathbb{E}_{R\leftarrow\mathcal{R}_D, \rho\leftarrow\mathcal{R}_S} f(R,\rho) \\ =&\kappa \cdot \delta\end{aligned}$$

where the second inequality comes from the Bernoulli's inequality. □

Lemma 3.3. $\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 2] \le 1/4$.

Proof. This comes from the collision resistant property of the non-programmable random oracle, which is a random function whose output is not controlled by the simulator.

Assuming that H has been queried (either by the adversary, the distinguisher or the simulator) Q times, where Q is a polynomial. Then the probability that there exists two distinct queries x_1, x_2 s.t. $H(x_1) = H(x_2)$ does not exceed $\frac{Q^2}{2^h}$, which is negligible.

However, if the experiment outputs 2 with a non-negligible probability (e.g., $1/4$), then, via running the experiment, one can find $\iota \in [\kappa]$ that

1) $(\mathsf{PP}^{\iota-1}, (PK_i^{\iota-1}, C_{i,j}^{\iota-1})_{i\in[n],j\in[k]}) \neq (\mathsf{PP}^{\iota}, (PK_i^{\iota}, C_{i,j}^{\iota})_{i\in[n],j\in[k]})$

2) $H(\mathsf{PP}^{\iota-1}, (PK_i^{\iota-1}, C_{i,j}^{\iota-1})_{i\in[n],j\in[k]}) = H(\mathsf{PP}^{\iota}, (PK_i^{\iota}, C_{i,j}^{\iota})_{i\in[n],j\in[k]}) = (t[1], \ldots, t[h])$, where $t[i] = 1$ iff $i \in \mathcal{I}$ (otherwise, the experiment will output 1)

with a non-negligible probability, which makes a contradiction.

□

Finally, combining Lemma 3.1 to Lemma 3.3, we have

$$1 \leq 1/4 + \kappa \cdot \delta + 1/4$$

which implies $\delta \geq \frac{1}{2\kappa}$ and this completes the proof.

□

Remark 3.1. Theorem 3.1 claims that if the key length of a PKE scheme is not large enough, then it is impossible to prove its SIM-RSO$_k$-CPA security even in the non-programmable random oracle model. At first glance, this also rules out standard model achievability of RSO$_k$ security for PKE schemes with short keys. However, as stated in [BO13], impossibility result in non-programmable random oracle model does not extend to that in standard model naturally, since the adversary in the non-programmable random oracle model is also able to access the random oracle and thus is stronger than a standard model adversary.

Nonetheless, We can adapt the proof for Theorem 3.1 to achieve the same lower bound (i.e. $l > mk - 1$) in the standard model. More precisely, the revised proof is identical to proof of Theorem 3.1, except that we use a collision resistant hash function to replace the use of non-programmable random oracle. But the proof only works in the auxiliary input model, where all participants, including the adversary, the distinguisher, and the simulator, are given some common auxiliary input in the beginning. Here, the auxiliary input is a random key for the underlying collision resistant hash function.

4 RSO$_k$ Security $\not\Rightarrow$ RSO$_{k+1}$ Security

We present counterexamples that separate the RSO$_k$ security and the RSO$_{k+1}$ security in this section. More precisely, for any polynomial k, we construct a PKE scheme Π that is SIM-RSO$_k$-CCA secure in the standard model but is not SIM-RSO$_{k+1}$-CPA secure in the non-programmable random oracle model.

Let λ be the security parameter and let k be a positive integer that is polynomial in λ.

Let $\mathsf{E} = (\mathtt{E.Setup}, \mathtt{E.Gen}, \mathtt{E.Enc}, \mathtt{E.Dec})$ be a CPA Secure PKE scheme with a deterministic decryption algorithm and an additional verification algorithm

$\texttt{Ver}$. The algorithm $\texttt{Ver}$ takes as input a public parameter pp and a public key/secret key pair (pk, sk), and outputs a bit indicating if (pk, sk) is a valid key pair. Also, we require that E has the following two properties:

- **Verification Correctness.** Let $\mathsf{pp} \leftarrow \mathsf{E}.\texttt{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{E}.\texttt{Gen}(\mathsf{pp})$, then $\Pr[\mathsf{E}.\texttt{Ver}(\mathsf{pp}, pk, sk) = 1] = 1$.
- **Key Uniqueness.** For any pp and for any pk, $|\{sk \mid \mathsf{E}.\texttt{Ver}(\mathsf{pp}, pk, sk) = 1\}| \leq 1$.

It is easy to see that the well-known ElGamal encryption scheme satisfies this property.

Let $\mathsf{NIZK} = (\mathsf{NIZK}.\texttt{Gen}, \mathsf{NIZK}.\texttt{Prove}, \mathsf{NIZK}.\texttt{Verify})$ be an unbounded simulation-sound NIZK proof system for NP. In particular, we will use it to prove the following language:

$$\{(\mathsf{pp}, (pk_{\imath,\jmath}, c_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}) : \exists((p_\imath, r_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}), \\ (c_{\imath,\jmath} = \mathsf{E}.\texttt{Enc}(\mathsf{pp}, pk_{\imath,\jmath}, p_\imath; r_{\imath,\jmath}))_{\imath\in[k],\jmath\in\{0,1\}}\}$$

The PKE scheme $\Pi = (\texttt{Setup}, \texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ works as follows:

- $\texttt{Setup}$. On input a security parameter λ, the setup algorithm computes $\mathsf{pp} \leftarrow \mathsf{E}.\texttt{Setup}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{NIZK}.\texttt{Gen}(1^\lambda)$. The public parameter for Π is $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs})$.
- $\texttt{Gen}$. On input a public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs})$, the key generation algorithm first computes $(pk_{\imath,\jmath}, sk_{\imath,\jmath}) \leftarrow \mathsf{E}.\texttt{Gen}(\mathsf{pp})$ for $\imath \in [k]$ and $\jmath \in \{0,1\}$. Then it samples $s_1, \ldots, s_k \overset{\$}{\leftarrow} \{0,1\}$. The public key $PK = (pk_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}$ and the secret key $SK = (s_\imath, sk_{\imath,s_\imath})_{\imath\in[k]}$.
- $\texttt{Enc}$. On input a public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs})$, a public key $PK = (pk_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}$ and a message $m \in \{0,1\}$, the encryption algorithm first samples $p_1, \ldots, p_k$ uniformly at random from $\{0,1\}$ s.t. $m = p_1 \oplus p_2 \oplus \ldots \oplus p_k$. Then for $\imath \in [k], \jmath \in \{0,1\}$, it samples $r_{\imath,\jmath}$ randomly from the randomness space of E and computes $c_{\imath,\jmath} = \mathsf{E}.\texttt{Enc}(\mathsf{pp}, pk_{\imath,\jmath}, p_\imath; r_{\imath,\jmath})$. Finally, it computes $\pi \leftarrow \mathsf{NIZK}.\texttt{Prove}(\mathsf{crs}, (\mathsf{pp}, (pk_{\imath,\jmath}, c_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}), ((p_\imath, r_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}))$. The ciphertext is $C = ((c_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}, \pi)$.
- $\texttt{Dec}$. On input a public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs})$, a public key $PK = (pk_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}$, a secret key $SK = (s_\imath, sk_{\imath,s_\imath})_{\imath\in[k]}$ and a ciphertext $C = ((c_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}, \pi)$, the decryption algorithm first checks if π is valid and aborts with a decryption failure symbol $\perp$ if it is not the case. Otherwise, it computes $p_\imath = \mathsf{E}.\texttt{Dec}(\mathsf{pp}, pk_{\imath,s_\imath}, sk_{\imath,s_\imath}, c_{\imath,s_\imath})$ and outputs $m = p_1 \oplus \ldots \oplus p_k$.

Theorem 4.1. *If E is an CPA secure PKE scheme and NIZK is a simulation-sound NIZK proof system, then Π is* SIM-RSO$_k$-CCA *secure in the standard model.*

Theorem 4.2. *If E is a PKE scheme with deterministic decryption algorithm, verification correctness and key uniqueness, then Π is not* SIM-RSO$_{k+1}$-CPA *secure in the non-programmable random oracle model.*

Proofs of Theorem 4.1 and Theorem 4.2 are provided in Appendix A.1 and Appendix A.2 respectively.

Note that we can also prove that Π is not $\text{SIM-RSO}_{k+1}\text{-CPA}$ secure in the standard model, but similar to the setting discussed in Remark 3.1, we need to assume that all participants, including the adversary, the distinguisher, and the simulator, are given some common auxiliary input in the beginning.

5 RSO_k Secure PKE with (Nearly) Optimal Secret Key Length

In this section, we construct RSO_k secure PKE schemes with secret key length $l = k + O(\lambda)$. Here the ratio of secret key length to the messages number k is $\frac{l}{k} = 1 + o(1)$. As shown in Sec. 3, no PKE scheme can achieve RSO_k security if $l \leq k - 1$ (i.e., $\frac{l}{k} < 1$). Thus, our schemes are optimal in an asymptotic sense.

Next, in Sec. 5.1, we first construct an optimal $\text{SIM-RSO}_k\text{-CPA}$ secure scheme from the DDH assumption. Then in Sec. 5.2, we upgrade the scheme to achieve $\text{SIM-RSO}_k\text{-CCA}$ security by using a NIZK proof system.

5.1 $\text{SIM-RSO}_k\text{-CPA}$ Secure PKE with (Nearly) Optimal Secret Key Length

Let λ be the security parameter and let k be a positive integer that is polynomial in λ. Let $\mathcal{G}$ be a group generator algorithm that takes as input a security parameter λ and outputs a multiplicative cyclic group $\mathbb{G}$ of prime order q and a generator g of $\mathbb{G}$.

The PKE scheme $\Pi = (\texttt{Setup}, \texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ works as follows:

- `Setup`. On input a security parameter λ, the setup algorithm first generates $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda)$ and samples $a_0, a_1, \ldots a_k, b \xleftarrow{\$} \mathbb{Z}_q$. Then it computes $g_\imath = g^{a_\imath}$ for $\imath \in [0, k]$ and $h = g^b$. The public parameter for Π is $\mathsf{PP} = (\mathbb{G}, q, g, g_0, g_1, \ldots, g_k, h)$.
- `Gen`. On input a public parameter $\mathsf{PP} = (\mathbb{G}, q, g, g_0, g_1, \ldots, g_k, h)$, the key generation algorithm first samples $s_0 \xleftarrow{\$} \mathbb{Z}_q$ and $s_1, \ldots s_k \xleftarrow{\$} \{0, 1\}$ and sets the secret key $sk = (s_0, s_1, \ldots, s_k)$. Then it computes the public key $pk = \prod_{\imath \in [0,k]} g_\imath^{s_\imath}$.
- `Enc`. On input a public parameter $\mathsf{PP} = (\mathbb{G}, q, g, g_0, g_1, \ldots, g_k, h)$, a public key pk and a message $m \in \{0, 1\}$, the encryption algorithm first samples $w \xleftarrow{\$} \mathbb{Z}_q$. Then it computes $\boldsymbol{x} = (x_0, x_1, \ldots, x_k) = (g_0^w, g_1^w, \ldots, g_k^w)$, $K = pk^w$ and $C = K \cdot h^m$. The ciphertext $CT = (\boldsymbol{x}, C)$.
- `Dec`. On input a public parameter $\mathsf{PP} = (\mathbb{G}, q, g, g_0, g_1, \ldots, g_k, h)$, a secret key $sk = (s_0, s_1, \ldots, s_k)$ and a ciphertext $CT = (x_0, x_1, \ldots, x_k, C)$, the decryption algorithm first computes $K' = \prod_{\imath \in [0,k]} x_\imath^{s_\imath}$. Then it outputs 0 if $C = K'$ and outputs 1 if $C = K' \cdot h$. Otherwise, it outputs a decryption failure symbol $\perp$.

Security. Security of $\mathsf{\Pi}$ is guaranteed by the following theorem. We put the proof of Theorem 5.1 in Sec. 5.3.

Theorem 5.1. *Assuming the DDH assumption holds in group* $\mathbb{G}$, $\mathsf{\Pi}$ *is a PKE scheme with* SIM-RSO$_k$-CPA *security.*

Key Length. The secret key length of $\mathsf{\Pi}$ is $k + \log q$, where $\log q$ is determined by the security parameter λ and is independent of the parameter k. For example, if we instantiate the scheme with an elliptic curve group and hope to achieve a 80-bit security, then we can fix $\log q = 160$. In this case, the ratio of key length to messages number k is $\frac{k+\log q}{k} = 1 + \frac{160}{k} = 1 + o(1)$.

5.2 SIM-RSO$_k$-CCA Secure PKE with (Nearly) Optimal Secret Key Length

Let λ be the security parameter and let k be a positive integer that is polynomial in λ. Let $\mathsf{\Pi}' = (\mathsf{\Pi}'.\mathtt{Setup}, \mathsf{\Pi}'.\mathtt{Gen}, \mathsf{\Pi}'.\mathtt{Enc}, \mathsf{\Pi}'.\mathtt{Dec})$ be a SIM-RSO$_k$-CPA secure PKE scheme. Let $\mathsf{E} = (\mathsf{E}.\mathtt{Setup}, \mathsf{E}.\mathtt{Gen}, \mathsf{E}.\mathtt{Enc}, \mathsf{E}.\mathtt{Dec})$ be a CPA-secure PKE scheme. Let $\mathsf{NIZK} = (\mathsf{NIZK}.\mathtt{Gen}, \mathsf{NIZK}.\mathtt{Prove}, \mathsf{NIZK}.\mathtt{Verify})$ be a an unbounded simulation-sound NIZK proof for NP. In particular, we will use it to prove the following language:

$$\{(\mathsf{pp}_1, pk_1, c_1, \mathsf{pp}_2, pk_2, c_2) : \exists (m, r_1, r_2),\\ c_1 = \mathsf{\Pi}'.\mathtt{Enc}(\mathsf{pp}_1, pk_1, m; r_1) \wedge c_2 = \mathsf{E}.\mathtt{Enc}(\mathsf{pp}_2, pk_2, m; r_2)\}$$

The PKE scheme $\mathsf{\Pi} = (\mathtt{Setup}, \mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ works as follows:

- $\mathtt{Setup}$. On input a security parameter λ, the setup algorithm computes $\mathsf{pp} \leftarrow \mathsf{\Pi}'.\mathtt{Setup}(1^\lambda)$, $\tilde{\mathsf{pp}} \leftarrow \mathsf{E}.\mathtt{Setup}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{NIZK}.\mathtt{Gen}(1^\lambda)$. Also, it generates $(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{E}.\mathtt{Gen}(\tilde{\mathsf{pp}})$. The public parameter for $\mathsf{\Pi}$ is $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs}, \tilde{\mathsf{pp}}, \tilde{pk})$.
- $\mathtt{Gen}$. On input a public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs}, \tilde{\mathsf{pp}}, \tilde{pk})$, the key generation algorithm computes $(pk, sk) \leftarrow \mathsf{\Pi}'.\mathtt{Gen}(\mathsf{pp})$. The public key $PK = pk$ and the secret key $SK = sk$.
- $\mathtt{Enc}$. On input a public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs}, \tilde{\mathsf{pp}}, \tilde{pk})$, a public key $PK = pk$ and a message m, the encryption algorithm first samples r, $\tilde{r}$ randomly from the encryption randomness space of $\mathsf{\Pi}'$ and E respectively. Then it computes $c = \mathsf{\Pi}'.\mathtt{Enc}(\mathsf{pp}, pk, m; r)$, $\tilde{c} = \mathsf{E}.\mathtt{Enc}(\tilde{\mathsf{pp}}, \tilde{pk}, m; \tilde{r})$ and $\pi \leftarrow \mathsf{NIZK}.\mathtt{Prove}(\mathsf{crs}, (\mathsf{pp}, pk, c, \tilde{\mathsf{pp}}, \tilde{pk}, \tilde{c}), (m, r, \tilde{r}))$. The ciphertext is $C = (c, \tilde{c}, \pi)$.
- $\mathtt{Dec}$. On input a public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs}, \tilde{\mathsf{pp}}, \tilde{pk})$, a public key $PK = pk$, a secret key $SK = sk$ and a ciphertext $C = (c, \tilde{c}, \pi)$, the decryption algorithm first checks if π is valid and aborts with a decryption failure symbol $\perp$ if it is not the case. Otherwise, it outputs $m \leftarrow \mathsf{\Pi}'.\mathtt{Dec}(\mathsf{pp}, pk, sk, c)$.

Security. Security of Π is guaranteed by the following theorem. We put the proof of Theorem 5.2 in Sec. 5.4.

Theorem 5.2. *If Π' is a* SIM-RSO$_k$-CPA *secure PKE scheme,* E *is a CPA-secure PKE scheme and* NIZK *is an unbounded simulation-sound NIZK proof, then Π is a PKE scheme with* SIM-RSO$_k$-CCA *security.*

Key Length. If we instantiate the underlying SIM-RSO$_k$-CPA secure PKE scheme Π' with the one we constructed in Sec. 5.1, then we can obtain a SIM-RSO$_k$-CCA secure PKE scheme Π, where the ratio of key length to messages number k is also $\frac{k+\log q}{k} = 1 + o(1)$.

5.3 Proof of Theorem 5.1

Proof. We provide the proof of Theorem 5.1 in this section.

Let K and K' be the random variables used in generating and decrypting the same ciphertext $(x_0, x_1, \ldots, x_k, C)$ respectively. It is easy to see that the decryption algorithm can recover the correct message iff $K = K'$. As we have

$$K = pk^w = (\prod_{\imath\in[0,k]} g_\imath^{s_\imath})^w = \prod_{\imath\in[0,k]} g_\imath^{w\cdot s_\imath} = \prod_{\imath\in[0,k]} (g_\imath^w)^{s_\imath} = \prod_{\imath\in[0,k]} x_\imath^{s_\imath} = K'$$

the correctness holds.

Next, we focus on the SIM-RSO$_k$-CPA security of Π. First, for any polynomial n, any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, and any distinguisher $\mathcal{D}$, we design the simulator $\mathcal{S}$ for $\mathcal{A}$, which works as in Figure 4.

Next, we prove that output of the simulator $\mathcal{S}$ is indistinguishable from output of the adversary $\mathcal{A}$ in a real game. We argue this via defining the following games:

- **Game 0.** This is the real experiment $\mathtt{Exp}_{\Pi,\mathcal{A},n}^{\mathrm{RSO_k-CPA-real}}$. In particular, the challenger interacts with the adversary as follows:
  1. On input a security parameter, the challenger first generates $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda)$ and samples $a_0, a_1, \ldots a_k, b \xleftarrow{\$} \mathbb{Z}_q$. Then it computes $g_\imath = g^{a_\imath}$ for $\imath \in [0,k]$, $h = g^b$, and sets $\mathsf{PP} = (\mathbb{G}, q, g, g_0, g_1, \ldots, g_k, h)$.
  2. Then, for $i \in [n]$, it samples $s_{i,0} \xleftarrow{\$} \mathbb{Z}_q$, $s_{i,1}, \ldots s_{i,k} \xleftarrow{\$} \{0,1\}$ and computes the public key $pk_i = \prod_{\imath\in[0,k]} g_\imath^{s_{i,\imath}}$.
  3. Next, the challenger sends $\mathsf{PP}, (pk_i)_{i\in[n]}$ to $\mathcal{A}$ and receives a distribution $\mathcal{M}$ from the adversary.
  4. Then, the challenger samples a matrix of messages $\boldsymbol{M} := (m_{i,j})_{i\in[n],j\in[k]} \leftarrow \mathcal{M}$ and for each $(i,j) \in [n]\times[k]$, it generates a challenge ciphertext for $m_{i,j}$ as follows:
     (a) Samples $w_{i,j} \xleftarrow{\$} \mathbb{Z}_q$.
     (b) Computes $\boldsymbol{x}_{i,j} = (x_{i,j,0}, x_{i,j,1}, \ldots, x_{i,j,k}) = (g_0^{w_{i,j}}, g_1^{w_{i,j}}, \ldots, g_k^{w_{i,j}})$.
     (c) Computes $C_{i,j} = pk_i^{w_{i,j}} \cdot h^{m_{i,j}}$.

```
$\mathcal{S}_1(1^\lambda)$:
  $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda)$
  For $\imath \in [0, k]$:
    $a_\imath \xleftarrow{\$} \mathbb{Z}_q$
    $g_\imath = g^{a_\imath}$
  $b \xleftarrow{\$} \mathbb{Z}_q$
  $h = g^b$
  $\mathsf{PP} = (\mathbb{G}, q, g, g_0, g_1, \ldots, g_k, h)$
  For $i \in [n]$:
    $s'_{i,0} \xleftarrow{\$} \mathbb{Z}_q$
    For $\imath \in [k]$:
      $s'_{i,\imath} \xleftarrow{\$} \{0, 1\}$
    $pk_i = \prod_{\imath \in [0,k]} g_\imath^{s'_{i,\imath}}$
  $(\mathcal{M}, \mathfrak{s}'_1) \leftarrow \mathcal{A}_1(\mathsf{PP}, (pk_i)_{i \in [n]})$
  $\mathfrak{s} = ((a_\imath)_{\imath \in [0,k]}, b, (s'_{i,\imath})_{i \in [n], \imath \in [0,k]})$
  $\mathfrak{s}_1 = (\mathfrak{s}'_1, \mathsf{PP}, (pk_i)_{i \in [n]}, \mathfrak{s})$
  Output $(\mathcal{M}, \mathfrak{s}_1)$
```

```
$\mathcal{S}_2(\mathfrak{s}_1)$:
  For $i \in [n], j \in [k]$:
    $w_{i,j} \xleftarrow{\$} \mathbb{Z}_q$
    If $s'_{i,j} = 1$: $\alpha_{i,j} = 1$
    Otherwise : $\alpha_{i,j} = -1$
    For $\imath \in [0, k] \wedge \imath \neq j$: $x_{i,j,\imath} = g_\imath^{w_{i,j}}$
    $x_{i,j,j} = g_j^{w_{i,j}} \cdot h^{\alpha_{i,j}}$
    $\boldsymbol{x}_{i,j} = (x_{i,j,0}, \ldots, x_{i,j,k})$
    $C_{i,j} = \prod_{\imath \in [0,k]} x_{i,j,\imath}^{s'_{i,\imath}}$
    $CT_{i,j} = (\boldsymbol{x}_{i,j}, C_{i,j})$
  $(\mathcal{I}, \mathfrak{s}'_2) \leftarrow \mathcal{A}_2((CT_{i,j})_{i \in [n], j \in [k]}, \mathfrak{s}'_1)$
  $\mathfrak{s}_2 = (\mathfrak{s}_1, \mathfrak{s}'_2)$
  Output $(\mathcal{I}, \mathfrak{s}_2)$
```

```
$\mathcal{S}_3((m_{i,j})_{i \in \mathcal{I}, j \in [k]}, \mathfrak{s}_2)$:
  For $i \in \mathcal{I}$:
    For $j \in [k]$:
      If $m_{i,j} = 0$ : $s_{i,j} = s'_{i,j}$
      Otherwise: $s_{i,j} = 1 - s'_{i,j}$
    $s_{i,0} = s'_{i,0} + a_0^{-1} \sum_{\imath \in [k]} (a_\imath \cdot (s'_{i,\imath} - s_{i,\imath}))$
    $sk_i = (s_{i,\imath})_{\imath \in [0,k]}$
  $out \leftarrow \mathcal{A}_3((sk_i, m_{i,j})_{i \in \mathcal{I}, j \in [k]}, \mathfrak{s}'_2)$
  Output $out$
```

Fig. 4 The simulator $\mathcal{S}$ for $\mathcal{A}$ in proving SIM-RSO$_k$-CPA security of Π.

(d) Sets $CT_{i,j} = (\boldsymbol{x}_{i,j}, C_{i,j})$.

5. Next, the challenger sends all challenge ciphertexts to $\mathcal{A}$ and receives a set $\mathcal{I} \subseteq [n]$ from the adversary.
6. Then, the challenger sets $sk_i = (s_{i,0}, s_{i,1}, \ldots, s_{i,k})$ for $i \in \mathcal{I}$ and sends $(sk_i, m_{i,j})_{i \in \mathcal{I}, j \in [k]}$ to $\mathcal{A}$.
7. Finally, on receiving $\mathcal{A}$'s output out, the challenger outputs $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$.

- **Game 1.** This is identical to Game 0 except that in step 4, the challenger computes new variables $(s'_{i,j}, \alpha_{i,j})_{i \in [n], j \in [k]}$. More precisely, for $i \in [n], j \in [k]$, it sets $s'_{i,j} = s_{i,j}$ if $m_{i,j} = 0$ and sets $s'_{i,j} = 1 - s_{i,j}$ otherwise. Besides, it sets $\alpha_{i,j} = 1$ if $s'_{i,j} = 1$ and sets $\alpha_{i,j} = -1$ otherwise.
- **Game 2.** This is identical to Game 1 except that the challenger changes the way to generate $C_{i,j}$. More precisely, for each $i \in [n], j \in [k]$, the challenger computes $C_{i,j} = (\prod_{\imath \in [0,k]} x_{i,j,\imath}^{s_{i,\imath}}) \cdot h^{m_{i,j}}$.
- **Game 3.** This is identical to Game 2 except that the j-th element in $\boldsymbol{x}_{i,j}$ (i.e., $x_{i,j,j}$) is generated dishonestly. More precisely, for each $i \in [n], j \in [k]$, it samples $x_{i,j,j} \xleftarrow{\$} \mathbb{G}$.

- **Game 4.** This is identical to Game 3 except that the challenger changes the way to generate $x_{i,j,j}$. More precisely, for each $i \in [n], j \in [k]$, it samples $x'_{i,j,j} \xleftarrow{\$} \mathbb{G}$ and computes $x_{i,j,j} = x'_{i,j,j} \cdot h^{\alpha_{i,j}}$.
- **Game 5.** This is identical to Game 4 except that the challenger changes the way to generate $x_{i,j,j}$. More precisely, for each $i \in [n], j \in [k]$, it computes $x'_{i,j,j} = g_j^{w_{i,j}}$ and $x_{i,j,j} = x'_{i,j,j} \cdot h^{\alpha_{i,j}}$.
- **Game 6.** This is identical to Game 5 except that the challenger changes the way to generate $C_{i,j}$. More precisely, in step 4, the challenger sets $s'_{i,0} = s_{i,0} + a_0^{-1} \sum_{\imath \in [k]} (a_\imath \cdot (s_{i,\imath} - s'_{i,\imath}))$ and for each $i \in [n], j \in [k]$, it computes $C_{i,j} = \prod_{\imath \in [0,k]} x_{i,j,\imath}^{s'_{i,\imath}}$.
- **Game 7.** This is identical to Game 6 except that the challenger changes the order in generating $s'_{i,j}$ and $s_{i,j}$:
  - In step 2, it samples $s'_{i,0} \xleftarrow{\$} \mathbb{Z}_q$ and $s'_{i,\imath} \xleftarrow{\$} \{0,1\}$ for $i \in [n], \imath \in [k]$ and computes $pk_i = \prod_{\imath \in [0,k]} g_\imath^{s'_{i,\imath}}$ for $i \in [n]$.
  - In step 4, for $i \in [n], j \in [k]$, it sets $s_{i,j} = s'_{i,j}$ if $m_{i,j} = 0$ and sets $s_{i,j} = 1 - s'_{i,j}$ otherwise. Also, it sets $s_{i,0} = s'_{i,0} + a_0^{-1} \sum_{\imath \in [k]} (a_\imath \cdot (s'_{i,\imath} - s_{i,\imath}))$ for $i \in [n]$.

Let $\mathfrak{p}_\iota$ be the probability that $\mathcal{D}$ outputs 1 when taking the output of Game ι as input, then we have $\mathfrak{p}_0 = \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{A},n}^{\mathrm{RSO_k-CPA-real}}(\lambda)) = 1]$. Also, it is easy to see that output of Game 7 is exactly the output of the ideal experiment, so, we have $\mathfrak{p}_7 = \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{S},n}^{\mathrm{RSO_k-CPA-ideal}}(\lambda)) = 1]$. Next, we prove that $\mathfrak{p}_0 - \mathfrak{p}_7$ is negligible via showing that $\mathfrak{p}_\iota - \mathfrak{p}_{\iota+1}$ is negligible for all $\iota \in [0,6]$.

Lemma 5.1. $|\mathfrak{p}_0 - \mathfrak{p}_1| = 0$.

Proof. Game 0 and Game 1 are identical except that in Game 1, the challenger generates some variables that are not used in this game. This will not affect the output of the game. □

Lemma 5.2. $|\mathfrak{p}_1 - \mathfrak{p}_2| = 0$.

Proof. In Game 1 and Game 2, each $C_{i,j}$ is computed in different ways. But as

$$pk_i^{w_{i,j}} = (\prod_{\imath \in [0,k]} g_\imath^{s_{i,\imath}})^{w_{i,j}} = \prod_{\imath \in [0,k]} g_\imath^{s_{i,\imath} \cdot w_{i,j}} = \prod_{\imath \in [0,k]} (g_\imath^{w_{i,j}})^{s_{i,\imath}} = \prod_{\imath \in [0,k]} x_{i,j,\imath}^{s_{i,\imath}}$$

the computation results are identical and thus outputs of these two games are identically distributed. □

Lemma 5.3. $|\mathfrak{p}_2 - \mathfrak{p}_3| \leq \mathtt{negl}(\lambda)$.

Proof. Indistinguishability between Game 2 and Game 3 comes from the DDH assumption by a standard hybrid argument.

In particular, for some fixed $i, j \in [n] \times [k]$, to show that $x_{i,j,j}$ is sampled from two computationally indistinguishable distributions in Game 2 and Game

3, we consider a DDH challenge $(g, \mathfrak{g}_1, \mathfrak{g}_2, \mathfrak{g}_3) = (g, g^x, g^y, g^z)$, where $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}_q$. The reduction sets $g_j = \mathfrak{g}_1$, $g^{w_{i,j}} = \mathfrak{g}_2$, $x_{i,j,j} = \mathfrak{g}_3$ Then, it simulates the view for $\mathcal{A}$ (as in Game 2 and Game 3) with them. Note that the exact value of x and y is not needed in the simulation since 1) the challenger does not use a_j in both Game 2 and Game 3 and 2) without $w_{i,j}$, the challenger can compute $x_{i,j,\imath} = \mathfrak{g}_2^{a_\imath}$ for $\imath \in [0,k] \backslash \{j\}$. It is easy to see that if $z = xy$, then $x_{i,j,j} = g_j^{w_{i,j}}$ as in Game 2, and if $z \xleftarrow{\$} \mathbb{Z}_q$, then $x_{i,j,j} \xleftarrow{\$} \mathbb{Z}_q$ as in Game 3. Therefore, indistinguishability between Game 2 and Game 3 is guaranteed assuming the hardness of the DDH assumption. □

Lemma 5.4. $|\mathfrak{p}_3 - \mathfrak{p}_4| = 0$.

Proof. Since in Game 3, $x_{i,j,j} \xleftarrow{\$} \mathbb{G}$, it will not change its distribution if we additionally multiply it with $h^{\alpha_{i,j}}$. Therefore, outputs of these two games are identically distributed. □

Lemma 5.5. $|\mathfrak{p}_4 - \mathfrak{p}_5| \le \mathtt{negl}(\lambda)$.

Proof. Similar to the proof of Lemma 5.3, indistinguishability between Game 4 and Game 5 comes from the DDH assumption by a standard hybrid argument. □

Lemma 5.6. $|\mathfrak{p}_5 - \mathfrak{p}_6| = 0$.

Proof. In Game 5 and Game 6, each $C_{i,j}$ is computed in different ways. But as

$$\begin{aligned}
&(\prod_{\imath \in [0,k]} x_{i,j,\imath}^{s_{i,\imath}}) \cdot h^{m_{i,j}} \\
=&(\prod_{\imath \in [0,k]} g_\imath^{w_{i,j} \cdot s_{i,\imath}}) \cdot h^{\alpha_{i,j} \cdot s_{i,j}} \cdot h^{m_{i,j}} \\
=&(g^{w_{i,j} \cdot (\sum_{\imath \in [0,k]} a_\imath \cdot s_{i,\imath})}) \cdot h^{\alpha_{i,j} \cdot s_{i,j} + m_{i,j}} \\
=&(g^{w_{i,j} \cdot (\sum_{\imath \in [0,k]} a_\imath \cdot s'_{i,\imath})}) \cdot h^{\alpha_{i,j} \cdot s_{i,j} + m_{i,j}} \\
=&(g^{w_{i,j} \cdot (\sum_{\imath \in [0,k]} a_\imath \cdot s'_{i,\imath})}) \cdot h^{\alpha_{i,j} \cdot s'_{i,j}} \\
=&(\prod_{\imath \in [0,k]} g_\imath^{w_{i,j} \cdot s'_{i,\imath}}) \cdot h^{\alpha_{i,j} \cdot s'_{i,j}} \\
=& \prod_{\imath \in [0,k]} x_{i,j,\imath}^{s'_{i,\imath}}
\end{aligned}$$

the computation results are identical and thus outputs of these two games are identically distributed.

Here, the first and the last equalities come from the fact that $x_{i,j,\imath} = g_\imath^{w_{i,j}}$ for $\imath \neq j$ and that $x_{i,j,j} = g_j^{w_{i,j}} \cdot h^{\alpha_{i,j}}$. Also, the third equality comes from the fact that $s'_{i,0} = s_{i,0} + a_0^{-1} \sum_{\imath \in [k]} (a_\imath \cdot (s_{i,\imath} - s'_{i,\imath}))$, which implies that $\sum_{\imath \in [0,k]} (a_\imath \cdot s'_{i,\imath}) =$

$\sum_{\imath\in[0,k]}(a_\imath \cdot s_{i,\imath})$. For the fourth equality, if $m_{i,j} = 0$, then $s_{i,j} = s'_{i,j}$ and thus $\alpha_{i,j} \cdot s_{i,j} + 0 = \alpha_{i,j} \cdot s'_{i,j}$; if $m_{i,j} = 1$, then either $s_{i,j} = 1, s'_{i,j} = 0$ or $s_{i,j} = 0, s'_{i,j} = 1$, and in both cases, $\alpha_{i,j} \cdot (s'_{i,j} - s_{i,j}) = 1$, which implies that $\alpha_{i,j} \cdot s_{i,j} + 1 = \alpha_{i,j} \cdot s'_{i,j}$. □

Lemma 5.7. $|\mathfrak{p}_6 - \mathfrak{p}_7| = 0$.

Proof. First, in both Game 6 and Game 7, each pk_i is a random element in $\mathbb{G}$, thus the adversary's views are identical in both games until step 4, where $(s_{i,\imath}, s'_{i,\imath})_{i\in[n],\imath\in[0,k]}$ are sampled in different ways.

In step 4, fixing the challenge messages $m_{i,j}$, then in both games the random variables $(s_{i,\imath}, s'_{i,\imath})_{i\in[n],\imath\in[0,k]}$ are randomly distributed in $\mathbb{Z}_q \times \mathbb{Z}_q \times \{0,1\}^{2k}$ with the restriction that for any $i \in [n]$:

$$\begin{cases} \sum_{\imath\in[0,k]}(a_\imath \cdot s'_{i,\imath}) = \sum_{\imath\in[0,k]}(a_\imath \cdot s_{i,\imath}) = \log_g pk_i \\ \forall \imath \in [k], s_{i,\imath} + s'_{i,\imath} = m_{i,j} \end{cases}$$

Therefore, they are identically distributed and that completes the proof of Lemma 5.7. □

Combining Lemma 5.1 to Lemma 5.7, we have $\mathfrak{p}_0 - \mathfrak{p}_7$ negligible and this completes the proof.

□

5.4 Proof of Theorem 5.2

Proof. We provide the proof of Theorem 5.2 in this section.

Correctness of Π comes from correctness of Π' and completeness of NIZK directly.

Next, we focus on the SIM-RSO$_k$-CCA security of Π. First, for any polynomial n, any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, we define an auxiliary adversary $\mathcal{B}$ for Π' as in Figure 5. Since Π' is a SIM-RSO$_k$-CPA secure PKE scheme, there exists a simulator $\mathcal{S}' = (\mathcal{S}'_1, \mathcal{S}'_2, \mathcal{S}'_3)$ for $\mathcal{B}$ such that the output of $\mathcal{S}'$ is indistinguishable from the output of $\mathcal{B}$ in a real RSO$_k$-CPA game. Then we define the simulator $\mathcal{S}$ for $\mathcal{A}$ as $\mathcal{S} = \mathcal{S}' = (\mathcal{S}'_1, \mathcal{S}'_2, \mathcal{S}'_3)$.

Next, we prove that output of the simulator $\mathcal{S}$ is indistinguishable from output of the adversary $\mathcal{A}$ in a real RSO$_k$-CCA game. We argue this via defining the following games:

- **Game 0.** This is the real experiment $\mathtt{Exp}^{\mathrm{RSO_k-CCA-real}}_{\Pi,\mathcal{A},n}$. In particular, the challenger interacts with the adversary as follows:
  1. On input a security parameter, the challenger first computes $\mathsf{pp} \leftarrow \Pi'.\mathtt{Setup}(1^\lambda)$, $\tilde{\mathsf{pp}} \leftarrow \mathsf{E}.\mathtt{Setup}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{NIZK}.\mathtt{Gen}(1^\lambda)$. Also, it generates $(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{E}.\mathtt{Gen}(\tilde{\mathsf{pp}})$. Then, it sets the public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs}, \tilde{\mathsf{pp}}, \tilde{pk})$.
  2. Then, for $i \in [n]$, it computes $(pk_i, sk_i) \leftarrow \Pi'.\mathtt{Gen}(\mathsf{pp})$.

```
$\mathcal{B}_1(\mathsf{pp}, (pk_i)_{i\in[n]})$:
    $\tilde{\mathsf{pp}} \leftarrow \mathsf{E}.\mathtt{Setup}(1^\lambda)$
    $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.\mathtt{S}_1(1^\lambda)$
    $(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{E}.\mathtt{Gen}(\tilde{\mathsf{pp}})$
    $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs}, \tilde{\mathsf{pp}}, \tilde{pk})$
    $\mathcal{C} = \emptyset$
    $(\mathcal{M}, \mathfrak{s}_1') \leftarrow \mathcal{A}_1^{\mathtt{DecB}}(\mathsf{PP}, (pk_i)_{i\in[n]})$
    $\mathfrak{s}_1 = (\mathfrak{s}_1', \mathsf{PP}, (pk_i)_{i\in[n]}, \mathsf{td}, \tilde{sk})$
    Output $(\mathcal{M}, \mathfrak{s}_1)$
```

```
$\mathtt{DecB}(i, C)$:
    If $(i, C) \in \mathcal{C}$: Return $\perp$
    Parse $C = (c, \tilde{c}, \pi)$
    $x = (\mathsf{pp}, pk_i, c, \tilde{\mathsf{pp}}, \tilde{pk}, \tilde{c})$
    If $\mathsf{NIZK}.\mathtt{Verify}(\mathsf{crs}, x, \pi) = 0$: Return $\perp$
    Return $\mathsf{E}.\mathtt{Dec}(\tilde{\mathsf{pp}}, \tilde{pk}, \tilde{sk}, \tilde{c})$
```

```
$\mathcal{B}_2((c_{i,j})_{i\in[n],j\in[k]}, \mathfrak{s}_1)$:
    For $i \in [n], j \in [k]$:
        $\tilde{c}_{i,j} \leftarrow \mathsf{E}.\mathtt{Enc}(\tilde{\mathsf{pp}}, \tilde{pk}, 0)$
        $x_{i,j} = (\mathsf{pp}, pk_i, c_{i,j}, \tilde{\mathsf{pp}}, \tilde{pk}, \tilde{c}_{i,j})$
        $\pi_{i,j} \leftarrow \mathsf{NIZK}.\mathtt{S}_2(\mathsf{crs}, \mathsf{td}, x_{i,j})$
        $C_{i,j} = (c_{i,j}, \tilde{c}_{i,j}, \pi_{i,j})$
    $\mathcal{C} = \{(i, C_{i,j}) \mid i \in [n], j \in [k]\}$
    $(\mathcal{I}, \mathfrak{s}_2') \leftarrow \mathcal{A}_2^{\mathtt{DecB}}((C_{i,j})_{i\in[n],j\in[k]}, \mathfrak{s}_1')$
    $\mathfrak{s}_2 = (\mathfrak{s}_1, \mathfrak{s}_2')$
    Output $(\mathcal{I}, \mathfrak{s}_2)$
```

```
$\mathcal{B}_3((sk_i, m_{i,j})_{i\in\mathcal{I},j\in[k]}, \mathfrak{s}_2)$:
    $out \leftarrow \mathcal{A}_3^{\mathtt{DecB}}((sk_i, m_{i,j})_{i\in\mathcal{I},j\in[k]}, \mathfrak{s}_2')$
    Output $out$
```

Fig. 5 The adversary $\mathcal{B}$ for Π'.

3. Next, the challenger sends $\mathsf{PP}, (pk_i)_{i\in[n]}$ to $\mathcal{A}$ and answers $\mathcal{A}$'s decryption oracle queries as follows:
   (a) On input a pair (i, C), where $C = (c, \tilde{c}, \pi)$, the challenger first checks if π is valid and returns an error symbol $\perp$ if π is not valid.
   (b) Otherwise, it computes $m \leftarrow \Pi'.\mathtt{Dec}(\mathsf{pp}, pk_i, sk_i, c)$.
   (c) Finally, it returns m to $\mathcal{A}$.
4. The adversary will send a distribution $\mathcal{M}$ to the challenger after querying the decryption oracle a few times. Then, the challenger samples a matrix of messages $\boldsymbol{M} \coloneqq (m_{i,j})_{i\in[n],j\in[k]} \leftarrow \mathcal{M}$ and for each $(i, j) \in [n] \times [k]$, it generates a challenge ciphertext for $m_{i,j}$ as follows:
   (a) Samples $r_{i,j}, \tilde{r}_{i,j}$ randomly from the encryption randomness space of Π' and E respectively.
   (b) Computes $c_{i,j} = \Pi'.\mathtt{Enc}(\mathsf{pp}, pk_i, m_{i,j}; r_{i,j})$.
   (c) Computes $\tilde{c}_{i,j} = \mathsf{E}.\mathtt{Enc}(\tilde{\mathsf{pp}}, \tilde{pk}, m_{i,j}; \tilde{r}_{i,j})$.
   (d) Computes $\pi_{i,j} \leftarrow \mathsf{NIZK}.\mathtt{Prove}(\mathsf{crs}, (\mathsf{pp}, pk_i, c_{i,j}, \tilde{\mathsf{pp}}, \tilde{pk}, \tilde{c}_{i,j}), (m_{i,j}, r_{i,j}, \tilde{r}_{i,j}))$.
   (e) Sets $C_{i,j} = (c_{i,j}, \tilde{c}_{i,j}, \pi_{i,j})$.
5. Next, the challenger sends all challenge ciphertexts to $\mathcal{A}$ and answers $\mathcal{A}$'s decryption oracle queries as follows:
   (a) On input a pair (i, C), the challenger first checks if $C = C_{i,j}$ for some $j \in [k]$. It returns $\perp$ if this is the case.
   (b) Otherwise, the challenger parses $C = (c, \tilde{c}, \pi)$ and checks if π is valid. It returns an error symbol $\perp$ if π is not valid.
   (c) Otherwise, it computes $m \leftarrow \Pi'.\mathtt{Dec}(\mathsf{pp}, pk_i, sk_i, c)$.
   (d) Finally, it returns m to $\mathcal{A}$.

6. The adversary will send a set $\mathcal{I} \subseteq [n]$ to the challenger after querying the decryption oracle a few times. Then, the challenger sends $(sk_i, m_{i,j})_{i\in\mathcal{I}, j\in[k]}$ to $\mathcal{A}$. The challenger will answer $\mathcal{A}$'s decryption queries exactly as in step 5.
7. Finally, on receiving $\mathcal{A}$'s output out, the challenger outputs $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$.

- **Game 1.** This is identical to Game 0 except that when generating the common reference string and proofs, the challenger uses the simulator of NIZK instead of generating them honestly. More precisely, in the first step, the challenger computes $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.S}_1(1^\lambda)$ and in step 4, the challenger computes $\pi_{i,j} \leftarrow \mathsf{NIZK.S}_2(\mathsf{crs}, \mathsf{td}, (\mathsf{pp}, pk_i, c_{i,j}, \tilde{\mathsf{pp}}, \tilde{pk}, \tilde{c}_{i,j}))$.
- **Game 2.** This is identical to Game 1 except that the challenger changes the way to generate challenge ciphertexts. More precisely, for each $i \in [n]$, $j \in [k]$, the challenger computes $\tilde{c}_{i,j} \leftarrow \mathsf{E}.\mathtt{Enc}(\tilde{\mathsf{pp}}, \tilde{pk}, 0)$.
- **Game 3.** This is identical to Game 2 except that the challenger changes the way to answer decryption queries. More precisely, for a ciphertext $(c, \tilde{c}, \pi)$, it returns $\mathsf{E}.\mathtt{Dec}(\tilde{\mathsf{pp}}, \tilde{pk}, \tilde{sk}, \tilde{c})$ in the last step of the decryption oracle.
- **Game 4.** In Game 4, the challenger proceeds as follows:
  1. $(\mathcal{M}, s_1) \leftarrow \mathcal{S}_1'(1^\lambda)$
  2. $\boldsymbol{M} := (m_{i,j})_{i\in[n], j\in[k]} \leftarrow \mathcal{M}$
  3. $(\mathcal{I}, s_2) \leftarrow \mathcal{S}_2'(s_1)$
  4. $out \leftarrow \mathcal{S}_3'((m_{i,j})_{i\in\mathcal{I}, j\in[k]}, s_2)$
  5. Return $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$

Let $\mathfrak{p}_\alpha$ be the probability that $\mathcal{D}$ outputs 1 when taking the output of Game α as input, then we have

$$\mathfrak{p}_0 = \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{A},n}^{\mathrm{RSO_k}-\mathtt{CCA}-\mathtt{real}}(\lambda)) = 1]$$

Besides, we can view Game 4 as the ideal experiment $\mathtt{Exp}_{\Pi,\mathcal{S},n}^{\mathrm{RSO_k}-\mathtt{CCA}-\mathtt{ideal}}$ (recall that $\mathcal{S} = \mathcal{S}' = (\mathcal{S}_1', \mathcal{S}_2', \mathcal{S}_3')$), so we have

$$\mathfrak{p}_4 = \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{S},n}^{\mathrm{RSO_k}-\mathtt{CCA}-\mathtt{ideal}}(\lambda)) = 1]$$

Next, we prove that $\mathfrak{p}_0 - \mathfrak{p}_4$ is negligible via showing that $\mathfrak{p}_\alpha - \mathfrak{p}_{\alpha+1}$ is negligible for all $\alpha \in [0, 3]$.

Lemma 5.8. $|\mathfrak{p}_0 - \mathfrak{p}_1| \leq \mathtt{negl}(\lambda)$.

Proof. This comes from the unbounded zero-knowledge property of NIZK directly. □

Lemma 5.9. $|\mathfrak{p}_1 - \mathfrak{p}_2| \leq \mathtt{negl}(\lambda)$.

Proof. This comes from the CPA-security of E directly. □

Lemma 5.10. $|\mathfrak{p}_2 - \mathfrak{p}_3| \leq \mathtt{negl}(\lambda)$.

Proof. This comes from the fact that for any ciphertext $(c, \tilde{c}, \pi)$ with a valid π, $\mathsf{E.Dec}(\tilde{\mathsf{pp}}, \tilde{pk}, \tilde{sk}, \tilde{c}) = \Pi'.\mathsf{Dec}(\mathsf{pp}, pk_i, sk_i, c)$ with all but negligible probability, which is guaranteed by the unbounded simulation-soundness of NIZK and correctness of Π' and E. $\square$

Lemma 5.11. $|\mathfrak{p}_3 - \mathfrak{p}_4| \leq \mathtt{negl}(\lambda)$.

Proof. It is easy to see that output of Game 3 is exactly the output of experiment $\mathtt{Exp}_{\Pi',\mathcal{B},n}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{real}}$ (since $\mathcal{A}$'s view in Game 3 is identical to its view in the experiment $\mathtt{Exp}_{\Pi',\mathcal{B},n}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{real}}$ when invoked by $\mathcal{B}$), thus we have

$$\mathfrak{p}_3 = \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi',\mathcal{B},n}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{real}}(\lambda)) = 1]$$

Also, we can view Game 4 as the ideal experiment $\mathtt{Exp}_{\Pi',\mathcal{S}',n}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{ideal}}$, so we have

$$\mathfrak{p}_4 = \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi',\mathcal{S}',n}^{\mathrm{RSO_k}-\mathtt{CPA}-\mathtt{ideal}}(\lambda)) = 1]$$

Therefore, lemma 5.11 comes from the SIM-RSO$_k$-CPA security of Π' directly. $\square$

Combining Lemma 5.8 to Lemma 5.11, we have $\mathfrak{p}_0 - \mathfrak{p}_4$ negligible and this completes the proof. $\square$

6 Conclusion

In this work, we initiate the study of receiver selective opening security for PKE schemes in the multi-challenge setting. Several interesting open questions remain.

First, our impossibility results only work in either the non-programmable random oracle model or the auxiliary input model. It is interesting to see if we can achieve the impossibility results in the standard model without auxiliary input. Another interesting direction is to explore the relation between PKE scheme with RSO$_k$ security and some related notions, e.g., (receiver) non-committing encryption, hash proof system, etc. Besides, one may note that in our constructions of RSO$_k$ secure PKE schemes, the ciphertexts sizes grow linearly with k. It will be an interesting future work to construct a RSO$_k$ secure PKE scheme with constant-size ciphertexts. Finally, in this work, we mainly focus on the feasibility of achieving RSO$_k$ secure PKE schemes and it will also be interesting to construct practical PKE schemes with RSO$_k$ security.

Acknowledgement. We appreciate the anonymous reviewers for their valuable comments. Part of this work was supported by the National Natural Science Foundation of China (Grant No. 61922036, 61702125, 61802078, 61972332, U1636205, 61632020), the Research Grant Council of Hong Kong (Grant No. 25206317), and the Major Innovation Project of Science and Technology of Shandong Province (Grant No. 2018CXGC0702).

References

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

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112. ACM, 1988.

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

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35. Springer, 2009.

[BL17] Xavier Boyen and Qinyi Li. All-but-many lossy trapdoor functions from lattices and applications. In *CRYPTO*, pages 298–331. Springer, 2017.

[BO13] Mihir Bellare and Adam ONeill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *CANS*, pages 218–234. Springer, 2013.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273. Springer, 2011.

[CDSMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *ASIACRYPT*, pages 287–302. Springer, 2009.

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

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

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25. Springer, 1998.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450. Springer, 2000.

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. In *FOCS*, pages 523–534. IEEE, 1999.

[FHKW10] Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In *EUROCRYPT*, pages 381–402. Springer, 2010.

[HJKS15] Felix Heuer, Tibor Jager, Eike Kiltz, and Sven Schäge. On the selective opening security of practical public-key encryption schemes. In *PKC*, pages 27–51. Springer, 2015.

[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 *SCN*, 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. *DCC*, 87(6):1345–1371, 2019.

[HLOV11] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *ASIACRYPT*, pages 70–88. Springer, 2011.

[HLQ13] Zhengan Huang, Shengli Liu, and Baodong Qin. Sender-equivocable encryption schemes secure against chosen-ciphertext attacks revisited. In *PKC*, pages 369–385. Springer, 2013.

[Hof12] Dennis Hofheinz. All-but-many lossy trapdoor functions. In *EUROCRYPT*, pages 209–227. Springer, 2012.

[HP16] Felix Heuer and Bertram Poettering. Selective opening security from simulatable data encapsulation. In *ASIACRYPT*, pages 248–277. Springer, 2016.

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

[HR14] Dennis Hofheinz and Andy Rupp. Standard versus selective opening security: separation and equivalence results. In *TCC*, pages 591–615. Springer, 2014.

[HRW16] Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In *TCC*, pages 121–145. Springer, 2016.

[JLL16] Dingding Jia, Xianhui Lu, and Bao Li. Receiver selective opening security from indistinguishability obfuscation. In *INDOCRYPT*, 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*, pages 417–431. Springer, 2017.

[KT18] Fuyuki Kitagawa and Keisuke Tanaka. Key dependent message security and receiver selective opening security for identity-based encryption. In *PKC*, pages 32–61. Springer, 2018.

[LLHG18] Lin Lyu, Shengli Liu, Shuai Han, and Dawu Gu. Tightly SIM-SO-CCA secure public key encryption from standard assumptions. In *PKC*, pages 62–92. Springer, 2018.

[LP15] Shengli Liu and Kenneth G. Paterson. Simulation-based selective opening CCA security for PKE from key encapsulation mechanisms. In *PKC*, pages 3–26. Springer, 2015.

[LSSS17] Benoît Libert, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In *CRYPTO*, pages 332–364. Springer, 2017.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, pages 111–126. Springer, 2002.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. ACM, 1990.

[ORV14] Rafail Ostrovsky, Vanishree Rao, and Ivan Visconti. On selective-opening attacks against encryption schemes. In *SCN*, pages 578–597. Springer, 2014.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553. IEEE, 1999.

A Deterred Proofs

A.1 Proof of Theorem 4.1

Proof. We provide the proof of Theorem 4.1 in this section.

Correctness of Π comes from correctness of E and completeness of NIZK directly. Next, we focus on the SIM-RSO$_k$-CCA security of Π.

First, for any polynomial n, any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, and any distinguisher $\mathcal{D}$, we design the simulator $\mathcal{S}$ for $\mathcal{A}$, which works as in Figure 6.

Next, we prove that output of the simulator $\mathcal{S}$ is indistinguishable from the output of the adversary in a real game. We argue this via defining the following games:

- **Game 0.** This is the real experiment $\mathtt{Exp}_{\Pi,\mathcal{A},n}^{\mathrm{RSO_k}-\mathtt{CCA-real}}$. In particular, the challenger interacts with the adversary as follows:
  1. On input a security parameter, the challenger first computes $\mathsf{pp} \leftarrow \mathsf{E}.\mathtt{Setup}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{NIZK}.\mathtt{Gen}(1^\lambda)$, and sets $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs})$.
  2. Then, for $i \in [n], \imath \in [k], \jmath \in \{0,1\}$, it computes $(pk_{i,(\imath,\jmath)}, sk_{i,(\imath,\jmath)}) \leftarrow \mathsf{E}.\mathtt{Gen}(\mathsf{pp})$. Also, it samples $s_{i,1}, \ldots, s_{i,k} \xleftarrow{\$} \{0,1\}$ for $i \in [n]$ and sets $PK_i = (pk_{i,(\imath,\jmath)})_{\imath\in[k],\jmath\in\{0,1\}}$.
  3. Next, the challenger sends $\mathsf{PP}, (PK_i)_{i\in[n]}$ to $\mathcal{A}$ and answers $\mathcal{A}$'s decryption oracle queries as follows:
     (a) On input a pair (i, C), where $C = ((c_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}, \pi)$, the challenger first checks if π is valid and returns an error symbol $\perp$ if π is not valid.
     (b) Otherwise, it computes $p_\imath = \mathsf{E}.\mathtt{Dec}(\mathsf{pp}, pk_{i,(\imath,s_{i,\imath})}, sk_{i,(\imath,s_{i,\imath})}, c_{\imath,s_{i,\imath}})$ for $\imath \in [k]$ and computes $m = p_1 \oplus \ldots \oplus p_k$.
     (c) Finally, it returns m to $\mathcal{A}$.

```
$\mathcal{S}_1(1^\lambda)$:
  $\mathsf{pp} \leftarrow \mathsf{E.Setup}(1^\lambda)$
  $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.S}_1(1^\lambda)$
  $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs})$
  $\mathcal{C} = \emptyset$
  For $i \in [n]$:
    For $\imath \in [k], \jmath \in \{0,1\}$:
      $(pk_{i,(\imath,\jmath)}, sk_{i,(\imath,\jmath)}) \leftarrow \mathsf{E.Gen}(\mathsf{pp})$
    $PK_i = (pk_{i,(\imath,\jmath)})_{\imath \in [k], \jmath \in \{0,1\}}$
  $(\mathcal{M}, \mathfrak{s}_1') \leftarrow \mathcal{A}_1^{\mathsf{DecS}}(\mathsf{PP}, (PK_i)_{i \in [n]})$
  $\mathfrak{s} = (\mathsf{td}, (sk_{i,(\imath,\jmath)})_{i \in [n], \imath \in [k], \jmath \in \{0,1\}})$
  $\mathfrak{s}_1 = (\mathfrak{s}_1', \mathsf{PP}, (PK_i)_{i \in [n]}, \mathfrak{s})$
  Output $(\mathcal{M}, \mathfrak{s}_1)$

$\mathsf{DecS}(i, C)$:
  If $(i, C) \in \mathcal{C}$: Return $\perp$
  Parse $C = ((c_{\imath,\jmath})_{\imath \in [k], \jmath \in \{0,1\}}, \pi)$
  $x = (\mathsf{pp}, (pk_{i,(\imath,\jmath)}, c_{\imath,\jmath})_{\imath \in [k], \jmath \in \{0,1\}})$
  If $\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) = 0$:
    Return $\perp$
  For $\imath \in [k]$:
    $p_\imath = \mathsf{E.Dec}(\mathsf{pp}, pk_{i,(\imath,0)}, sk_{i,(\imath,0)}, c_{\imath,0})$
  $m = p_1 \oplus \ldots \oplus p_k$
  Return $m$

$\mathcal{S}_2(\mathfrak{s}_1)$:
  For $i \in [n], j \in [k]$:
    For $\imath \in [k] \wedge \imath \neq j$:
      $p_{i,j,\imath} \xleftarrow{\$} \{0,1\}$
      $c_{(i,j),(\imath,0)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(\imath,0)}, p_{i,j,\imath})$
      $c_{(i,j),(\imath,1)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(\imath,1)}, p_{i,j,\imath})$
    $p_{i,j,j} \xleftarrow{\$} \{0,1\}$
    $c_{(i,j),(j,0)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(j,0)}, p_{i,j,j})$
    $c_{(i,j),(j,1)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(j,1)}, 1 - p_{i,j,k})$
    $x_{i,j} = (\mathsf{pp}, (pk_{i,(\imath,\jmath)}, c_{(i,j),(\imath,\jmath)})_{\imath \in [k], \jmath \in \{0,1\}})$
    $\pi_{i,j} \leftarrow \mathsf{NIZK.S}_2(\mathsf{crs}, \mathsf{td}, x_{i,j})$
    $C_{i,j} = ((c_{(i,j),(\imath,\jmath)})_{\imath \in [k], \jmath \in \{0,1\}}, \pi_{i,j})$
  $\mathcal{C} = \{(i, C_{i,j}) \mid i \in [n], j \in [k]\}$
  $(\mathcal{I}, \mathfrak{s}_2') \leftarrow \mathcal{A}_2^{\mathsf{DecS}}((C_{i,j})_{i \in [n], j \in [k]}, \mathfrak{s}_1')$
  $\mathfrak{s}_2 = (\mathfrak{s}_1, \mathfrak{s}_2', (p_{(i,j),(\imath)})_{i \in [n], j \in [k], \imath \in [k]})$
  Output $(\mathcal{I}, \mathfrak{s}_2)$

$\mathcal{S}_3((m_{i,j})_{i \in \mathcal{I}, j \in [k]}, \mathfrak{s}_2)$:
  For $i \in \mathcal{I}$:
    For $j \in [k]$:
      $s_{i,j} = m_{i,j} \oplus p_{i,j,1} \oplus p_{i,j,2} \ldots \oplus p_{i,j,k}$
    $SK_i = (s_{i,\imath}, sk_{i,(\imath, s_{i,\imath})})_{\imath \in [k]}$
  $out \leftarrow \mathcal{A}_3^{\mathsf{DecS}}((SK_i, m_{i,j})_{i \in \mathcal{I}, j \in [k]}, \mathfrak{s}_2')$
  Output $out$
```

Fig. 6 The simulator $\mathcal{S}$ for $\mathcal{A}$ in proving SIM-RSO$_k$-CCA security of Π.

4. The adversary will send a distribution $\mathcal{M}$ to the challenger after querying the decryption oracle a few times. Then, the challenger samples a matrix of messages $\boldsymbol{M} \coloneqq (m_{i,j})_{i \in [n], j \in [k]} \leftarrow \mathcal{M}$ and for each $(i, j) \in [n] \times [k]$, it generates a challenge ciphertext for $m_{i,j}$ as follows:
   (a) Samples $p_{i,j,\imath} \leftarrow \{0,1\}$ for each $\imath \in [k]$ and $\imath \neq j$.
   (b) Computes $p_{i,j,j} = (\bigoplus_{\imath \in [k] \wedge \imath \neq j} p_{i,j,\imath}) \oplus m_{i,j}$.
   (c) For $\imath \in [k], \jmath \in \{0,1\}$, samples $r_{(i,j),(\imath,\jmath)}$ randomly from the randomness space of E, and computes $c_{(i,j),(\imath,\jmath)} = \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(\imath,\jmath)}, p_{i,j,\imath}; r_{(i,j),(\imath,\jmath)})$.
   (d) Computes $\pi_{i,j} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pp}, (pk_{i,(\imath,\jmath)}, c_{(i,j),(\imath,\jmath)})_{\imath \in [k], \jmath \in \{0,1\}}), ((p_{i,j,\imath}, r_{(i,j),(\imath,\jmath)})_{\imath \in [k], \jmath \in \{0,1\}}))$.
   (e) Sets $C_{i,j} = ((c_{(i,j),(\imath,\jmath)})_{\imath \in [k], \jmath \in \{0,1\}}, \pi_{i,j})$.
5. Next, the challenger sends all challenge ciphertexts to $\mathcal{A}$ and answers $\mathcal{A}$'s decryption oracle queries as follows:
   (a) On input a pair (i, C), the challenger first checks if $C = C_{i,j}$ for some $j \in [k]$. It returns $\perp$ if this is the case.

   (b) Otherwise, the challenger parses $C = ((c_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}, \pi)$ and checks if π is valid. It returns an error symbol $\perp$ if π is not valid.
   (c) Otherwise, it computes $p_\imath = \mathsf{E.Dec}(\mathsf{pp}, pk_{i,(\imath,s_{i,\imath})}, sk_{i,(\imath,s_{i,\imath})}, c_{\imath,s_{i,\imath}})$ for $\imath \in [k]$ and computes $m = p_1 \oplus \ldots \oplus p_k$.
   (d) Finally, it returns m to $\mathcal{A}$.
6. The adversary will send a set $\mathcal{I} \subseteq [n]$ to the challenger after querying the decryption oracle a few times. Then, the challenger sets $SK_i = (s_{i,\imath}, sk_{i,(\imath,s_{i,\imath})})_{\imath\in[k]}$ for $i \in \mathcal{I}$ and sends $(SK_i, m_{i,j})_{i\in\mathcal{I},j\in[k]}$ to $\mathcal{A}$. The challenger will answer $\mathcal{A}$'s decryption queries exactly as in step 5.
7. Finally, on receiving $\mathcal{A}$'s output out, the challenger outputs $(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out)$.

- **Game 1.** This is identical to Game 0 except that when generating the common reference string and proofs, the challenger uses the simulator of NIZK instead of generating them honestly. More precisely, in the first step, the challenger computes $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.S}_1(1^\lambda)$ and in step 4, the challenger computes $\pi_{i,j} \leftarrow \mathsf{NIZK.S}_2(\mathsf{crs}, \mathsf{td}, (\mathsf{pp}, (pk_{i,(\imath,\jmath)}, c_{(i,j),(\imath,\jmath)})_{\imath\in[k],\jmath\in\{0,1\}}))$.
- **Game 2.** This is identical to Game 1 except that the challenger changes the way to generate challenge ciphertexts. More precisely, for each $i \in [n]$, $j \in [k]$, the challenger computes $c_{(i,j),(j,1-s_{i,j})} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(j,1-s_{i,j})}, 1-p_{i,j,j})$. That is to say, let $p'_{i,j,j} = m_{i,j} \oplus (\bigoplus_{\imath\in[k]\wedge\imath\neq j} p_{i,j,\imath}) \oplus s_{i,j}$, then the challenger computes $c_{(i,j),(j,0)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(j,0)}, p'_{i,j,j})$ and $c_{(i,j),(j,1)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(j,1)}, 1 - p'_{i,j,k})$ for each $i \in [n], j \in [k]$.
- **Game 3.** This is identical to Game 2 except that the challenger changes the way to answer decryption queries. More precisely, it will use $sk_{i,(\imath,0)}$ instead of $sk_{i,(\imath,s_{i,\imath})}$ when answering decryption queries.
- **Game 4.** This is identical to Game 3 except that the challenger changes the way to generate challenge ciphertexts and answer the opening query. More precisely:
  1. When generating the challenge ciphertexts (in step 4), it samples $p'_{i,j,j} \xleftarrow{\$} \{0,1\}$ and computes $c_{(i,j),(j,0)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(j,0)}, p'_{i,j,j})$ and $c_{(i,j),(j,1)} \leftarrow \mathsf{E.Enc}(\mathsf{pp}, pk_{i,(j,1)}, 1 - p'_{i,j,k})$ for each $i \in [n], j \in [k]$.
  2. The challenger does not sample $s_{i,j}$ in the first step and when answering the opening query (in step 6), it sets $s_{i,j} = m_{i,j} \oplus (\bigoplus_{\imath\in[k]\wedge\imath\neq j} p_{i,j,\imath}) \oplus p'_{i,j,j}$ for $i \in \mathcal{I}, j \in [k]$.

Let $\mathfrak{p}_\alpha$ be the probability that $\mathcal{D}$ outputs 1 when taking the output of Game α as input, then we have $\mathfrak{p}_0 = \Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CCA}-\mathtt{real}}_{\Pi,\mathcal{A},n}(\lambda)) = 1]$. Also, it is easy to see that output of Game 4 is exactly the output of the ideal experiment, so, we have $\mathfrak{p}_4 = \Pr[\mathcal{D}(\mathtt{Exp}^{\mathrm{RSO_k}-\mathtt{CCA}-\mathtt{ideal}}_{\Pi,\mathcal{S},n}(\lambda)) = 1]$. Next, we prove that $\mathfrak{p}_0 - \mathfrak{p}_4$ is negligible via showing that $\mathfrak{p}_\alpha - \mathfrak{p}_{\alpha+1}$ is negligible for all $\alpha \in [0, 3]$.

Lemma A.1. $|\mathfrak{p}_0 - \mathfrak{p}_1| \leq \mathsf{negl}(\lambda)$.

Proof. This comes from the unbounded zero-knowledge property of NIZK directly. □

Lemma A.2. $|\mathfrak{p}_1 - \mathfrak{p}_2| \leq \mathsf{negl}(\lambda)$.

Proof. In Game 1 and Game 2, the challenger will use $sk_{i,(\imath,s_{i,\imath})}$ to answer decryption queries and the key opening query. So, for all $i \in [n], \imath \in [k]$, $sk_{i,(\imath,1-s_{i,\imath})}$ is hidden from the view of the adversary. Therefore, from the CPA security of E, we have indistinguishability between $\mathsf{E}.\mathtt{Enc}(\mathsf{pp}, pk_{i,(j,1-s_{i,j})}, p_{i,j,j})$ and $\mathsf{E}.\mathtt{Enc}(\mathsf{pp}, pk_{i,(j,1-s_{i,j})}, 1-p_{i,j,j})$, and as a result, indistinguishability between Game 1 and Game 2 follows. □

Lemma A.3. $|\mathfrak{p}_2 - \mathfrak{p}_3| \leq \mathsf{negl}(\lambda)$.

Proof. This comes from the unbounded simulation-soundness of NIZK and the correctness of E straightforwardly. □

Lemma A.4. $|\mathfrak{p}_3 - \mathfrak{p}_4| = 0$.

Proof. The only difference between Game 3 and Game 4 is the order for generating some variables and it is easy to see that the view of $\mathcal{A}$ is identical in both games. □

Combining Lemma A.1 to Lemma A.4, we have $\mathfrak{p}_0 - \mathfrak{p}_4$ negligible and this completes the proof.

□

A.2 Proof of Theorem 4.2

Proof. In this section, we provide the proof of Theorem 4.2. The proof is similar to the proof of Theorem 3.1.

We prove that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, a PPT distinguisher $\mathcal{D}$, a polynomial n, and a non-negligible ϵ that for any PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$,

$$|\Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{A},n}^{\mathrm{RSO_{k+1}}-\mathtt{CPA}-\mathtt{real}}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{S},n}^{\mathrm{RSO_{k+1}}-\mathtt{CPA}-\mathtt{ideal}}(\lambda)) = 1]| \geq \epsilon$$

Before presenting our main proof, we first show that for any public key PK of Π, there are at most 2^k different valid secret keys for PK. To show this, we define an algorithm $\mathtt{Ver}$ for Π as follows:

- **Ver.** On input a public parameter $\mathsf{PP} = (\mathsf{pp}, \mathsf{crs})$, a public key $PK = (pk_{\imath,\jmath})_{\imath\in[k],\jmath\in\{0,1\}}$ and a secret key $SK = (s_\imath, sk_\imath)_{\imath\in[k]}$, the verification algorithm computes $b_\imath = \mathsf{E}.\mathtt{Ver}(\mathsf{pp}, pk_{\imath,s_\imath}, sk_\imath)$ for each $\imath \in [k]$ and outputs $\bigwedge_{\imath\in[k]} b_\imath$.

Lemma A.5. *Let* $\mathsf{PP} \leftarrow \mathtt{Setup}(1^\lambda)$ *and* $(PK, SK) \leftarrow \mathtt{Gen}(\mathsf{PP})$*, then* $\Pr[\mathtt{Ver}(\mathsf{PP}, PK, SK) = 1] = 1$.

Proof. This comes from the verification correctness of E directly. □

Lemma A.6. *For any* PP *and for any* PK, $|\{SK \mid \mathtt{Ver}(\mathsf{PP}, PK, SK) = 1\}| \leq 2^k$.

Proof. First, for any PP, PK and for any fixed $\boldsymbol{s^*} = (s_1^*, \ldots, s_k^*)$, due to the key uniqueness of E, there is at most one secret key SK satisfying 1) $SK = (s_\imath^*, sk_\imath)_{\imath \in [k]}$ and 2) $\mathtt{Ver}(\mathsf{PP}, PK, SK) = 1$ simultaneously. Besides, as $\boldsymbol{s}^*$ is a k-bit string, there are only 2^k different $\boldsymbol{s}^*$. Therefore, the number of SK that satisfies $\mathtt{Ver}(\mathsf{PP}, PK, SK) = 1$ is at most 2^k. $\square$

Now, we are ready to show our main proof. Let $H : \{0,1\}^* \to \{0,1\}^h$ be a hash function, which is modeled as a non-programmable random oracle. Let $\mathcal{PP}$, $\mathcal{PK}$ and $\mathcal{C}$ be the public parameters set, the public key space and the ciphertext space of Π respectively. Also, let $a = \lceil \log |\mathcal{PP}| \rceil$, $b = \lceil \log |\mathcal{PK}| \rceil$, $c = \lceil \log |\mathcal{C}| \rceil$ and let $\kappa = a + b + c(k+1) + 2$.

First, we define $\mathcal{A}$ and $\mathcal{D}$ as in Figure 7. Also, we define $n = h + 1$ and $\epsilon = 1/(4\kappa)$.

Next, fixing any PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ [10], let

$$\delta = \Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{S},n}^{\mathrm{RSO}_{k+1}-\mathtt{CPA-ideal}}(\lambda)) = 1]$$

Note that $\Pr[\mathcal{D}(\mathtt{Exp}_{\Pi,\mathcal{A},n}^{\mathrm{RSO}_{k+1}-\mathtt{CPA-real}}(\lambda)) = 1]$ is negligible due to the correctness and the verification correctness of Π, therefore, it is sufficient to show that δ is notably larger than ϵ. Concretely, we will argue that $\delta \geq 1/(2\kappa)$ in the remaining part of the proof.

To lower bound δ, we consider an auxiliary experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$ defined in Figure 8 and analyze the distribution of its output.

Lemma A.7. $\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 0] \leq 1/4$.

Proof. Assume the experiment outputs 0. First, we have $\mathcal{M} = \mathcal{U}_{n(k+1)}$, thus, for each $\iota \in [\kappa], i \in [n], j \in [k+1]$, $m_{i,j}^\iota$ is sampled uniformly at random from $\{0,1\}$. Also, we know that $n \in \mathcal{I}$ and for $\iota \in [\kappa]$ and $j \in [k+1]$, we set $PK^\iota = PK_n^\iota$, $C_j^\iota = C_{n,j}^\iota$, $SK^\iota = SK_n^\iota$, and $m_j^\iota = m_{n,j}^\iota$. Moreover, we have $(\mathsf{PP}^{\iota-1}, (PK^{\iota-1}, C_j^{\iota-1})_{j\in[k+1]}) = (\mathsf{PP}^\iota, (PK^\iota, C_j^\iota)_{j\in[k+1]})$ for all $\iota \in [n]$ and thus we can write PP^ι as PP, PK^ι as PK and C_j^ι as C_j. Besides, we have $\mathtt{Ver}(\mathsf{PP}, PK, SK^\iota) = 1$ and from Lemma A.6, we can conclude that for any fixed PP and PK, there exist at most 2^k different SK^ι that satisfies this condition. Finally, for all $\iota \in [\kappa]$ and $j \in [k+1]$, $m_j^\iota = \mathtt{Dec}(\mathsf{PP}, PK, SK^\iota, C_j)$.

Next, we analyze the probability that all above requirements are satisfied.

First, fix any tuple $(\mathsf{PP}, PK, \boldsymbol{C} = (C_1, \ldots, C_{k+1}), \boldsymbol{SK} = (SK^1, \ldots, SK^\kappa))$, which is *not* necessary the output of the simulator, then we have

$$\Pr[\forall \iota \in [\kappa], j \in [k+1], m_j^\iota = \mathtt{Dec}(\mathsf{PP}, PK, SK^\iota, C_j)] = \frac{1}{2^{\kappa(k+1)}}$$

where the probability is taken over the random choice of each m_j^ι.

[10] Here, w.l.o.g., we assume that $\mathcal{S}_2$ and $\mathcal{S}_3$ are deterministic. This will not restrict the power of $\mathcal{S}$ since we can feed coins for $\mathcal{S}_2$ and $\mathcal{S}_3$ to $\mathcal{S}_1$ and require $\mathcal{S}_1$ (and $\mathcal{S}_2$) to put coins for $\mathcal{S}_2$ and $\mathcal{S}_3$ (resp. $\mathcal{S}_3$) in its outputted state s_1 (resp. s_2).

```
$\mathcal{A}_1(\mathsf{PP}, (PK_i)_{i\in[n]})$ :
  $\mathcal{M} = \mathcal{U}_{n(k+1)}$
  $s_1 = (\mathsf{PP}, (PK_i)_{i\in[n]})$
  Output $(\mathcal{M}, s_1)$
$\mathcal{A}_2((C_{i,j})_{i\in[n],j\in[k+1]}, s_1)$ :
  $t = H(\mathsf{PP}, (PK_i, C_{i,j})_{i\in[n],j\in[k+1]})$
  $\mathcal{I} = \{i \mid i \in [h] \wedge t[i] = 1\} \cup \{n\}$
  $s_2 = (s_1, (C_{i,j})_{i\in[n],j\in[k+1]})$
  Output $(\mathcal{I}, s_2)$
$\mathcal{A}_3((SK_i, m_{i,j})_{i\in\mathcal{I},j\in[k+1]}, s_2)$ :
  $out = (s_2, (SK_i)_{i\in\mathcal{I}})$
  Output $out$
```

```
$\mathcal{D}((m_{i,j})_{i\in[n],j\in[k+1]}, \mathcal{M}, \mathcal{I}, out)$ :
  If $\mathcal{M} \neq \mathcal{U}_{n(k+1)}$ : **Output 1**
  $(\mathsf{PP}, (PK_i, C_{i,j})_{i\in[n],j\in[k+1]}, (SK_i)_{i\in\mathcal{I}}) = out$
  $t = H(\mathsf{PP}, (PK_i, C_{i,j})_{i\in[n],j\in[k+1]})$
  $\mathcal{I}' = \{i \mid i \in [h] \wedge t[i] = 1\} \cup \{n\}$
  If $\mathcal{I} \neq \mathcal{I}'$ : **Output 1**
  $(m'_{i,j} = \mathtt{Dec}(\mathsf{PP}, PK_i, SK_i, C_{i,j}))_{i\in\mathcal{I},j\in[k+1]}$
  For $i \in \mathcal{I}, j \in [k+1]$ :
    If $m_{i,j} \neq m'_{i,j}$ : **Output 1**
  For $i \in [n]$ :
    If $\mathtt{Ver}(\mathsf{PP}, PK_i, SK_i) = 0$ : **Output 1**
  **Output 0**
```

Fig. 7 The adversary $\mathcal{A}$ and $\mathcal{D}$ in attacking SIM-RSO$_{k+1}$-CPA security of Π. Here, we abuse the notation of $\mathcal{U}_{n(k+1)}$ to denote the description of an algorithm that outputs uniform $n(k+1)$-bit string and assume that this description is hardwired in both $\mathcal{A}$ and $\mathcal{D}$.

```
$\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$ :
  $(\mathcal{M}, s_1) \leftarrow \mathcal{S}_1(1^\lambda)$ ; $(\mathcal{I}, s_2) = \mathcal{S}_2(s_1)$
  For $\iota \in [1, \kappa]$ :
    $(m^\iota_{i,j})_{i\in[n],j\in[k+1]} \leftarrow \mathcal{M}$ ; $out^\iota = \mathcal{S}_3((m^\iota_{i,j})_{i\in\mathcal{I},j\in[k+1]}, s_2)$
    If $\mathcal{D}((m^\iota_{i,j})_{i\in[n],j\in[k+1]}, \mathcal{M}, \mathcal{I}, out^\iota) = 1$ : **Output 1**
  For $\iota \in [2, \kappa]$ :
    Parse $out^\iota = (\mathsf{PP}^\iota, (PK^\iota_i, C^\iota_{i,j})_{i\in[n],j\in[k+1]}, (SK^\iota_i)_{i\in\mathcal{I}})$
    If $(\mathsf{PP}^{\iota-1}, (PK^{\iota-1}_i, C^{\iota-1}_{i,j})_{i\in[n],j\in[k+1]}) \neq (\mathsf{PP}^\iota, (PK^\iota_i, C^\iota_{i,j})_{i\in[n],j\in[k+1]})$ :
      **Output 2**
  **Output 0**
```

Fig. 8 The auxiliary experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$.

Next, we only fix PP, PK and $\boldsymbol{C} = (C_1, \ldots, C_{k+1})$ and allow the simulator to choose $\boldsymbol{SK}$. As for fixed PP and PK, each SK^ι can only be chosen from a set of at most 2^k elements, we have

$$\begin{aligned}&\Pr[\exists \boldsymbol{SK} : (\forall \iota \in [\kappa], \mathtt{Ver}(\mathsf{PP}, PK, SK^\iota) = 1) \wedge \\ &\qquad (\forall \iota \in [\kappa], j \in [k+1], m^\iota_j = \mathtt{Dec}(\mathsf{PP}, PK, SK^\iota, C_j))] \leq \frac{2^{\kappa\cdot k}}{2^{\kappa(k+1)}} = \frac{1}{2^\kappa}\end{aligned}$$

Finally, as the total possible ways to choose PP, PK and $\boldsymbol{C} = (C_1, \ldots, C_{k+1})$ does not exceed $2^{a+b+c(k+1)} = 2^{\kappa-2}$, we have

$$\Pr[\exists \mathsf{PP}, PK, \boldsymbol{C}, \boldsymbol{SK} : (\forall \iota \in [\kappa], \mathtt{Ver}(\mathsf{PP}, PK, SK^{\iota}) = 1) \wedge$$
$$(\forall \iota \in [\kappa], j \in [k+1], m_j^{\iota} = \mathtt{Dec}(\mathsf{PP}, PK, SK^{\iota}, C_j))] \leq \frac{2^{\kappa-2}}{2^{\kappa}} = \frac{1}{4}$$

Therefore, the probability that the auxiliary experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$ outputs 0 does not exceed $1/4$. □

Lemma A.8. $\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 1] \leq \kappa \cdot \delta$.

Proof. First, note that randomness of the experiment $\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa}$ comes from two parts, namely, randomness of the simulator $\mathcal{S}$ (denoted as ρ here) and randomness used in sampling $m_{i,j}^{\iota}$. Let $\mathcal{P}$ be the randomness space of $\mathcal{S}$. Let

$$f(\rho) = \Pr\left[\begin{array}{l} (\mathcal{M}, s_1) \leftarrow \mathcal{S}_1(1^{\lambda}; \rho); \\ (\mathcal{I}, s_2) = \mathcal{S}_2(s_1); \\ \boldsymbol{M} := (m_{i,j})_{i\in[n],j\in[k+1]} \leftarrow \mathcal{M}; \\ out = \mathcal{S}_3((m_{i,j})_{i\in\mathcal{I},j\in[k+1]}, s_2); \end{array} : \quad \mathcal{D}(\boldsymbol{M}, \mathcal{M}, \mathcal{I}, out) = 1\right]$$

where the probability is taken over the random choice of each $\boldsymbol{M}$. Then, we have

$$\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 1] = \sum_{\rho \xleftarrow{\$} \mathcal{P}} \frac{1}{\mathcal{P}} \cdot (1 - (1 - f(\rho))^{\kappa}) \leq \sum_{\rho \xleftarrow{\$} \mathcal{P}} \frac{1}{\mathcal{P}} \cdot \kappa \cdot f(\rho) = \kappa \cdot \delta$$

where the second inequality comes from the Bernoulli's inequality. □

Lemma A.9. $\Pr[\mathtt{Exp}_{\Pi,\mathcal{S},\mathcal{D},n,k,\kappa} = 2] \leq 1/4$.

Proof. This comes from the collision resistant property of the non-programmable random oracle, which is a random function whose output is not controlled by the simulator.

Assuming that H has been queried (either by the adversary, the distinguisher or the simulator) Q times, where Q is a polynomial. Then the probability that there exists two distinct queries x_1, x_2 s.t. $H(x_1) = H(x_2)$ does not exceed $\frac{Q^2}{2^h}$, which is negligible.

However, if the experiment outputs 2 with a non-negligible probability (e.g., $1/4$), then, via running the experiment, one can find $\iota \in [\kappa]$ that

1) $(\mathsf{PP}^{\iota-1}, (PK_i^{\iota-1}, C_{i,j}^{\iota-1})_{i\in[n],j\in[k+1]}) \neq (\mathsf{PP}^{\iota}, (PK_i^{\iota}, C_{i,j}^{\iota})_{i\in[n],j\in[k+1]})$

2) $H(\mathsf{PP}^{\iota-1}, (PK_i^{\iota-1}, C_{i,j}^{\iota-1})_{i\in[n],j\in[k+1]}) = H(\mathsf{PP}^{\iota}, (PK_i^{\iota}, C_{i,j}^{\iota})_{i\in[n],j\in[k+1]}) = (t_1, \ldots, t_h)$, where $t_i = 1$ iff $i \in \mathcal{I}$ (otherwise, the experiment will output 1)

with a non-negligible probability, which makes a contradiction. □

Finally, combining Lemma A.7 to Lemma A.9, we have

$$1 \leq 1/4 + \kappa \cdot \delta + 1/4$$

which implies $\delta \geq \frac{1}{2\kappa}$ and this completes the proof. □