Public Key Encryption with Secure Key Leasing

Shweta Agrawal*, Fuyuki Kitagawa†, Ryo Nishimaki†,
Shota Yamada‡, Takashi Yamakawa†

*IIT Madras, Chennai, India
shweta.a@cse.iitm.ac.in
†NTT Social Informatics Laboratories, Tokyo, Japan
{fuyuki.kitagawa.yh,ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp
‡National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
yamada-shota@aist.go.jp

April 6, 2023

Abstract

We introduce the notion of public key encryption with secure key leasing (PKE-SKL). Our notion supports the leasing of decryption keys so that a leased key achieves the decryption functionality but comes with the guarantee that if the quantum decryption key returned by a user passes a validity test, then the user has lost the ability to decrypt. Our notion is similar in spirit to the notion of secure software leasing (SSL) introduced by Ananth and La Placa (Eurocrypt 2021) but captures significantly more general adversarial strategies[1]. Our results can be summarized as follows:

1. *Definitions:* We introduce the definition of PKE with secure key leasing and formalize a security notion that we call indistinguishability against key leasing attacks (IND-KLA security). We also define a one-wayness notion for PKE-SKL that we call OW-KLA security and show that an OW-KLA secure PKE-SKL scheme can be lifted to an IND-KLA secure one by using the (quantum) Goldreich-Levin lemma.
2. *Constructing IND-KLA PKE with Secure Key Leasing:* We provide a construction of OW-KLA secure PKE-SKL (which implies IND-KLA secure PKE-SKL as discussed above) by leveraging a PKE scheme that satisfies a new security notion that we call *consistent or inconsistent security against key leasing attacks (CoIC-KLA security)*. We then construct a CoIC-KLA secure PKE scheme using 1-key Ciphertext-Policy Functional Encryption (CPFE) that in turn can be based on any IND-CPA secure PKE scheme.
3. *Identity Based Encryption, Attribute Based Encryption and Functional Encryption with Secure Key Leasing:* We provide definitions of secure key leasing in the context of advanced encryption schemes such as identity based encryption (IBE), attribute-based encryption (ABE) and functional encryption (FE). Then we provide constructions by combining the above PKE-SKL with standard IBE, ABE and FE schemes.

   Notably, our definitions allow the adversary to request *distinguishing* keys in the security game, namely, keys that distinguish the challenge bit by simply decrypting the challenge ciphertext, as long as it returns them (and they pass the validity test) before it sees the challenge ciphertext. All our constructions satisfy this stronger definition, albeit with the restriction that only a bounded number of such keys is allowed to the adversary in the IBE and ABE (but not FE) security games.

Prior to our work, the notion of single decryptor encryption (SDE) has been studied in the context of PKE (Georgiou and Zhandry, Eprint 2020) and FE (Kitigawa and Nishimaki, Asiacrypt 2022) but all their constructions rely on strong assumptions including indistinguishability obfuscation. In contrast, our constructions do not require any additional assumptions, showing that PKE/IBE/ABE/FE can be upgraded to support secure key leasing for free.

[1] In more detail, our adversary is not restricted to use an honest evaluation algorithm to run pirated software.

Contents

1 Introduction

Recent years have seen amazing advances in cryptography by leveraging the power of quantum computation. Several novel primitives such as perfectly secure key agreement [BB20], quantum money [Wie83], quantum copy protection [Aar09], one shot signatures [AGKZ20] and such others, which are not known to exist in the classical world, can be constructed in the quantum setting, significantly advancing cryptographic capabilities.

In this work, we continue to study harnessing quantum powers to protect against software piracy. The quantum no-cloning principle intuitively suggests applicability to anti-piracy, an approach which was first investigated in the seminal work of Aaronson [Aar09], who introduced the notion of quantum copy protection. At a high level, quantum copy protection prevents users from copying software in the sense that it guarantees that when an adversary is given a copy protected circuit for computing some function f, it cannot create two (possibly entangled) quantum states, both of which can compute f. While interesting in its own right for preventing software piracy, quantum copy protection (for some class of circuits) also has the amazing application of public-key quantum money [AC12]. Perhaps unsurprisingly, constructions of quantum copy protection schemes from standard cryptographic assumptions have remained largely elusive. This motivates the study of primitives weaker than quantum copy protection, which nevertheless offer meaningful guarantees for anti-piracy.

Secure software leasing (SSL), introduced by Ananth and La Placa [AL21], is such a primitive, which while being weaker than quantum copy-protection, is nevertheless still meaningful for software anti-piracy. Intuitively, this notion allows to encode software into a version which may be leased or rented out, for some specific term at some given cost. Once the lease expires, the lessee returns the software and the lessor can run an efficient procedure to verify its validity. If the software passes the test, we have the guarantee that the lessee is no longer able to run the software (using the honest evaluation algorithm).

In this work, we explore the possibility of equipping public key encryption (PKE) with a key leasing capability. The benefits of such a capability are indisputable – in the real world, decryption keys of users often need to be revoked, for instance, when a user leaves an organization. In the classical setting, nothing prevents the user from maintaining a copy of her decryption key and misusing its power. Revocation mechanisms have been designed to prevent such attacks, but these are often cumbersome in practice. Typically, such a mechanism entails the revoked key being included in a Certificate Revocation List (CRL) or Certificate Revocation Trees (CRT), or some database which is publicly available, so that other users are warned against its usage. However, the challenges of effective certificate revocation are well acknowledged in public key infrastructure – please see [BDTW01] for a detailed discussion. If the decryption keys of a PKE could be encoded as quantum states and allow for verifiable leasing, this would constitute a natural and well-fitting solution to the challenge of key revocation.

1.1 Prior Work

In this section, we discuss prior work related to public key encryption (PKE) and public key functional encryption (PKFE), where decryption keys are encoded into quantum states to benefit from uncloneability. For a broader discussion on prior work related to quantum copy protection and secure software leasing, we refer the reader to Section 1.4.

Georgiou and Zhandry [GZ20] introduced the notion of single decryptor encryption (SDE), where the decryption keys are unclonable quantum objects. They showed how to use one-shot signatures together with extractable witness encryption with quantum auxiliary information to achieve public key SDE. Subsequently, Coladangelo, Liu, Liu, and Zhandry [CLLZ21] achieved SDE assuming iO and extractable witness encryption or assuming subexponential iO, subexponential OWF, LWE and a strong monogamy property (which was subsequently shown to be true [CV22]). Very recently, Kitagawa and Nishimaki [KN22a] introduced the notion of single-decryptor functional encryption (SDFE), where each functional decryption key is copy protected and provided collusion-resistant single decryptor PKFE for $\mathsf{P/poly}$ from the subexponential hardness of iO and LWE.

It is well-known [ALL+21, AL21] that copy protection is a stronger notion than SSL[2] – intuitively, if an adversary can generate two copies of a program, then it can return one of them while keeping the other for later use. Thus, constructions of single decryptor encryption [GZ20, CLLZ21, KN22a] imply our notion of PKE with secure key leasing

[2]The informed reader may observe that this implication may not always be true due to some subtleties, but we ignore these for the purpose of the overview.

from their respective assumptions, which all include at least the assumption of iO (see Appendix A for the detail). Additionally, in the context of public key FE, the only prior work by Kitagawa and Nishimaki [KN22a] considers the restricted single-key setting where an adversary is given a single decryption key that can be used to detect the challenge bit. In contrast, we consider the more powerful multi-key setting, which makes our definition of FE-SKL incomparable to the SDFE considered by [KN22a]. For the primitives of IBE and ABE, there has been no prior work achieving any notion of key leasing to the best of our knowledge. We also note that Aaronson et al. [ALL+21] studied the notion of "copy-detection", which is a weaker form of copy protection, for any "watermarkable" functionalities based on iO and OWF. In particular, by instantiating the construction with the watermarkable PKE of [GKM+19], they obtain PKE with copy-detection from iO + PKE.

Overall, all previous works that imply PKE-SKL are designed to achieve the stronger goal of copy protection (or the incomparable goal of copy detection) and rely at least on the strong assumption of iO. In this work, our goal is to achieve the weaker goal of PKE-SKL from standard assumptions.

1.2 Our Results

In this work, we initiate the study of public key encryption with secure key leasing. Our results can be summarized as follows:

1. *Definitions:* We introduce the definition of PKE with secure key leasing (PKE-SKL) to formalize the arguably natural requirement that decryption keys of a PKE scheme is encoded into a leased version so that the leased key continues to achieve the decryption functionality but now comes with an additional "returnability" guarantee. In more detail, the security of PKE-SKL requires that if the quantum decryption key returned by a user passes a validity test, then the user has lost the ability to decrypt. To capture this intuition, we formalize a security notion that we call indistinguishability against key leasing attacks (IND-KLA security). We also define a one-wayness notion for PKE-SKL that we call OW-KLA security and show that an OW-KLA secure PKE-SKL scheme can be lifted to an IND-KLA secure one by using the (quantum) Goldreich-Levin lemma.

2. *Constructing IND-KLA PKE with Secure Key Leasing:* We provide a construction of OW-KLA secure PKE-SKL (which imples IND-KLA PKE-SKL as discussed above) by leveraging a PKE scheme that satisfies a new security notion that we call *consistent or inconsistent security against key leasing attacks (CoIC-KLA security)*. We then construct a CoIC-KLA secure PKE scheme using 1-key Ciphertext-Policy Functional Encryption (CPFE) that in turn can be based on any IND-CPA secure PKE scheme.

3. *Identity Based Encryption, Attribute Based Encryption and Functional Encryption with Secure Key Leasing:* We provide definitions of secure key leasing in the context of advanced encryption schemes such as identity based encryption (IBE), attribute-based encryption (ABE) and functional encryption (FE). Then we provide constructions by combining the above PKE-SKL with standard IBE, ABE and FE schemes.

   Notably, our definitions allow the adversary to request *distinguishing* keys in the security game, namely, keys that distinguish the challenge bit by simply decrypting the challenge ciphertext. Recall that this was not permitted in the classical setting to avoid trivializing the security definition. However, in the quantum setting, we consider a stronger definition where the adversary can request such keys so long as it returns them (and they pass the validity test) before it sees the challenge ciphertext. All our constructions satisfy this stronger definition, albeit with the restriction that only a bounded number of such keys be allowed to the adversary in the IBE and ABE (but not FE) security games. We emphasize that this restriction is a result of our techniques and could potentially be removed in future work.

We note that, in general, secure software leasing (SSL) only ensures a notion of security where the adversary is forced to use an honest evaluation algorithm for the software. However, our definition (and hence constructions) of PKE/ABE/FE SKL do not suffer from this limitation. Our constructions do not require any additional assumptions, showing that PKE/IBE/ABE/FE can be upgraded to support secure key leasing for free.

1.3 Technical Overview

We proceed to give a technical overview of this work.

Definition of PKE with secure key leasing. We first introduce the definition of PKE with secure key leasing (PKE-SKL). A PKE-SKL scheme SKL consists of four algorithms $(\mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$, where the first three algorithms form a standard PKE scheme except the following differences on $\mathcal{KG}$.[3]

- $\mathcal{KG}$ outputs a quantum decryption key dk instead of a classical decryption key.
- $\mathcal{KG}$ outputs a (secret) verification key vk, together with a public encryption key and quantum decryption key.

The verification algorithm $\mathcal{Vrfy}$ takes as input a verification key and a quantum decryption key, and outputs $\top$ or $\bot$. In addition to decryption correctness, SKL should satisfy verification correctness that states that $\mathcal{Vrfy}(\mathsf{vk}, \mathit{dk}) = \top$ holds, where $(\mathsf{ek}, \mathit{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$.

The security of PKE-SKL requires that once a user holding a quantum decryption key returns the key correctly, the user can no longer use the key and lose the ability to decrypt. We formalize this as a security notion that we call indistinguishability against key leasing attacks (IND-KLA security). It is defined by using the following security game.

1. First, the challenger generates $(\mathsf{ek}, \mathit{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$ and sends ek and dk to an adversary $\mathcal{A}$.
2. $\mathcal{A}$ sends two challenge plaintexts $(\mathsf{m}_0^*, \mathsf{m}_1^*)$ and a quantum state $\widetilde{\mathit{dk}}$ that is supposed to be a correct decryption key. The challenger checks if $\mathcal{Vrfy}(\mathsf{vk}, \widetilde{\mathit{dk}}) = \top$ holds. If not, $\mathcal{A}$ is regarded as invalid and the game ends here. Otherwise, the game goes to the next step.[4]
3. The challenger generates $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}^*_{\mathsf{coin}})$ and sends it to $\mathcal{A}$, where $\mathsf{coin} \leftarrow \{0,1\}$.
4. $\mathcal{A}$ outputs coin'.

IND-KLA security guarantees that any QPT $\mathcal{A}$ cannot guess coin correctly significantly better than random guessing, conditioned on $\mathcal{A}$ being valid. In more detail, for any QPT adversary $\mathcal{A}$ that passes the verification with a non-negligible probability, we have $\left|\Pr\left[\mathsf{coin}' = \mathsf{coin} \mid \mathcal{Vrfy}(\mathsf{vk}, \widetilde{\mathit{dk}}) = \top\right] - 1/2\right| = \mathsf{negl}(\lambda)$.

One-wayness to indistinguishability. It is natural to define a one-wayness notion for PKE-SKL, which we call OW-KLA security, by modifying the above definition so that the adversary is required to recover entire bits of a randomly chosen message from its ciphertext. Similarly to standard PKE, we can transform a OW-KLA secure PKE-SKL scheme into an IND-KLA secure one by using (quantum) Goldreich-Levin lemma [AC02, CLLZ21]. Hence, though our goal is to construct an IND-KLA secure scheme, it suffices to construct an OW-KLA secure one.

Basic idea for OW-KLA secure scheme. Towards realizing a OW-KLA secure PKE-SKL scheme, we construct an intermediate scheme $\mathsf{Basic} = (\mathsf{Basic}.\mathcal{KG}, \mathsf{Basic.Enc}, \mathsf{Basic}.\mathcal{Dec}, \mathsf{Basic}.\mathcal{Vrfy})$ using two instances of a standard PKE scheme, with parallel repetition. Let $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a standard PKE scheme. $\mathsf{Basic}.\mathcal{KG}$ generates two key pairs $(\mathsf{ek}_0, \mathsf{dk}_0)$ and $(\mathsf{ek}_1, \mathsf{dk}_1)$ using $\mathsf{PKE.KG}$ and outputs $\mathsf{ek} := (\mathsf{ek}_0, \mathsf{ek}_1)$, $\mathit{dk} := 1/\sqrt{2}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle)$, and $\mathsf{vk} := (\mathsf{dk}_0, \mathsf{dk}_1)$. Given m and ek, $\mathsf{Basic.Enc}$ generates $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_0, \mathsf{m})$ and $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{m})$ and outputs $\mathsf{ct} := (\mathsf{ct}_0, \mathsf{ct}_1)$. $\mathsf{Basic}.\mathcal{Dec}$ can decrypt this ciphertext using the decryption keys dk_0 and dk_1, respectively, in superposition. Since both decryptions result in the same message m, we can decrypt ciphertexts without collapsing dk. Finally, $\mathsf{Basic}.\mathcal{Vrfy}$ checks if the input decryption key is an equal-weight superposition of dk_0 and dk_1. Concretely, it applies a binary outcome measurement w.r.t. a projection $\Pi_{\mathrm{vrfy}} := \frac{1}{2}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle)(\langle 0| \langle \mathsf{dk}_0| + \langle 1| \langle \mathsf{dk}_1|)$, and returns $\top$ if and only if the state is projected onto Π_{vrfy}.

Intuitively, if the adversary has returned the correct decryption key, then it no longer has the capability to decrypt since the decryption key cannot be cloned. However, this scheme does not satisfy OW-KLA because an adversary can pass the verification with probability $1/2$ simply by measuring the decryption key and returning the collapsed

[3] In this paper, standard math or sans serif font stands for classical algorithms and classical variables. The calligraphic font stands for quantum algorithms and the calligraphic font and/or the bracket notation for (mixed) quantum states.

[4] We also consider a slightly stronger definition where the adversary can get access to a verification oracle many times, and the adversary is regarded as valid if the answer to at least one query $\widetilde{\mathit{dk}}$ is $\top$. In this overview, we focus on the "1-query" security for simplicity.

decryption key. Such an adversary can keep the decryption capability even after passing verification because the decryption key collapses to a classical string, which can be easily copied. Nonetheless, it is reasonable to expect that this attack strategy is optimal because there appears to be no obvious way to attack with a better advantage. That said, it is unclear how to turn this intuition into a formal proof assuming only IND-CPA security of the underlying PKE. To address this gap, we introduce a new security notion for PKE, that we call *consistent or inconsistent security against key leasing attacks (CoIC-KLA security)*. Using this, we can prove that the aforementioned adversarial strategy is optimal and Basic satisfies $1/2$-OW-KLA security.

By being $1/2$-OW-KLA secure, we mean that the probability that an adversary can correctly return a decryption key and recover the challenge plaintext simultaneously is at most $1/2 + \mathsf{negl}(\lambda)$. Below, we introduce the definition of CoIC-KLA security and how to prove $1/2$-OW-KLA security of Basic using CoIC-KLA security. Then, we explain how to achieve a full OW-KLA secure scheme by applying parallel amplification to Basic.

Definition of CoIC-KLA security. CoIC-KLA security is defined by using the following game.

1. The challenger generates $(\mathsf{ek}_0, \mathsf{dk}_0)$ and $(\mathsf{ek}_1, \mathsf{dk}_1)$ using $\mathsf{PKE.KG}$, and generates $\mathcal{dK} := 1/\sqrt{2}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle)$. The challenger sends ek_0, ek_1, and $\mathcal{dK}$ to an adversary $\mathcal{A}$. In this game, $\mathcal{A}$ can access the verification oracle only once, where the oracle is given a quantum state and returns the outcome of the projective measurement $(\Pi_{\mathrm{vrfy}}, I - \Pi_{\mathrm{vrfy}})$.

2. $\mathcal{A}$ sends two plaintexts $(\mathsf{m}_0^*, \mathsf{m}_1^*)$ to the challenger. The challenger picks random bits a, b and generates $\mathsf{ct}_0 = \mathsf{Enc}(\mathsf{ek}_0, \mathsf{m}_a)$ and $\mathsf{ct}_1 = \mathsf{Enc}(\mathsf{ek}_1, \mathsf{m}_{a\oplus b})$. Then, the challenger sends ct_0 and ct_1 to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a bit b'.

Then, CoIC-KLA security requires that any QPT $\mathcal{A}$ cannot guess b significantly better than random guessing. In the above game, if $b = 0$, ct_0 and ct_1 are ciphertexts of the same plaintext m_a^*. On the other hand, if $b = 1$, ct_0 and ct_1 are ciphertexts of the different plaintexts m_a^* and $\mathsf{m}_{1\oplus a}^*$. Thus, we call this security notion consistent or inconsistent security.

$1/2$-OW-KLA security of Basic. We explain how to prove $1/2$-OW-KLA security of Basic based on CoIC-KLA security of PKE. The OW-KLA security game for Basic is as follows.

1. The challenger generates $(\mathsf{ek}_0, \mathsf{dk}_0)$ and $(\mathsf{ek}_1, \mathsf{dk}_1)$ using $\mathsf{PKE.KG}$, sets $\mathsf{ek} := (\mathsf{ek}_0, \mathsf{ek}_1)$ and $\mathcal{dK} := 1/\sqrt{2}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle)$, and sends ek and $\mathcal{dK}$ to an adversary $\mathcal{A}$.

2. The adversary returns a quantum state $\widetilde{\mathcal{dK}}$ that is supposed to be a correct decryption key. The challenger checks if the result of applying Π_{vrfy} defined above to $\widetilde{\mathcal{dK}}$ is 1. If not, $\mathcal{A}$ is regarded as invalid and the game ends here. Otherwise, the game goes to the next step.

3. The challenger generates random plaintext m^* and two ciphertexts $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_0, \mathsf{m}^*)$ and $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{m}^*)$, and sends $\mathsf{ct} := (\mathsf{ct}_0, \mathsf{ct}_1)$ to $\mathcal{A}$.

4. $\mathcal{A}$ outputs m'.

In this game, we say that $\mathcal{A}$ wins if (a) $\widetilde{\mathcal{dK}}$ passes the verification, that is, the result of applying Π_{vrfy} to $\widetilde{\mathcal{dK}}$ is 1, and (b) $m' = m^*$ holds. $\mathcal{A}$ can win this game with probability at least $1/2$ by just measuring $1/\sqrt{2}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle)$, returns collapsed key, and decrypt the challenge ciphertext with the key. As stated above, we can prove that this is the optimal strategy for $\mathcal{A}$, that is, we can bound the advantage of $\mathcal{A}$ by $1/2 + \mathsf{negl}(\lambda)$. The proof can be done by using game sequences. We denote the probability that $\mathcal{A}$ wins in Game i as $\Pr[S_i]$.

Game 0**:** This is exactly the above game.

Game 1**:** We defer the verification of the returned key $\widetilde{\mathcal{dK}}$ after $\mathcal{A}$ outputs m'.

From the deferred measurement principle, we have $\Pr[S_0] = \Pr[S_1]$.

Game 2: We change $\mathcal{A}$'s winning condition (b). Concretely, we replace (b) with (b') $\mathsf{m}' \in \{\mathsf{m}^*, \tilde{\mathsf{m}}\}$ holds, where $\tilde{m}$ is a random plaintext.

Since we relaxed $\mathcal{A}$'s winning condition, we have $\Pr[S_1] \leq \Pr[S_2]$.

Game 3: We generate ct_1 as $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \tilde{\mathsf{m}})$ instead of $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{m}^*)$.

The only difference between Game 2 and 3 is that ct_0 and ct_1 are ciphertexts of the same plaintext in Game 2, but they are ciphertexts of different plaintexts in Game 3. Thus, we obtain $|\Pr[S_2] - \Pr[S_3]| = \mathsf{negl}(\lambda)$ using CoIC security of PKE.

We complete the proof by showing that $\Pr[S_3] \leq 1/2 + \mathsf{negl}(\lambda)$ holds if PKE satisfies one-wayness (that is implied by CoIC-KLA security). To show it, we use the following Fact 1.

Fact 1: Assume PKE satisfies one-wayness. Then, given $1/\sqrt{2}(|0\rangle\,|\mathsf{dk}_0\rangle + |1\rangle\,|\mathsf{dk}_1\rangle)$, $\mathsf{PKE.Enc}(\mathsf{ek}_0, \mathsf{m}^*)$, and $\mathsf{PKE.Enc}(\mathsf{ek}_1, \tilde{\mathsf{m}})$, no adversary can obtain $(\mathsf{dk}_0, \tilde{\mathsf{m}})$ or $(\mathsf{dk}_1, \mathsf{m}^*)$ with non-negligible probability.

This can be proved by using the fact that even if we measure $1/\sqrt{2}(|0\rangle\,|\mathsf{dk}_0\rangle + |1\rangle\,|\mathsf{dk}_1\rangle)$ in the computational basis before giving it to the adversary, the adversary still has success probability at least $\epsilon/2$, where ϵ is the success probability of the original experiment [BZ13, Lemma 2.1] (which is stated as Lemma 2.21).

Suppose $\Pr[S_3] = 1/2 + 1/\mathrm{poly}(\lambda)$ for some polynomial poly. This means that conditioned that $\mathsf{m}' \in \{m^*, \tilde{m}\}$, $\widetilde{d\mathcal{K}}$ returned by $\mathcal{A}$ passes the verification with probability significantly greater than $1/2$. Thus, if we measure $\widetilde{d\mathcal{K}}$ in the computational basis, we obtain dk_0 with some inverse polynomial probability and also dk_1 with some inverse polynomial probability. (If either one is obtained with overwhelming probability, $\widetilde{d\mathcal{K}}$ cannot pass the verification with probability significantly greater than $1/2$.) This means that using $\mathcal{A}$, we can obtain either one pair of $(\mathsf{dk}_0, \tilde{\mathsf{m}})$ or $(\mathsf{dk}_1, \mathsf{m}^*)$ with inverse polynomial probability, which contradicts Fact 1. Thus, we obtain $\Pr[S_3] \leq 1/2 + \mathsf{negl}(\lambda)$.

From the above discussions, we can conclude that if PKE satisfies CoIC-KLA security, Basic satisfies $1/2$-OW-KLA security.

Full OW-KLA security by parallel repetition. To achieve a fully OW-KLA secure scheme, we apply parallel amplification to Basic in the following way. When generating a key tuple, we generate λ key tuples $(\mathsf{ek}_i, d\mathcal{K}_i, \mathsf{vk}_i)$ of Basic and set $\mathsf{ek}' := (\mathsf{ek}_i)_{i\in[\lambda]}$, $d\mathcal{K}' := (d\mathcal{K}_i)_{i\in[\lambda]}$, and $\mathsf{vk}' := (\mathsf{vk}_i)_{i\in[\lambda]}$. When encrypting a plaintext m, we divide it into λ pieces $\mathsf{m}_1, \cdots, \mathsf{m}_\lambda$, and encrypt each m_i using ek_i. Then decryption and verification are performed naturally by running the underlying procedures in Basic for every $i \in [\lambda]$. We can prove the full OW-KLA security of this construction using a strategy analogous to that used to achieve $1/2$-OW-KLA security of Basic. We remark that it is unclear whether we can amplify $1/2$-OW-KLA security to full OW-KLA security in a black box way and our security proof relies on the specific structure of our scheme.

Constructing CoIC-KLA secure PKE scheme. In the rest of this overview, we mainly explain how to construct CoIC-KLA secure PKE scheme. We construct it using 1-key Ciphertext-Policy Functional Encryption (CPFE) that in turn can be based on any IND-CPA secure PKE scheme.

We first review the definition of 1-key CPFE scheme. A 1-key CPFE scheme CPFE consists of four algorithms $(\mathsf{FE.Setup}, \mathsf{FE.KG}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$. Given a security parameter, $\mathsf{FE.Setup}$ outputs a master public key mpk and a master secret key msk. $\mathsf{FE.KG}$ takes as input msk and a string x and outputs a decryption key sk_x tied to the string x. $\mathsf{FE.Enc}$ takes as input mpk and a description of a circuit C and outputs a ciphertext ct. If we decrypt this ciphertext ct with sk_x using $\mathsf{FE.Dec}$, we can obtain $C(x)$. The security of it states that ciphertexts of two circuits C_0 and C_1 are computationally indistinguishable for an adversary who has decryption key sk_x for x of its choice, as long as $C_0(x) = C_1(x)$ holds.

Letting $\mathsf{CPFE} = (\mathsf{FE.Setup}, \mathsf{FE.KG}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be a 1-key CPFE scheme, we construct a CoIC secure PKE scheme $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ as follows. $\mathsf{PKE.KG}$ generates $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{CPFE.Setup}(1^\lambda)$ and a decryption key $\mathsf{sk}_x \leftarrow \mathsf{CPFE.KG}(\mathsf{msk}, x)$ for random string x, and outputs an encryption key $\mathsf{ek} := \mathsf{mpk}$ and the corresponding decryption key $\mathsf{dk} := \mathsf{sk}_x$. Given $\mathsf{ek} = \mathsf{mpk}$ and m, $\mathsf{PKE.Enc}$ outputs $\mathsf{FE.Enc}(\mathsf{mpk}, C[\mathsf{m}])$, where $C[\mathsf{m}]$ is the constant circuit that outputs m on any input. Given $\mathsf{dk} = \mathsf{sk}_x$ and ct, $\mathsf{PKE.Dec}$ simply outputs $\mathsf{CPFE.Dec}(\mathsf{sk}_x, \mathsf{ct})$. We see that PKE satisfies decryption correctness from that of CPFE.

Before proving CoIC-KLA security of PKE, we explain a nice tracing property of PKE that plays an important role in the proof. It says that if there exists a decoder that can distinguish $\mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_0^*)$ and $\mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_1^*)$ with probability $1/2 + 1/\mathrm{poly}(\lambda)$ for some plaintexts $\mathsf{m}_0^*, \mathsf{m}_1^*$ and polynomial poly, we can extract the string x tied to the decryption key from the decoder. Concretely, the following fact holds.

Fact 2: Consider the following experiment. The challenger generates $(\mathsf{ek} := \mathsf{mpk}, \mathsf{dk} := \mathsf{sk}_x)$ using PKE.KG and sends them to an adversary $\mathcal{A}$. $\mathcal{A}$ outputs a decoder D together with $\mathsf{m}_0^*, \mathsf{m}_1^*$ that can predict random bit b from $\mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_b^*)$ with probability $1/2 + 1/\mathrm{poly}(\lambda)$ for some polynomial poly. Then, we can extract x from D with inverse polynomial probability.

In fact, if the decoder D is a classical decoder, we can extract x from D with a probability close to 1 as follows. Let $\tilde{C}[b, \mathsf{m}_0, \mathsf{m}_1, i]$ be the circuit that is given x as an input and outputs $\mathsf{m}_{b \oplus x[i]}$, where $x[i]$ is the i-th bit of x. Then, suppose we generate many random $(b, \mathsf{FE.Enc}(\mathsf{mpk}, \tilde{C}[b, \mathsf{m}_0^*, \mathsf{m}_1^*, i]))$ and estimate the probability that the decoder D outputs b given $\mathsf{FE.Enc}(\mathsf{mpk}, \tilde{C}[b, \mathsf{m}_0^*, \mathsf{m}_1^*, i])$ as an input. By the CPFE's security, $\mathsf{FE.Enc}(\mathsf{mpk}, \tilde{C}[b, \mathsf{m}_0^*, \mathsf{m}_1^*, i])$ is indistinguishable from a correctly generated ciphertext of $\mathsf{m}_{b \oplus x_i}^*$, that is, $\mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_{b \oplus x_i}^*) = \mathsf{FE.Enc}(\mathsf{mpk}, C[\mathsf{m}_{b \oplus x_i}^*])$ from the view of $\mathcal{A}$ and D who has sk_x, since $\tilde{C}[b, \mathsf{m}_0^*, \mathsf{m}_1^*, i](x) = C[\mathsf{m}_{b \oplus x_i}^*](x) = \mathsf{m}_{b \oplus x_i}^*$. Then, the result of the estimation should be as follows.

- In the case of $x[i] = 0$, each sample used for the estimation looks $(b, \mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_b))$ from the view of D. Thus, the result of the estimation should be greater than $1/2$ from the fact that D correctly predicts random bit b from $\mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_b)$ with probability $1/2 + 1/\mathrm{poly}(\lambda)$.
- In the case of $x[i] = 1$, each sample used for the estimation looks $(b, \mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_{1 \oplus b}))$ from the view of D. Thus, the result of the estimation should be smaller than $1/2$ since D outputs $1 \oplus b$ given $\mathsf{PKE.Enc}(\mathsf{ek}, \mathsf{m}_{1 \oplus b})$ with probability $1/2 + 1/\mathrm{poly}(\lambda)$.

Therefore, by checking if the result of the estimation is greater than $1/2$ or not, we can extract $x[i]$. By doing this for every i, we can extract entire bits of x.

The above extraction technique is a direct application of that used by Kitagawa and Nishimaki [KN22b] to realize watermarking scheme secure against quantum adversaries. By using their technique, even if the decoder is a quantum decoder $\mathcal{D}$ that consists of a unitary and an initial quantum state, we can extract x from $\mathcal{D}$ with inverse polynomial probability, as long as $\mathcal{D}$ has a high distinguishing advantage. Roughly speaking, this is done by performing the above estimation using (approximate) projective implementation proposed by Zhandry [Zha20] that is based on the technique by Marriott and Watrous [MW05]. By extending the above extraction technique, we can obtain the following fact.

Fact 3: Consider the following experiment. The challenger generates $(\mathsf{ek}_0 := \mathsf{mpk}_0, \mathsf{dk}_0 := \mathsf{sk}_{x_0})$ and $(\mathsf{ek}_1 := \mathsf{mpk}_1, \mathsf{dk}_1 := \mathsf{sk}_{x_1})$ using PKE.KG, and sends ek_0, ek_1, and $1/\sqrt{2}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle) = 1/\sqrt{2}(|0\rangle |\mathsf{sk}_{x_0}\rangle + |1\rangle |\mathsf{sk}_{x_1}\rangle)$ to an adversary $\mathcal{A}$. $\mathcal{A}$ outputs a quantum decoder $\mathcal{D}$ together with $(\mathsf{m}_0^*, \mathsf{m}_1^*)$ that can predict b from $\mathsf{PKE.Enc}(\mathsf{ek}_0, \mathsf{m}_a)$ and $\mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{m}_{a \oplus b})$ with probability $1/2 + 1/\mathrm{poly}(\lambda)$ for some polynomial poly. Then, we can extract both x_0 and x_1 from $\mathcal{D}$ with inverse polynomial probability.

We now explain how we can prove CoIC-KLA security of PKE using Fact 3. To this end, we introduce one more fact.

Fact 4: Given mpk_0, mpk_1, and $1/\sqrt{2}(|0\rangle |\mathsf{sk}_{x_0}\rangle + |1\rangle |\mathsf{sk}_{x_1}\rangle)$, where $(\mathsf{mpk}_0, \mathsf{sk}_{x_0})$ and $(\mathsf{mpk}_1, \mathsf{sk}_{x_1})$ are generated as in PKE.KG, no adversary can compute both x_0 and x_1 with non-negligible probability.

Similarly to Fact 1, we can prove this from the fact that even if we measure $1/\sqrt{2}(|0\rangle |\mathsf{sk}_{x_0}\rangle + |1\rangle |\mathsf{sk}_{x_1}\rangle)$ in the computational basis before giving it to the adversary, the adversary still has success probability at least $\epsilon/2$, where ϵ is the success probability of the original experiment [BZ13, Lemma 2.1].

Suppose there exists a QPT adversary $\mathcal{A}$ that breaks CoIC-KLA security of PKE. We consider the following adversary $\mathcal{B}$ using $\mathcal{A}$. Given mpk_0, mpk_1, and $1/\sqrt{2}(|0\rangle |\mathsf{sk}_{x_0}\rangle + |1\rangle |\mathsf{sk}_{x_1}\rangle)$, $\mathcal{B}$ simulates CoIC-KLA security game for $\mathcal{A}$ by setting $\mathsf{ek}_0 := \mathsf{mpk}_0$, $\mathsf{ek}_1 := \mathsf{mpk}_1$, and $\mathsf{dk} := 1/\sqrt{2}(|0\rangle |\mathsf{sk}_{x_0}\rangle + |1\rangle |\mathsf{sk}_{x_1}\rangle)$ until $\mathcal{A}$ outputs two plaintexts $(\mathsf{m}_0^*, \mathsf{m}_1^*)$. When $\mathcal{A}$ makes a verification query, $\mathcal{B}$ just returns a random bit. Let U be the unitary that performs the

rest of $\mathcal{A}$'s actions given the challenge ciphertexts. Also, let q be the internal state of $\mathcal{A}$ at this point. Then, from the averaging argument and the fact that $\mathcal{B}$ correctly answers to $\mathcal{A}$'s verification query with probability $1/2$, with some inverse polynomial probability, the quantum decoder $\mathcal{D} = (\boldsymbol{U}, q)$ is a decoder that can predict b from $\mathsf{PKE.Enc}(\mathsf{ek}_0, \mathsf{m}_a^*)$ and $\mathsf{PKE.Enc}(\mathsf{ek}_1, \mathsf{m}_{a\oplus b}^*)$ with probability $1/2 + 1/\mathrm{poly}(\lambda)$ for some polynomial poly. Thus, by using the extractor that is guaranteed to exist by Fact 3, $\mathcal{B}$ can obtain both x_0 and x_1 with some inverse polynomial probability, which contradicts Fact 4. This means that PKE satisfies CoIC-KLA security.

Extension to Advanced Encryption Systems with Secure Key Leasing. We also provide constructions of advanced encryption schemes such as ABE and FE with secure key leasing. We do not focus on IBE in this paper since IBE is a special case of ABE and our transformation preserves the underlying function class.[5] We construct these schemes by carefully combining standard ABE (resp. FE) with PKE-SKL in the way that each decryption key of the resulting ABE-SKL (resp. FE-SKL) scheme includes a decryption key of the underlying PKE-SKL scheme and a ciphertext of the ABE-SKL (resp. FE-SKL) scheme cannot be decrypted without the decryption key of the underlying PKE-SKL scheme. By doing so, our ABE-SKL and FE-SKL take over the secure key leasing security from the underlying PKE-SKL. Moreover, since PKE-SKL can be based on any PKE, our ABE-SKL and FE-SKL can be based on any standard ABE and FE, respectively.

ABE-SKL. Here, we provide an overview of ABE with secure key leasing. Let us start with the definition of plain ABE (without key leasing). An ABE scheme ABE consists of four algorithms $(\mathsf{ABE.Setup}, \mathsf{ABE.KG}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ and is associated with a relation R. Given a security parameter, $\mathsf{ABE.Setup}$ outputs a master public key mpk and a master secret key msk. $\mathsf{ABE.KG}$ takes as input msk and a key attribute y and outputs a user secret key sk_y tied to the attribute y. $\mathsf{ABE.Enc}$ takes as input mpk, a ciphertext attribute x, and a message m and outputs a ciphertext ct. The decryption of the ciphertext is possible only when $R(x,y) = 1$. For this reason, we call a user secret key for attribute y satisfying $R(x,y) = 1$ a decrypting key (for a ciphertext associated with x). As for the security, we require that $\mathsf{ABE.Enc}(x^*, \mathsf{m}_0^*)$ should be computationally indistinguishable from $\mathsf{ABE.Enc}(x^*, \mathsf{m}_1^*)$ as long as an adversary is only given non-decrypting keys for the ciphertext (i.e., user secret keys for y satisfying $R(x^*, y) = 0$).

We now define the notion of ABE with secure key leasing (ABE-SKL) by extending the syntax of ABE. The difference from the above is that the key generation algorithm is now quantum and it outputs user secret key usk_y along with verification key vk. We also additionally introduce a verification algorithm that takes vk and a quantum state usk' and outputs $\top$ if it judges that the user secret key corresponding to vk is correctly returned and $\bot$ otherwise. As for the security, we require that $\mathsf{ABE.Enc}(x^*, \mathsf{m}_0)$ should be computationally indistinguishable from $\mathsf{ABE.Enc}(x^*, \mathsf{m}_1)$ if the adversary returns all decrypting keys before it is given the challenge ciphertext. Here, we say the adversary returns the key if the adversary provides the challenger with a quantum state that makes the verification algorithm output $\top$.

For the construction, the basic idea is to use ABE for access control and PKE-SKL for obtaining security against key leasing attacks. To enable this idea, we encrypt a message m for an attribute x so that the decryptor recovers PKE-SKL ciphertext $\mathsf{skl.ct} = \mathsf{SKL.Enc}(\mathsf{skl.ek}, \mathsf{m})$ if it has decrypting key and nothing otherwise, where $\mathsf{skl.ek}$ is an individual encryption key corresponding to the user. The user is given the corresponding decryption key $\mathsf{skl.dk}$ and can recover the message by decrypting $\mathsf{skl.ct}$. Roughly speaking, the security follows since (1) a user with a non-decrypting key cannot obtain any information and (2) even a user with a decrypting key cannot recover the message from $\mathsf{skl.ct}$ once it returns $\mathsf{skl.dk}$ due to the security of SKL.

The generation of user individual SKL ciphertext is somewhat non-trivial since ABE can only encrypt a single message. In order to achieve this, we use an idea similar to [SS10, GKW16] that combines encryption with the garbled circuits. In particular, we garble the encryption circuit of SKL that hardwires a message and encrypt the labels by ABE. We then provide a secret key of ABE for a user only for the positions corresponding to $\mathsf{skl.ek}$. This allows a user with decrypting key to recover the labels corresponding to $\mathsf{skl.ek}$ and then run the garbled circuit on input the labels to recover $\mathsf{skl.ct}$.

Unfortunately, the introduction of the garbled circuits in the construction poses some limitations on the security of the scheme. In particular, once the adversary obtains two decrypting user secret keys, the message can be revealed from the garbled circuit in the ciphertext since the security of garbled circuits is compromised when labels for two different

[5] Although ABE is a special case of FE, we need stronger assumptions for (collusion-resistant) FE to instantiate them. In addition, the security level of FE-SKL that we can achieve is different from that of ABE-SKL. Hence, we consider both ABE and FE.

inputs are revealed. Therefore, we are only able to prove 1-bounded distinguishing key security,[6] where the adversary can make a single decrypting key query and should return the key before the challenge ciphertext is given. We note that the adversary can make an arbitrary number of non-decrypting key queries throughout the game, unlike bounded collusion ABE [GVW12, ISV+17] and only the number of decrypting keys is bounded.

Ideally, we would like to have a scheme without restriction on the number of decrypting keys. However, we do not know how to achieve it without strong assumptions like functional encryption or indistinguishability obfuscation. Instead, we achieve intermediate security notion that we call q-bounded distinguishing key security without introducing additional assumption, where the number of decrypting keys is bounded by some pre-determined polynomial. To do so, we use the same idea as [ISV+17], which converts single bounded collusion ABE into q-bounded collusion ABE. The construction is based on the balls and bins idea, where we prepare multiple "bins", each of which consists of multiple instances of 1-bounded distinguishing key secure ABE-SKL $\mathsf{1ABE}$. The key generation algorithm chooses a single instance from each bin randomly and generates a user secret key for each of them. The encryption algorithm secret shares the message and encrypts them using the instances of the $\mathsf{1ABE}$ so that the same share is encrypted by the instances in the same bin. By careful choices of the parameters and analysis, in the security proof, we can argue that there exists a bin such that $\mathsf{1ABE}$ instances used for generating decrypting keys in that bin are all distinct. This means that for every $\mathsf{1ABE}$ instance in that bin, only a single decrypting key is generated and thus, we can use 1-bounded distinguishing key security for each of them. While this overall proof strategy is the same as [ISV+17], our proof is a little bit more complex than theirs because the adversary is allowed to make an unbounded number of (non-decrypting) key queries. We refer to Section 6 for further details.

PKFE-SKL. We move to the overview of PKFE-SKL. In this work, we focus on Key-Policy FE (KPFE) with secure key leasing. We start with the definition of plain FE (without key leasing). An FE scheme FE consists of four algorithms $(\mathsf{FE.Setup}, \mathsf{FE.KG}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ and is associated with a function class $\mathcal{F}$. Given a security parameter, $\mathsf{FE.Setup}$ outputs a public key pk and a master secret key msk. $\mathsf{FE.KG}$ takes as input msk and a function $f \in \mathcal{F}$ and outputs a functional decryption key sk_f tied to the function f. $\mathsf{FE.Enc}$ takes as input pk and a plaintext x and outputs a ciphertext ct. The decryption result is $f(x)$. For security, we require that $\mathsf{FE.Enc}(\mathsf{pk}, x_0)$ should be computationally indistinguishable from $\mathsf{FE.Enc}(\mathsf{pk}, x_1)$ as long as an adversary is only given functional decryption keys for $\{f_i\}_i$ such that $f_i(x_0) = f_i(x_1)$ for all i.

We define the notion of FE with secure key leasing (FE-SKL) by extending the syntax of FE like ABE-SKL. The key generation algorithm is now quantum and it outputs functional decryption key sk_f along with verification key vk. We also introduce a verification algorithm that takes vk and a quantum state sk' and outputs $\top$ if it judges that the functional decryption key corresponding to vk is correctly returned and $\bot$ otherwise.

In the security game of PKFE-SKL, the adversary can send a *distinguishing* key query f such that $f(x_0^*) \neq f(x_1^*)$ where (x_0^*, x_1^*) are the challenge plaintexts *as long as it returns a valid functional decryption key for* f. We consider a security game where the adversary can send unbounded polynomially many distinguishing and non-distinguishing (that is, $f(x_0^*) = f(x_1^*)$) key queries and tries to distinguish $\mathsf{FE.Enc}(\mathsf{pk}, x_0)$ from $\mathsf{FE.Enc}(\mathsf{pk}, x_1)$.

We transform a (classical) PKFE scheme into a PKFE scheme with secure key leasing by using the power of PKE-SKL. The basic idea is as follows. When we generate a functional decryption key for function f, we generate a key triple of PKE-SKL and a functional decryption key of the classical PKFE for a function W that computes a PKE-SKL ciphertext of $f(x)$. That is, we wrap $f(x)$ by PKE-SKL encryption. A decryption key of PKE-SKL is appended to $\mathsf{fe.sk}_W$, which is the functional decryption key for W. Hence, we can decrypt the PKE-SKL ciphertext and obtain $f(x)$. The PKE-SKL decryption key for f is useless for another function g since we use different key triples of PKE-SKL for each function.

More specifically, we generate PKE-SKL keys $(\mathsf{skl.ek}, \mathsf{skl.}\mathsf{sk}, \mathsf{skl.vk})$ and a PKFE functional decryption key $\mathsf{fe.sk}_W \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f, \mathsf{skl.ek}])$, where function $W[f, \mathsf{skl.ek}]$ takes as input x and outputs a PKE-SKL ciphertext $\mathsf{SKL.Enc}(\mathsf{skl.ek}, f(x))$.[7] A functional decryption key for f consists of $(\mathsf{fe.sk}_W, \mathsf{skl.}\mathsf{sk})$. A ciphertext of x is a (classical) PKFE ciphertext $\mathsf{FE.Enc}(\mathsf{fe.pk}, x)$. If we return $\mathsf{skl.}\mathsf{sk}$ for f (verified by $\mathsf{skl.vk}$) before we obtain $\mathsf{FE.Enc}(\mathsf{fe.pk}, x)$, we cannot obtain $f(x)$ from $\mathsf{SKL.Enc}(\mathsf{skl.ek}, f(x))$ by the security of PKE-SKL.

[6] When we consider the security game for ABE-SKL, a decrypting key can be used for distinguishing the challenge bit by decrypting the challenge ciphertext (if it is not returned). Therefore, we use the term "decrypting key" and "distinguishing key" interchangeably.

[7] We ignore the issue of encryption randomness here. In our construction, we use (puncturable) PRFs to generate encryption randomness.

We need to prove security against an adversary that obtains a functional decryption key for f such that $f(x_0^*) \neq f(x_1^*)$ where (x_0^*, x_1^*) is a pair of challenge plaintexts if the adversary returns the functional decryption key. To handle this issue, we rely on IND-KLA security and need to embed a challenge ciphertext of PKE-SKL into a PKFE ciphertext. We use the trapdoor method of FE (a.k.a. Trojan method) [ABSV15, BS18] for this purpose. We embed an SKFE functional decryption key and ciphertext in a PKFE functional decryption key and ciphertext, respectively. We use these SKFE functional decryption key and ciphertext for the trapdoor mode of PKFE. We gradually change SKFE ciphertexts and keys so that we can embed a PKE-SKL challenge ciphertext by using the adaptively single-ciphertext function privacy of SKFE. Once we succeed in embedding a PKE-SKL challenge ciphertext, we can change a ciphertext of x_0^* into a ciphertext of x_1^* such that $f(x_0^*) \neq f(x_1^*)$ as long as the functional decryption key $\mathsf{sk}_f = (\mathsf{fe.sk}_W, \mathsf{skl}.\mathsf{sk})$ for f is returned. This is because $\mathsf{skl}.\mathsf{sk}$ is returned and we can use IND-KLA security under $\mathsf{skl}.\mathsf{ek}$. See Section 7 for more details.

1.4 Other Related Work

Quantum Copy Protection. Aaronson [Aar09] introduced the notion of quantum copy protection and constructed a quantum copy protection scheme for arbitrary unlearnable Boolean functions relative to a quantum oracle. He also provided two heuristic copy-protection schemes for point functions in the standard model. Coladangelo et al. [CMP20] provided a quantum copy-protection scheme for a class of evasive functions in the QROM. Subsequently, Aaronson et al. [ALL+21] constructed a quantum copy protection scheme for unlearnable functions relative to classical oracles. By instantiating the oracle with post-quantum candidate obfuscation schemes, they obtained a heuristic construction of copy protection. Coladangelo et al. [CLLZ21] provided a copy-protection scheme for pseudorandom functions in the plain model assuming iO, OWF and extractable witness encryption, or assuming subexponential iO, subexponential OWF, LWE and a strong "monogamy property" (which was was proven to be true in a follow-up work [CV22]). Ananth et al. [AK21, AKL+22] also constructed copy protection for point functions, which in turn can be transformed into copy protection for compute-and-compare programs. Sattath and Wyborski [SW22] studied unclonable decryptors, which are an extension of SDE. Their unclonable decryptors scheme is *secret key* encryption and can be instantiated with iO and OWF, or quantum oracles.

Secure software leasing. Secure software leasing (SSL) was introduced by Ananth and La Placa [AL21], where they also provided the first SSL scheme supporting a subclass of "evasive" functions by relying on the existence of public key quantum money and the learning with errors assumption. Evasive functions is a class of functions for which it is hard to find an accepting input given only black-box access to the function. Their construction achieves a strong security notion called *infinite term security*. They also demonstrate that there exists an unlearnable function class such that it is impossible to achieve an SSL scheme for that function class, even in the CRS model. Later, Coladangelo et al. [CMP20] improved the security notion achieved by [AL21] by relying on the QROM, for the same class of evasive functions. Additionally, Kitagawa, Nishimaki and Yamakawa [KNY21] provided a finite term secure SSL scheme for pseudorandom functions (PRFs) in the CRS model by assuming the hardness of the LWE problem against polynomial time quantum adversaries. Additionally, this work achieves classical communication. Further, Broadbent et al. [BJL+21] showed that SSL is achievable for the aforementioned evasive circuits without any setup or computational assumptions that were required by previous work, but with finite term security, quantum communication and correctness based on a distribution. The notion of secure leasing for the powerful primitive of functional encryption was studied by Kitagawa and Nishimaki [KN22a], who introduced the notion of *secret key* functional encryption (SKFE) with secure key leasing and provided a transformation from standard SKFE into SKFE with secure key leasing without relying on any additional assumptions.

Certified deletion. Broadbent and Islam [BI20] introduced the notion of quantum encryption with certified deletion, where we can generate a (classical) certificate to ensure that *a ciphertext* is deleted. They constructed a one-time SKE scheme with certified deletion without computational assumptions. After that, many works presented various quantum encryption primitives (PKE, ABE, FE and so on) with certified deletion [HMNY21, Por23, BK22, HMNY22]. The root of quantum encryption with certified deletion is revocable quantum time-released encryption by Unruh [Unr15]. It is an extension of time-released encryption where a sender can revoke quantum encrypted data before a pre-determined time. If the revocation succeeds, the receiver cannot obtain the plaintext information.

Related technique. The basic idea of our PKE-SKL is to prepare a superposition of two decryption keys and coherently run the decryption algorithm in each branch. Previous works by Zhang [Zha21, Zha22] use a similar idea of running some algorithm (which is an evaluation of "lookup tables" in their case) on two branches in superposition though their motivation is to construct efficient blind quantum computation and classical verification of quantum computation, which are completely irrelevant to PKE-SKL.

1.5 Concurrent Work

A concurrent and independent work by Ananth, Poremba, and Vaikuntanathan [APV23] introduces key-revocable PKE, which is similar to PKE-SKL. They construct key-revocable PKE based on the LWE assumption while our construction of PKE-SKL only assumes the existence of IND-CPA secure PKE. In addition, they only prove somewhat weaker security notion called 1-bit unpredictability. Roughly, it ensures that the probability that the adversary passes the verification for the returned key *and* wins the IND game is at most $1/2 + \mathsf{negl}(\lambda)$. For example, even if an adversary passes the verification with probability $1/3$ and has a distinguishing advantage 1 conditioned on the acceptance, it is not considered to break the security while such an adversary breaks IND-KLA security. Thus, we believe that IND-KLA security is more desirable security notion than 1-bit unpredictability.[8] On the other hand, the advantages of their work are that their construction of key-revocable PKE is based on dual-Regev encryption, which is likely to be more efficient than our PKE-SKL, and that they also show a fully homomorphic encryption variant.

1.6 Organization of the paper

In Section 2 we define the notation and preliminaries that we require in this work. In Section 3, we define the notion of public key encryption with secure key leasing (PKE-SKL) and its various security notions. We also show several general relationships among those security notions. In Section 4, we define and construct Public Key Encryption with CoIC-KLA security. In Section 5, we provide our construction of PKE with secure key leasing. In Section 6 and Section 7 we provide our construction of Attribute Based Encryption with secure key leasing and public key Functional Encryption with secure key leasing respectively.

2 Preliminaries

Notations and conventions. In this paper, standard math or sans serif font stands for classical algorithms (e.g., C or Gen) and classical variables (e.g., x or pk). Calligraphic font stands for quantum algorithms (e.g., $\mathcal{Gen}$) and calligraphic font and/or the bracket notation for (mixed) quantum states (e.g., q or $|\psi\rangle$).

Let $[\ell]$ denote the set of integers $\{1, \cdots, \ell\}$, λ denote a security parameter, and $y := z$ denote that y is set, defined, or substituted by z. For a finite set X and a distribution D, $x \leftarrow X$ denotes selecting an element from X uniformly at random, $x \leftarrow D$ denotes sampling an element x according to D. Let $y \leftarrow \mathsf{A}(x)$ and $y \leftarrow \mathcal{A}(\chi)$ denote assigning to y the output of a probabilistic or deterministic algorithm A and a quantum algorithm $\mathcal{A}$ on an input x and χ, respectively. When we explicitly show that A uses randomness r, we write $y \leftarrow \mathsf{A}(x;r)$. PPT and QPT algorithms stand for probabilistic polynomial-time algorithms and polynomial-time quantum algorithms, respectively. Let negl denote a negligible function. For strings $x, y \in \{0,1\}^n$, $x \cdot y$ denotes $\bigoplus_{i\in[n]} x_i y_i$ where x_i and y_i denote the ith bit of x and y, respectively.

2.1 Standard Cryptographic Tools

Secret-key encryption.

Definition 2.1 (Secret Key Encryption). *An SKE scheme* SKE *is a two tuple* (E, D) *of PPT algorithms.*

[8] Strictly speaking, IND-KLA security and 1-bit unpredictability are incomparable because the former requires the indistinguishability between ciphertexts of two different messages whereas the latter requires the indistinguishability between a ciphertext of some message and a uniformly random string.

- *The encryption algorithm* E*, given a key* $K \in \{0,1\}^\lambda$ *and a plaintext* $m \in \mathcal{M}$*, outputs a ciphertext* ct*, where* $\mathcal{M}$ *is the plaintext space of* SKE*.*
- *The decryption algorithm* D*, given a key* K *and a ciphertext* ct*, outputs a plaintext* $\tilde{m} \in \{\bot\} \cup \mathcal{M}$*. This algorithm is deterministic.*

We require SKE *to satisfy correctness.*

Correctness: *We require* $\mathsf{D}(K, \mathsf{E}(K, m)) = m$ *for every* $m \in \mathcal{M}$ *and key* $K \in \{0,1\}^\lambda$*.*

Definition 2.2 (Ciphertext Pseudorandomness for SKE). *Let* $\{0,1\}^\ell$ *be the ciphertext space of* SKE*. We define the following experiment* $\mathsf{Exp}^{\mathsf{pr\text{-}ct}}_{\mathcal{A},\mathsf{SKE}}(1^\lambda, \mathsf{coin})$ *between a challenger and an adversary* $\mathcal{A}$*.*

1. *The challenger generates* $K \leftarrow \{0,1\}^\lambda$*. Then, the challenger sends* 1^λ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *may make polynomially many encryption queries adaptively.* $\mathcal{A}$ *sends* $m \in \mathcal{M}$ *to the challenger. Then, the challenger returns* $\mathsf{ct} \leftarrow \mathsf{E}(K, m)$ *if* $\mathsf{coin} = 0$*, otherwise* $\mathsf{ct} \leftarrow \{0,1\}^\ell$*.*
3. $\mathcal{A}$ *outputs* $\mathsf{coin}' \in \{0,1\}$*.*

We say that SKE *is pseudorandom-secure if for any QPT adversary* $\mathcal{A}$*, we have*

$$\mathsf{Adv}^{\mathsf{pr\text{-}ct}}_{\mathsf{SKE},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{pr\text{-}ct}}_{\mathcal{A},\mathsf{SKE}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{pr\text{-}ct}}_{\mathcal{A},\mathsf{SKE}}(1^\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Theorem 2.3. *If OWFs exist, there exists a pseudorandom-secure SKE scheme.*

Public-key encryption.

Definition 2.4 (PKE). *A PKE scheme* PKE *is a tuple of three algorithms* $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$*. Below, let* $\mathcal{X}$ *be the message space of* PKE*.*

$\mathsf{KG}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk})$**:** *The key generation algorithm takes a security parameter* 1^λ*, and outputs an encryption key* ek *and a decryption key* dk*.*

$\mathsf{Enc}(\mathsf{ek}, \mathsf{m}) \to \mathsf{ct}$**:** *The encryption algorithm takes an encryption key* ek *and a message* $\mathsf{m} \in \mathcal{X}$*, and outputs a ciphertext* ct*.*

$\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}) \to \tilde{\mathsf{m}}$**:** *The decryption algorithm is a deterministic algorithm that takes a decryption key* dk *and a ciphertext* ct*, and outputs a value* $\tilde{\mathsf{m}}$*.*

Correctness: *For every* $\mathsf{m} \in \mathcal{X}$*, we have*

$$\Pr\left[\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}) = \mathsf{m} \;\middle|\; \begin{array}{l} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 2.5 (IND-CPA Security). *We say that a PKE scheme* PKE *with the message space* $\mathcal{X}$ *is IND-CPA secure if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger:*

1. *The challenger runs* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(1^\lambda)$ *and sends* ek *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends* $(\mathsf{m}^*_0, \mathsf{m}^*_1) \in \mathcal{X}^2$ *to the challenger.*
3. *The challenger generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}^*_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* coin' *for* coin*. The challenger outputs* coin' *as the final output of the experiment.*

For any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda,0)\to 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda,1)\to 1\right]\right| \le \mathsf{negl}(\lambda).$$

Definition 2.6 (OW-CPA Security). *We say that a PKE scheme PKE with the message space $\mathcal{X}$ is OW-CPA secure if it satisfies the following requirement, formalized from the experiment $\mathsf{Exp}^{\mathsf{ow\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$ between an adversary $\mathcal{A}$ and a challenger:*

1. *The challenger runs $(\mathsf{ek},\mathsf{dk})\leftarrow\mathsf{KG}(1^\lambda)$, chooses $\mathsf{m}^*\leftarrow\mathcal{X}$, runs $\mathsf{ct}^*\leftarrow\mathsf{Enc}(\mathsf{ek},\mathsf{m}^*)$, and sends $(\mathsf{ek},\mathsf{ct}^*)$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends $\mathsf{m}'\in\mathcal{X}$ to the challenger.*
3. *The challenger outputs 1 if $\mathsf{m}'=\mathsf{m}^*$ and otherwise 0 as the final output of the experiment.*

For any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{ow\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{ow\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)\to 1\right] \le \mathsf{negl}(\lambda).$$

It is well-known that IND-CPA security implies OW-CPA security if $|\mathcal{X}|$ is super-polynomial.

Pseudorandom functions.

Definition 2.7 (Puncturable PRF). *A puncturable PRF (PPRF) is a tuple of algorithms $\mathsf{PPRF}=(\mathsf{PRF.Gen},\mathsf{F},\mathsf{Puncture})$ where $\{\mathsf{F}_\mathsf{K}:\{0,1\}^{\ell_1}\to\{0,1\}^{\ell_2}\mid \mathsf{K}\in\{0,1\}^\lambda\}$ is a PRF family and satisfies the following two conditions. Note that ℓ_1 and ℓ_2 are polynomials of λ.*

Punctured correctness: *For any polynomial-size set $S\subseteq\{0,1\}^{\ell_1}$ and any $x\in\{0,1\}^{\ell_1}\setminus S$, it holds that*

$$\Pr\left[\mathsf{F}_\mathsf{K}(x)=\mathsf{F}_{\mathsf{K}_{\notin S}}(x)\mid \mathsf{K}\leftarrow\mathsf{PRF.Gen}(1^\lambda),\mathsf{K}_{\notin S}\leftarrow\mathsf{Puncture}(\mathsf{K},S)\right]=1.$$

Pseudorandom at punctured point: *For any polynomial-size set $S\subseteq\{0,1\}^{\ell_1}$ and any QPT distinguisher $\mathcal{A}$, it holds that*

$$|\Pr[\mathcal{A}(\mathsf{F}_{\mathsf{K}_{\notin S}},\{\mathsf{F}_\mathsf{K}(x_i)\}_{x_i\in S})\to 1] - \Pr\left[\mathcal{A}(\mathsf{F}_{\mathsf{K}_{\notin S}},(\mathcal{U}_{\ell_2})^{|S|})\to 1\right]| \le \mathsf{negl}(\lambda),$$

where $\mathsf{K}\leftarrow\mathsf{PRF.Gen}(1^\lambda)$, $\mathsf{K}_{\notin S}\leftarrow\mathsf{Puncture}(\mathsf{K},S)$ and $\mathcal{U}_{\ell_2}$ denotes the uniform distribution over $\{0,1\}^{\ell_2}$.

If $S=\{x^*\}$ (i.e., puncturing a single point), we simply write $\mathsf{F}_{\neq x^*}(\cdot)$ instead of $\mathsf{F}_{\mathsf{K}_{\notin S}}(\cdot)$ and consider $\mathsf{F}_{\neq x^*}$ as a keyed function.

It is easy to see that the Goldwasser-Goldreich-Micali tree-based construction of PRFs (GGM PRF) [GGM86] from OWF yield puncturable PRFs where the size of the punctured key grows polynomially with the size of the set S being punctured [BW13, BGI14, KPTZ13]. Thus, we have:

Theorem 2.8 ([GGM86, BW13, BGI14, KPTZ13]). *If OWFs exist, then for any polynomials $\ell_1(\lambda)$ and $\ell_2(\lambda)$, there exists a PPRF that maps ℓ_1-bits to ℓ_2-bits.*

Garbling schemes.

Definition 2.9 (Garbling schemes). *A garbling scheme GC is a tuple of PPT algorithms $\mathsf{GC}=(\mathsf{Grbl},\mathsf{GCEval})$.*

$\mathsf{Grbl}(1^\lambda,C)\to(\{\mathsf{lab}_{i,b}\}_{i\in[\ell],b\in\{0,1\}},\widetilde{C})$**:** *The garbling algorithm takes a security parameter 1^λ and a circuit C and outputs labels $\{\mathsf{lab}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ and garbled version of the circuit $\widetilde{C}$, where ℓ is the input length of C.*

$\mathsf{GCEval}(\widetilde{C}, \{\mathsf{lab}_i\}_{i\in[\ell]}) \to z$**:** *The evaluation algorithm* GCEval *takes the garbled circuit* $\widetilde{C}$ *and labels* $\{\mathsf{lab}_i\}_{i\in[\ell]}$ *and outputs an evaluation result* z*.*

Correctness: *We require that*

$$\Pr\left[\mathsf{GCEval}(\widetilde{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[\ell]}) = C(x) \;\middle|\; \mathsf{Grbl}(1^\lambda, C)) \to (\{\mathsf{lab}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \widetilde{C}) \;\right] = 1 - \mathsf{negl}(\lambda)$$

holds for all $\ell \in \mathbb{N}$*,* $x \in \{0,1\}^\ell$ *and* C *with input length* ℓ*, where* x_i *is the* i*-th bit of* x*.*

Security: *We require that there exists a PPT algorithm* Sim.GC *such the following distributions are computationally indistinguishable for all* $\ell \in \mathbb{N}$*,* $x \in \{0,1\}^\ell$*, and circuit* C *with input length* ℓ*:*

$$(\{\mathsf{lab}_{i,x_i}\}_{i\in[\ell]}, \widetilde{C}) \approx_c \mathsf{Sim.GC}(1^\lambda, \mathsf{info}(C), C(x))$$

where $\mathsf{Grbl}(1^\lambda, C)) \to (\{\mathsf{lab}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \widetilde{C})$ *and* $\mathsf{info}(C)$ *refers to the size of* C*, input and output lengths of* C*.*

We note that we will drop $\mathsf{info}(C)$ *from the inputs to* Sim.GC *when it is clear from the context.*

Theorem 2.10. *[Yao86, LP09] If there exists a one-way function, there exists secure garbling scheme.*

Attribute-based encryption.

Definition 2.11 (Attribute-Based Encryption). *An ABE scheme* ABE *is a tuple of four PPT algorithms* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$*. Below, let* $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$*,* $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_\lambda$*, and* $R = \{R_\lambda : \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}\}_\lambda$ *be the ciphertext attribute space, key attribute space, and the relation associated with* ABE*, respectively. We note that we will abuse the notation and occasionally drop the subscript for these spaces for notational simplicity. We also note that the message space is set to be* $\{0,1\}^\ell$ *below.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{msk})$**:** *The setup algorithm takes a security parameter* 1^λ *and outputs a public key* pk *and master secret key* msk*.*

$\mathsf{KG}(\mathsf{msk}, y) \to \mathsf{sk}_y$**:** *The key generation algorithm* KG *takes a master secret key* msk *and a key attribute* $y \in \mathcal{Y}$*, and outputs a decryption key* sk_y*.*

$\mathsf{Enc}(\mathsf{pk}, x, m) \to \mathsf{ct}$**:** *The encryption algorithm takes a public key* pk*, a ciphertext attribute* $x \in \mathcal{X}$*, and a message* x*, and outputs a ciphertext* ct*.*

$\mathsf{Dec}(\mathsf{sk}_y, x, \mathsf{ct}) \to z$**:** *The decryption algorithm takes a secret key* sk_f*, a ciphertext attribute* x*, and the corresponding ciphertext* ct *and outputs* $z \in \{\bot\} \cup \{0,1\}^\ell$*.*

Correctness: *We require that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_y, x, \mathsf{ct}) = m \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda),\\ \mathsf{sk}_y \leftarrow \mathsf{KG}(\mathsf{msk}, y),\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x, m)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

holds for all $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *such that* $R(x,y) = 1$ *and* $m \in \{0,1\}^\ell$*.*

Definition 2.12 (Adaptive Security for ABE). *We say that* ABE *is an* adaptively secure *ABE scheme for relation* $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger:*

1. *The challenger runs* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends arbitrary key queries. That is,* $\mathcal{A}$ *sends a key attribute* $y \in \mathcal{Y}$ *to the challenger and the challenger responds with* $\mathsf{sk}_y \leftarrow \mathsf{KG}(\mathsf{msk}, y)$ *for the query.*

3. *At some point, $\mathcal{A}$ sends (x, m_0, m_1) to the challenger. If $R(x,y) = 0$ for all queried y, the challenger generates a ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x, m_{\mathsf{coin}})$. The challenger sends ct^* to $\mathcal{A}$.*

4. *Again, $\mathcal{A}$ can send key queries y such that $R(x,y) = 0$.*

5. *$\mathcal{A}$ outputs a guess coin' for coin.*

6. *The experiment outputs coin'.*

We say that ABE *is adaptively secure if, for any QPT $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{ada\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda, 0) \to 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda, 1) \to 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 2.13 (Selective Security for ABE). *We also define selective security for* ABE*. For doing so, we consider the same security game as that for adaptive security except that the adversary $\mathcal{A}$ should declare its target x at the beginning of the game (even before it is given* pk*). We then define the advantage $\mathsf{Adv}^{\mathsf{sel\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ for the selective security similarly. We say* ABE *is selectively indistinguishably-secure if for any QPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{sel\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ is negligible.*

By setting $\mathcal{X}$, $\mathcal{Y}$, and R appropriately, we can recover important classes of ABE. In particular, if we set $\mathcal{X}_\lambda = \mathcal{Y}_\lambda = \{0,1\}^*$ and define R so that $R(x,y) = 1$ if $x = y$ and $R(x,y) = 0$ otherwise, we recover the definition of identity-based encryption (IBE). If we set $\mathcal{X}_\lambda = \{0,1\}^{n(\lambda)}$ and $\mathcal{Y}_\lambda$ to be the set of all circuits with input space $\{0,1\}^{n(\lambda)}$ and depth at most $d(\lambda)$, where n and d are some polynomials, and define R so that $R(x,y) = y(x)$, we recover the definition of ABE for circuits.

Functional encryption.

Definition 2.14 (Secret-Key Functional Encryption). *An SKFE scheme* SKFE *is a tuple of four PPT algorithms* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$*. Below, let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{F}$ be the plaintext, output, and function spaces* SKFE*, respectively.*

$\mathsf{Setup}(1^\lambda) \to \mathsf{msk}$**:** *The setup algorithm takes a security parameter 1^λ, and outputs a master secret key* msk*.*

$\mathsf{KG}(\mathsf{msk}, f) \to \mathsf{sk}_f$**:** *The key generation algorithm takes a master secret key* msk *and a function $f \in \mathcal{F}$, and outputs a functional decryption key sk_f.*

$\mathsf{Enc}(\mathsf{msk}, x) \to \mathsf{ct}$**:** *The encryption algorithm takes a master secret key* msk *and a plaintext $x \in \mathcal{X}$, and outputs a ciphertext* ct*.*

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y$**:** *The decryption algorithm takes a functional decryption key sk_f and a ciphertext* ct*, and outputs $y \in \{\bot\} \cup \mathcal{Y}$.*

Correctness: *We require that for every $x \in \mathcal{X}$, $f \in \mathcal{F}$, $q \in \mathbb{N}$, we have that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) = f(x) \;\middle|\; \begin{array}{l} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{msk}, f), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, x) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 2.15 (Function Privacy). *We formalize the experiment $\mathsf{Exp}^{\mathsf{full\text{-}fp}}_{\mathcal{A},\mathsf{SKFE}}(1^\lambda, \mathsf{coin})$ between an adversary $\mathcal{A}$ and a challenger for SKFE scheme for $\mathcal{X}, \mathcal{Y}$, and $\mathcal{F}$ as follows:*

1. *At the beginning, the challenger runs $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$. Throughout the experiment, $\mathcal{A}$ can access the following oracles.*

   $O_{\mathsf{Enc}}(x_0, x_1)$**:** *Given (x_0, x_1), it returns $\mathsf{Enc}(\mathsf{msk}, x_{\mathsf{coin}})$.*

   $O_{\mathsf{KG}}(f_0, f_1)$**:** *Given (f_0, f_1), it returns $\mathsf{KG}(\mathsf{msk}, f_{\mathsf{coin}})$.*

2. *If the following happens during the oracle queries above, the experiment aborts:* $f_0(x_0) \neq f_1(x_1)$ *or* $|x_0| \neq |x_1|$ *or* $|f_0| \neq |f_1|$.

3. $\mathcal{A}$ *outputs a guess* coin' *for* coin. *The challenger outputs* coin' *as the final output of the experiment.*

We say that SKFE *is fully function private if, for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{full\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{full\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(1^\lambda, 0) \to 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{full\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(1^\lambda, 1) \to 1\right]\right| \leq \mathsf{negl}(\lambda).$$

If $\mathcal{A}$ *can access* O_{Enc} *only once in* $\mathsf{Exp}^{\mathsf{full\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}$*, we say that* SKFE *is adaptively single-ciphertext function private.*

Theorem 2.16 ([GVW12, BS18, ABSV15, AV19]). *If there exist OWFs, there exists adaptively single-ciphertext function private SKFE for* P/poly.

Definition 2.17 (Public-Key Functional Encryption). *A PKFE scheme* PKFE *is a tuple of four PPT algorithms* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. *Below, let* $\mathcal{X}$, $\mathcal{Y}$, *and* $\mathcal{F}$ *be the plaintext, output, and function spaces of* PKFE, *respectively.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{msk})$**:** *The setup algorithm takes a security parameter* 1^λ *and outputs a public key* pk *and master secret key* msk.

$\mathsf{KG}(\mathsf{msk}, f) \to \mathsf{sk}_f$**:** *The key generation algorithm* KG *takes a master secret key* msk *and a function* $f \in \mathcal{F}$*, and outputs a functional decryption key* sk_f.

$\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$**:** *The encryption algorithm takes a public key* pk *and a message* $x \in \mathcal{X}$*, and outputs a ciphertext* ct.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y$**:** *The decryption algorithm takes a functional decryption key* sk_f *and a ciphertext* ct*, and outputs* $y \in \{\bot\} \cup \mathcal{Y}$.

Correctness: *We require we have that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) = f(x) \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda),\\ \mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{msk}, f),\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 2.18 (Adaptive Security for PKFE). *We formalize the experiment* $\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger for PKFE scheme for* $\mathcal{X}$, $\mathcal{Y}$, *and* $\mathcal{F}$ *as follows:*

1. *The challenger runs* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$.

2. $\mathcal{A}$ *sends arbitrary key queries. That is,* $\mathcal{A}$ *sends function* $f_i \in \mathcal{F}$ *to the challenger and the challenger responds with* $\mathsf{sk}_{f_i} \leftarrow \mathsf{KG}(\mathsf{msk}, f_i)$ *for the i-th query* f_i.

3. *At some point,* $\mathcal{A}$ *sends* (x_0, x_1) *to the challenger. If* $f_i(x_0) = f_i(x_1)$ *for all i, the challenger generates a ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x_{\mathsf{coin}})$. *The challenger sends* ct^* *to* $\mathcal{A}$.

4. *Again,* $\mathcal{A}$ *can sends function queries* f_i *such that* $f_i(x_0) = f_i(x_1)$.

5. $\mathcal{A}$ *outputs a guess* coin' *for* coin.

6. *The experiment outputs* coin'.

We say that PKFE *is adaptively secure if, for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ada\text{-}ind}}_{\mathsf{PKFE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\mathsf{PKFE},\mathcal{A}}(1^\lambda, 0) \to 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\mathsf{PKFE},\mathcal{A}}(1^\lambda, 1) \to 1\right]\right| \leq \mathsf{negl}(\lambda).$$

If $\mathcal{A}$ *can send only* q *key queries in* $\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\mathsf{PKFE},\mathcal{A}}$ *where* q *is a bounded polynomial, we say that* PKFE *is* q*-bounded adaptively secure.*

Theorem 2.19 ([GVW12, AV19]). *If there exists IND-CPA secure PKE, there exists q-bounded adaptively secure PKFE for* P/poly.

Remark 2.20. We defined FE as key-policy FE (KPFE) here. There is another type of FE called ciphertext-policy FE (CPFE). Since we use CPFE only as a building block of the CoIC-KLA secure PKE scheme in Section 4, we defer its definition to Section 4.1.

2.2 Useful Lemmata

The following lemma is taken verbatim from [BZ13, Lemma 2.1].

Lemma 2.21 ([BZ13, Lemma 2.1]). *Let $\mathcal{A}$ be a quantum algorithm, and let $\Pr[x]$ be the probability that $\mathcal{A}$ outputs x. Let $\mathcal{A}'$ be another quantum algorithm obtained from $\mathcal{A}$ by pausing $\mathcal{A}$ at an arbitrary stage of execution, performing a partial measurement that obtains one of k outcomes, and then resuming $\mathcal{A}$. Let $\Pr'[x]$ be the probability $\mathcal{A}'$ outputs x. Then $\Pr'[x] \geq \Pr[x]/k$.*

We will also need the quantum Goldreich-Levin lemma established by [CLLZ21] based on [AC02].

Lemma 2.22 (Quantum Goldreich-Levin with Quantum Auxiliary Input [CLLZ21, Lemma B.12]). *There exists a QPT algorithm $\mathcal{E}xt$ that satisfies the following. Let $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $\epsilon \in [0,1/2]$, and $\mathcal{A}$ be a quantum algorithm with a quantum auxiliary input aux such that*

$$\Pr\left[\mathcal{A}(aux, r) \to x \cdot r \mid r \leftarrow \{0,1\}^n\right] \geq \frac{1}{2} + \epsilon.$$

Then, we have

$$\Pr\left[\mathcal{E}xt([\mathcal{A}], aux) \to x\right] \geq 4\epsilon^2.$$

where $[\mathcal{A}]$ means the description of $\mathcal{A}$.

3 Public Key Encryption with Secure Key Leasing

In this section, we define the notion of public key encryption with secure key leasing (PKE-SKL) and its various security notions. Then we show several general relationships among those security notions.

3.1 Definitions

The syntax of PKE-SKL is defined as follows.

Definition 3.1 (PKE with Secure Key Leasing). *A PKE-SKL scheme SKL is a tuple of four algorithms $(\mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy)$. Below, let $\mathcal{X}$ be the message space of SKL.*

$\mathcal{KG}(1^\lambda) \to (\mathsf{ek}, d\mathsf{k}, \mathsf{vk})$**:** *The key generation algorithm takes a security parameter 1^λ, and outputs an encryption key ek, a decryption key $d\mathsf{k}$, and a verification key vk.*

$\mathsf{Enc}(\mathsf{ek}, \mathsf{m}) \to \mathsf{ct}$**:** *The encryption algorithm takes an encryption key ek and a message $\mathsf{m} \in \mathcal{X}$, and outputs a ciphertext ct.*

$\mathcal{D}ec(d\mathsf{k}, \mathsf{ct}) \to \tilde{\mathsf{m}}$**:** *The decryption algorithm takes a decryption key $d\mathsf{k}$ and a ciphertext ct, and outputs a value $\tilde{\mathsf{m}}$.*

$\mathcal{V}rfy(\mathsf{vk}, \widetilde{d\mathsf{k}}) \to \top/\bot$**:** *The verification algorithm takes a verification key vk and a (possibly malformed) decryption key $\widetilde{d\mathsf{k}}$, and outputs $\top$ or $\bot$.*

Decryption correctness: *For every* $\mathsf{m} \in \mathcal{X}$*, we have*

$$\Pr\left[\mathcal{D}ec(\mathit{dk}, \mathsf{ct}) = \mathsf{m} \;\middle|\; \begin{array}{l}(\mathsf{ek}, \mathit{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m})\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Verification correctness: *We have*

$$\Pr\left[\mathcal{V}rfy(\mathsf{vk}, \mathit{dk}) = \top \;\middle|\; (\mathsf{ek}, \mathit{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)\right] = 1 - \mathsf{negl}(\lambda).$$

Remark 3.2. We can assume without loss of generality that a decryption key of a PKE-SKL scheme is reusable, i.e., it can be reused to decrypt (polynomially) many ciphertexts. In particular, we can asusme that for honestly generated ct and dk, if we decrypt ct by using dk, the state of the decryption key after the decryption is negligibly close to that before the decryption in terms of trace distance. This is because the output of the decryption is almost deterministic by decryption correctness, and thus such an operation can be done without almost disturbing the input state by the gentle measurement lemma [Win99]. A similar remark applies to all variants of PKE-SKL (IBE, ABE, and FE with SKL) defined in this paper.

Remark 3.3. Though we are the first to define PKE with secure key leasing, SKFE with secure key leasing was already defined by Kitagawa and Nishimaki [KN22a]. The above definition is a natural adaptation of their definition with the important difference that we do not require classical certificate of deletion.

We define several security notions for PKE-SKL. The first is a natural indistinguishability security definition, which is our primary taget.

Definition 3.4 (IND-KLA Security). *We say that a PKE-SKL scheme* SKL *with the message space* $\mathcal{X}$ *is IND-KLA secure, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

1. $\mathcal{C}$ *runs* $(\mathsf{ek}, \mathit{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$ *and sends* ek *and* dk *to* $\mathcal{A}$*.*
2. *Throughout the experiment,* $\mathcal{A}$ *can access the following (stateful) verification oracle* $O_{\mathcal{V}rfy}$ *where* V *is initialized to be* $\bot$*:*

   $O_{\mathcal{V}rfy}(\widetilde{\mathit{dk}})$**:** *It runs* $d \leftarrow \mathsf{Vrfy}(\mathsf{vk}, \widetilde{\mathit{dk}})$ *and returns* d*. If* $V = \bot$ *and* $d = \top$*, it updates* $V := \top$*.*
3. $\mathcal{A}$ *sends* $(\mathsf{m}^*_0, \mathsf{m}^*_1) \in \mathcal{X}^2$ *to* $\mathcal{C}$*. If* $V = \bot$*,* $\mathcal{C}$ *output* 0 *as the final output of this experiment. Otherwise,* $\mathcal{C}$ *generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}^*_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* coin' *for* coin*.* $\mathcal{C}$ *outputs* coin' *as the final output of the experiment.*

For any QPT $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda, 0) \to 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda, 1) \to 1\right]\right| \leq \mathsf{negl}(\lambda).$$

We say that SKL *is 1-query IND-KLA secure if the above holds for any QPT* $\mathcal{A}$ *that makes at most one query to* $O_{\mathcal{V}rfy}$*.*

Remark 3.5. When we consider a 1-query adversary, we can assume that its query is made before receiving the challenge ciphertext ct^* without loss of generality. This is because otherwise the experiment always outputs 0.

Remark 3.6. By a standard hybrid argument, one can show that IND-KLA security implies multi-challenge IND-KLA security where the adversary is allowed to request arbitrarily many challenge ciphertexts. Thus, if we have an IND-KLA secure PKE-SKL scheme for single-bit messages, we can extend the plaintext length to an arbitrary polynomial by bit-by-bit encryption.

We also define the one-way variant of the above security.

Definition 3.7 (OW-KLA Security). *We say that a PKE-SKL scheme* SKL *with the message space* $\mathcal{X}$ *is OW-KLA secure, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ow\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

1. $\mathcal{C}$ *runs* $(\mathsf{ek}, d\mathsf{k}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$ *and sends* ek *and* $d\mathsf{k}$ *to* $\mathcal{A}$*.*
2. *Throughout the experiment,* $\mathcal{A}$ *can access the following (stateful) verification oracle* $O_{\mathcal{V}rfy}$ *where* V *is initialized to be* $\bot$*:*

   $O_{\mathcal{V}rfy}(\widetilde{d\mathsf{k}})$**:** *It runs* $d \leftarrow \mathsf{Vrfy}(\mathsf{vk}, \widetilde{d\mathsf{k}})$ *and returns* d*. If* $V = \bot$ *and* $d = \top$*, it updates* $V := \top$*.*
3. $\mathcal{A}$ *sends* $\mathsf{RequestChallenge}$ *to* $\mathcal{C}$*. If* $V = \bot$*,* $\mathcal{C}$ *outputs* 0 *as the final output of this experiment. Otherwise,* $\mathcal{C}$ *chooses* $\mathsf{m}^* \leftarrow \mathcal{X}$*, generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}^*)$ *and sends* ct^* *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* m*.* $\mathcal{C}$ *outputs* 1 *if* $\mathsf{m} = \mathsf{m}^*$ *and otherwise outputs* 0 *as the final output of the experiment.*

For any QPT $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ow\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{ow\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda) \to 1\right] \leq \mathsf{negl}(\lambda).$$

We say that SKL *is 1-query OW-KLA secure if the above holds for any QPT* $\mathcal{A}$ *that makes at most one query to* $O_{\mathcal{V}rfy}$*.*

Similar to normal PKE, IND-KLA security implies OW-KLA security if $|\mathcal{X}|$ is super-polynomial in λ.

Finally, we define a security notion which we call one-more unreturnability (OMUR), which requires that an adversary given a single copy of the decryption key cannot pass the verification more than once. Though this does not seem very meaningful by itself, this is a useful intermediate tool for our final goal of constructing IND-KLA secure scheme.

Definition 3.8 (One-More Unreturnability). *We say that a PKE-SKL scheme* SKL *with the message space* $\mathcal{X}$ *satisfies One-More UnReturnability (OMUR), if it satisfies the following requirement, formalized from the experiment* $\mathsf{Expt}^{\mathsf{omur}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

1. $\mathcal{C}$ *runs* $(\mathsf{ek}, d\mathsf{k}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$ *and sends* ek *and* $d\mathsf{k}$ *to* $\mathcal{A}$*.*
2. *Throughout the experiment,* $\mathcal{A}$ *can access the following (stateful) verification oracle* $O_{\mathcal{V}rfy}$ *where* count *is initialized to be* 0*:*

   $O_{\mathcal{V}rfy}(\widetilde{d\mathsf{k}})$**:** *It runs* $d \leftarrow \mathsf{Vrfy}(\mathsf{vk}, \widetilde{d\mathsf{k}})$ *and returns* d*. It updates* $\mathsf{count} := \mathsf{count} + 1$ *if* $d = \top$*.*
3. $\mathcal{A}$ *sends* Finish *to* $\mathcal{C}$*. If* $\mathsf{count} \geq 2$*,* $\mathcal{C}$ *outputs* 1 *and* 0 *otherwise as the final output of this experiment.*

For any QPT $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{omur}}_{\mathsf{SKL},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Expt}^{\mathsf{omur}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda) \to 1\right] \leq \mathsf{negl}(\lambda).$$

3.2 Relationships among Security Notions

We show several relationships among different security notions for PKE-SKL. In particular, we show the following theorem.

Theorem 3.9. *If there exists a 1-query OW-KLA secure PKE-SKL scheme, there exists an IND-KLA secure PKE-SKL scheme.*

This theorem simplifies our task: For constructing a (poly-query) IND-KLA secure scheme, it suffices to construct a 1-query OW-KLA secure scheme. We construct a 1-query OW-KLA secure scheme in Section 5.

We prove Theorem 3.9 in the following three steps.

1. Give a conversion to add OMUR to any 1-query OW-KLA secure scheme (Lemma 3.10).
2. Convert a 1-query OW-KLA secure scheme that satisfies OMUR to a 1-query IND-KLA secure scheme that satisfies OMUR (Lemma 3.12).
3. Show that any 1-query IND-KLA secure scheme that satisfies OMUR is IND-KLA secure (Lemma 3.14).

It is clear that Theorem 3.9 follows from Lemmata 3.10, 3.12 and 3.14. We prove them in the following.

Lemma 3.10. *If there exists a 1-query OW-KLA secure PKE-SKL scheme, then there exists a 1-query OW-KLA secure PKE-SKL scheme that satisfies OMUR.*

Remark 3.11. This lemma is actually not needed for the purpose of this paper since our construction of a 1-query OW-KLA secure PKE-SKL scheme in Section 5 already satisfies OMUR as mentioned in Remark 5.7. We include this lemma in the paper because this general reduction may be useful in future works.

Proof of Lemma 3.10. Let $\mathsf{OW} = (\mathsf{OW}.\mathcal{KG}, \mathsf{OW}.\mathsf{Enc}, \mathsf{OW}.\mathcal{D}ec, \mathsf{OW}.\mathcal{V}rfy)$ be a 1-query OW-KLA secure PKE-SKL scheme with the message space $\mathcal{X}$. We assume that a decryption key of OW is reusable in the sense of Remark 3.2 and vk contains ek without loss of generality. Then we consider a modified PKE-SKL scheme $\mathsf{OW}' = (\mathsf{OW}'.\mathcal{KG}, \mathsf{OW}'.\mathsf{Enc}, \mathsf{OW}'.\mathcal{D}ec, \mathsf{OW}'.\mathcal{V}rfy)$ with the same message space $\mathcal{X}$ defined as follows. The algorithms $\mathsf{OW}'.\mathcal{KG}$, $\mathsf{OW}'.\mathsf{Enc}$, and $\mathsf{OW}'.\mathcal{D}ec$ are identical to $\mathsf{OW}.\mathcal{KG}$, $\mathsf{OW}.\mathsf{Enc}$, and $\mathsf{OW}.\mathcal{D}ec$, respectively. The algorithm $\mathsf{OW}'.\mathcal{V}rfy$ works as follows:

$\mathsf{OW}'.\mathcal{V}rfy(\mathsf{vk}, \widetilde{dk})$**:** On input a verification key vk and a (possibly malformed) decryption key $\widetilde{dk}$, do the following:

- **Decryptability verification:** Choose $\mathsf{m} \leftarrow \mathcal{X}$ and run $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m})$ and $\mathsf{m}' \leftarrow \mathsf{Dec}(\widetilde{dk}, \mathsf{ct})$. If $\mathsf{m}' \neq \mathsf{m}$, return $\bot$.
- **Original verification:** Otherwise, let $\widetilde{dk}'$ be the state of the decryption key after running the decryption algorithm. Run $\mathsf{OW}.\mathcal{V}rfy(\mathsf{vk}, \widetilde{dk}')$ and return whatever $\mathsf{OW}.\mathcal{V}rfy$ returns.

Correctness. The decryption correctness of OW' follows from that of OW because the only difference between these schemes is the verification algorithm, which is irrelevant to the decryption correctness. The verification correctness of OW' follows from that of OW because we assume that OW has reusable decryption keys and thus $\widetilde{dk}'$ in $\mathsf{OW}'.\mathcal{V}rfy$ has a negligible trace distance from $\widetilde{dk}$, which passes $\mathsf{OW}.\mathcal{V}rfy$ except for a negligible probability by the verification correctness of OW.

1-query OW-SKL security. The 1-query OW-SKL security of OW' follows from that of OW by a straightforward reduction. Specifically, let $\mathcal{A}$ be an QPT adversary that breaks the 1-query OW-SKL security of OW'. Then, we construct a QPT adversary $\mathcal{B}$ that breaks the 1-query OW-SKL security of OW as follows:

$\mathcal{B}(\mathsf{ek}, dk)$**:** Run $\mathcal{A}(\mathsf{ek}, dk)$ until $\mathcal{A}$ makes a verification query $\widetilde{dk}$. For simulating the verification oracle to $\mathcal{A}$, choose $\mathsf{m} \leftarrow \mathcal{X}$, run $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m})$ and $\mathsf{m}' \leftarrow \mathsf{Dec}(\widetilde{dk}, \mathsf{ct})$, and let $\widetilde{dk}'$ be the state of the decryption key after running the decryption algorithm. If $\mathsf{m}' \neq \mathsf{m}$, output 0 and immediately halt. Otherwise, query $\widetilde{dk}'$ to its own verification oracle, and forward the response to $\mathcal{A}$. When $\mathcal{A}$ sends $\mathsf{RequestChallenge}$, forward it to the external challenger to receive ct^* and forward it to $\mathcal{A}$. Run $\mathcal{A}$ until it halts and output whatever $\mathcal{A}$ outputs.

We can see that the experiment which $\mathcal{B}$ plays outputs 1 if and only if the (simulated) experiment which $\mathcal{A}$ plays outputs 1. Therefore, $\mathcal{B}$ breaks the 1-query OW-SKL security of OW. Thus, the 1-query OW-SKL security of OW' follows from that of OW.

OMUR. In the following, we show that OW' satisfies OMUR. Let $\mathcal{A}$ be a QPT adversary against the OMUR of OW' that makes $Q = \mathrm{poly}(\lambda)$ verification queries. Then we consider the following sequence of hybrids.

Hyb_0**:** This is identical to the experiment $\mathsf{Expt}^{\mathsf{omur}}_{\mathsf{OW}',\mathcal{A}}(1^\lambda)$ as defined in Definition 3.8.

Note that we have

$$\Pr[\mathsf{Hyb}_0 = 1] = \mathsf{Adv}^{\mathsf{omur}}_{\mathsf{OW}',\mathcal{A}}(\lambda).$$

Hyb_1**:** This is identical to Hyb_0 except that the challenger uniformly chooses integers $1 \leq i_1 < i_2 \leq Q$ at the beginning of the experiment and outputs 1 if and only if i_1-th and i_2-th verification queries are the first two queries to which the verification oracle returned $\top$.

Whenever Hyb_0 returns 1, there are at least 2 verification queries accepted by the verification oracle. Therefore, when we uniformly choose $1 \leq i_1 < i_2 \leq Q$, the probability that i_1-th and i_2-th queries are the first two queries to be accepted is $\binom{Q}{2}^{-1} = \frac{2}{Q(Q-1)}$. Therefore we have

$$\Pr[\mathsf{Hyb}_1 = 1] = \frac{2}{Q(Q-1)} \Pr[\mathsf{Hyb}_0 = 1].$$

Hyb_2**:** This is identical to Hyb_1 except that the verification oracle just returns $\bot$ without running the verification algorithm to i-th query for all $i \in [i_2 - 1] \setminus \{i_1\}$ and the experiment halts right after running the verification oracle for the i_2-th query where it outputs 1 if and only if the verification oracle returned $\top$ to both i_1-th and i_2-th queries.

When Hyb_1 returns 1, the verification oracle returns $\bot$ to i-th query for all $i \in [i_2 - 1] \setminus \{i_1\}$ since otherwise i_1-th and i_2-th queries cannot be the first 2 queries to be accepted. Therefore, these hybrids are identical until $\mathcal{A}$ makes i_2-th query when Hyb_1 returns 1.[9] Moreover, Hyb_2 outputs 1 whenever Hyb_1 outputs 1 if we run the rest of $\mathcal{A}$ to complete Hyb_1. Therefore, we have

$$\Pr[\mathsf{Hyb}_2 = 1] \geq \Pr[\mathsf{Hyb}_1 = 1].$$

Hyb_3**:** This is identical to Hyb_2 except that the experiment outputs 1 if and only if i_1-th query is accepted and i_2-th query passes the "Decryptability verification" part of $\mathsf{OW}'.\mathcal{V}rfy$, i.e., $\mathsf{m} = \mathsf{m}'$ in the notation of the description of $\mathsf{OW}'.\mathcal{V}rfy$.

Since the condition to output 1 is just relaxed, we have

$$\Pr[\mathsf{Hyb}_3 = 1] \geq \Pr[\mathsf{Hyb}_2 = 1].$$

Below, we prove

$$\Pr[\mathsf{Hyb}_3 = 1] = \mathsf{negl}(\lambda).$$

To prove this, we consider the following QPT adversary $\mathcal{B}$ against the 1-query OW-SKL security of OW that works as follows:

$\mathcal{B}(\mathsf{ek}, \mathit{dk})$**:** Uniformly choose integers $1 \leq i_1 < i_2 \leq Q$ and run $\mathcal{A}(\mathsf{ek}, \mathit{dk})$ until it makes i_2-th query where the response by the verification oracle to $\mathcal{A}$'s i-th query for $i \in [i_2 - 1]$ is simulated as follows: If $i \neq i_1$, return $\bot$ as the response from the verification oracle. If $i = i_1$, forward the query to its own verification oracle and forward the response to $\mathcal{A}$. Let dk_{i_2} be $\mathcal{A}$'s i_2-th verification query. Send $\mathsf{RequestChallenge}$ to the external challenger to receive ct^*. Run $\mathsf{m}' \leftarrow \mathsf{OW.Dec}(\mathsf{ct}^*, \mathit{dk})$ and output m'.

[9] Note that there is a superficial difference that the verification oracle of Hyb_1 runs the verification algorithm to i-th query for all $i \in [i_2 - 1] \setminus \{i_1\}$ in Hyb_1 but it does not in Hyb_2. However since these query registers are not used at all for generating the output of Hyb_2, the difference of if measurements are applied on them cannot affect the probability to output 1.

By the definitions of Hyb_3 and $\mathcal{B}$, we can see that

$$\mathsf{Adv}^{\mathsf{ow\text{-}kla}}_{\mathsf{OW},\mathcal{B}}(\lambda) = \Pr[\mathsf{Hyb}_3 = 1].$$

Thus, we have $\Pr[\mathsf{Hyb}_3 = 1] = \mathsf{negl}(\lambda)$ by the 1-query OW-SKL security of OW.

Combining the above, we have $\mathsf{Adv}^{\mathsf{omur}}_{\mathsf{OW}',\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$, which means that OW satisfies OMUR. This completes the proof of Lemma 3.10. □

Lemma 3.12. *If there exists a 1-query OW-KLA secure PKE-SKL scheme, then there exists a 1-query IND-KLA secure PKE-SKL scheme. Moreover, if the base scheme satisfies OMUR, then then the resulting scheme satisfies OMUR.*

Proof. Let $\mathsf{OW} = (\mathsf{OW}.\mathcal{KG}, \mathsf{OW.Enc}, \mathsf{OW}.\mathcal{Dec}, \mathsf{OW}.\mathcal{Vrfy})$ be a 1-query OW-KLA secure PKE-SKL scheme with the message space $\{0,1\}^n$ that satisfies OMUR. Then, we construct an IND-KLA secure PKE-SKL scheme $\mathsf{IND} = (\mathsf{IND}.\mathcal{KG}, \mathsf{IND.Enc}, \mathsf{IND}.\mathcal{Dec}, \mathsf{IND}.\mathcal{Vrfy})$ with the message space $\{0,1\}$ as follows.

$\mathsf{IND}.\mathcal{KG}(1^\lambda) \to (\mathsf{ek}, d\mathsf{k}, \mathsf{vk})$**:** On input the security parameter 1^λ, run $(\mathsf{ek}, d\mathsf{k}, \mathsf{vk}) \leftarrow \mathsf{OW}.\mathcal{KG}(1^\lambda)$ and output $(\mathsf{ek}, d\mathsf{k}, \mathsf{vk})$.

$\mathsf{IND.Enc}(\mathsf{ek}, \mathsf{m}) \to \mathsf{IND.ct}$**:** On input an encryption key ek and a message $\mathsf{m} \in \{0,1\}$, choose $r, x \leftarrow \{0,1\}^n$, generate $\mathsf{OW.ct} \leftarrow \mathsf{OW.Enc}(\mathsf{ek}, x)$, set $b := (x \cdot r) \oplus \mathsf{m}$, and output a ciphertext $\mathsf{IND.ct} := (\mathsf{OW.ct}, r, b)$.

$\mathsf{IND}.\mathcal{Dec}(d\mathsf{k}, \mathsf{IND.ct}) \to \tilde{\mathsf{m}}$**:** On input a decryption key $d\mathsf{k}$ and a ciphertext $\mathsf{IND.ct} = (\mathsf{OW.ct}, r, b)$, compute $\tilde{x} \leftarrow \mathsf{OW}.\mathcal{Dec}(d\mathsf{k}, \mathsf{OW.ct})$ and output $\tilde{\mathsf{m}} := (\tilde{x} \cdot r) \oplus b$.

$\mathsf{IND}.\mathcal{Vrfy}(\mathsf{vk}, \widetilde{d\mathsf{k}}) \to \top/\bot$**:** On input a verification key vk and a (possibly malformed) decryption key $\widetilde{d\mathsf{k}}$, run $\mathsf{OW}.\mathcal{Vrfy}(\mathsf{vk}, \widetilde{d\mathsf{k}})$ and output whatever $\mathsf{OW}.\mathcal{Vrfy}$ outputs.

The decryption correctness and verification correctness of IND immediately follow from those of OW. The OMUR of IND immediately follows from that of OW since their key generation and verification algorithms are identical and the definition of OMUR only depends on these algorithms. In the following, we prove that IND is IND-KLA secure assuming that OW is OW-KLA secure. Toward contradiction, suppose that IND is not IND-KLA secure. Then, there is a QPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{ind\text{-}kla}}_{\mathsf{IND},\mathcal{A}}(\lambda)$ is non-negligible. Without loss of generality, we assume that

$$\Pr_{\mathsf{coin}\leftarrow\{0,1\}}[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{IND},\mathcal{A}}(1^\lambda, \mathsf{coin}) \to \mathsf{coin}] \geq 1/2 + \epsilon(\lambda) \tag{1}$$

for a non-negligible $\epsilon(\lambda)$. Since IND is a bit encryption, we assume that the challenge message pair $(\mathsf{m}_0, \mathsf{m}_1)$ is $(0,1)$ without loss of generality. We divide $\mathcal{A}$ into the following two stages $\mathcal{A}_0$ and $\mathcal{A}_1$:

$\mathcal{A}_0^{O_{\mathcal{V}rfy}}(\mathsf{ek}, d\mathsf{k}) \to st_{\mathcal{A}}$**:** Upon receiving $(\mathsf{ek}, d\mathsf{k})$ from $\mathcal{C}$, makes a single query to $O_{\mathcal{V}rfy}$ and outputs a quantum state $st_{\mathcal{A}}$.

$\mathcal{A}_1(st_{\mathcal{A}}, \mathsf{IND.ct}) \to \mathsf{coin}'$**:** Upon receiving the state $st_{\mathcal{A}}$ from $\mathcal{A}_0$ and $\mathsf{IND.ct} = (\mathsf{OW.ct}, r, b)$ from $\mathcal{C}$, output coin'.

We remark that we can assume that $\mathcal{A}_1$ does not make any query to $O_{\mathcal{V}rfy}$ without loss of generality by Remark 3.5.

We have

$$\begin{aligned}
&\Pr_{\mathsf{coin}\leftarrow\{0,1\}}[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{IND},\mathcal{A}}(1^\lambda, \mathsf{coin}) \to \mathsf{coin}] \\
&= \Pr_{\mathsf{coin}\leftarrow\{0,1\}}[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{IND},\mathcal{A}}(1^\lambda, \mathsf{coin}) \to \mathsf{coin} \wedge V = \top] \\
&+ \Pr_{\mathsf{coin}\leftarrow\{0,1\}}[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{IND},\mathcal{A}}(1^\lambda, \mathsf{coin}) \to \mathsf{coin} \wedge V = \bot] \\
&= \Pr[V = \top] \cdot \Pr_{\mathsf{coin}\leftarrow\{0,1\}}[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{IND},\mathcal{A}}(1^\lambda, \mathsf{coin}) \to \mathsf{coin} \mid V = \top] \\
&+ \frac{1}{2}(1 - \Pr[V = \top]).
\end{aligned} \tag{2}$$

By Equations (1) and (2), we have[10]

$$\Pr_{\mathsf{coin}\leftarrow\{0,1\}}[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{IND},\mathcal{A}}(1^\lambda,\mathsf{coin})\to\mathsf{coin}\mid V=\top]\geq\frac{1}{2}+\frac{\epsilon(\lambda)}{\Pr[V=\top]}. \tag{3}$$

Then, we construct an adversary $\mathcal{B}=(\mathcal{B}_0,\mathcal{B}_1)$ against OW-KLA security of OW that works as follows.

$\mathcal{B}_0^{O_{\mathcal{V}rfy}}(\mathsf{ek},d\mathcal{K})\to st_{\mathcal{A}}$**:** This is identical to $\mathcal{A}_0$. Specifically, run $st_{\mathcal{A}}\leftarrow\mathcal{A}_0^{O_{\mathcal{V}rfy}}(\mathsf{ek},d\mathcal{K})$ and output $st_{\mathcal{A}}$.

$\mathcal{B}_1(st_{\mathcal{A}},\mathsf{OW.ct})\to x$**:** Upon receiving $st_{\mathcal{A}}$ from $\mathcal{B}_0$, send RequestChallenge to $\mathcal{C}$ and receive OW.ct from $\mathcal{C}$. Then set $aux:=(st_{\mathcal{A}},\mathsf{OW.ct})$ and define an algorithm $\mathcal{A}'$ as follows.

$\mathcal{A}'(aux,r)$**:** On input $aux=(st_{\mathcal{A}},\mathsf{OW.ct})$ and $r\in\{0,1\}^n$, choose $b\leftarrow\{0,1\}$, set $\mathsf{IND.ct}=(\mathsf{OW.ct},r,b)$, run $\mathsf{coin}'\leftarrow\mathcal{A}_1(st_{\mathcal{A}},\mathsf{IND.ct})$, and output $\mathsf{coin}'\oplus b$.

Run $x\leftarrow\mathcal{E}xt([\mathcal{A}'],aux)$, and output x where $\mathcal{E}xt$ is the algorithm as in Lemma 2.22 and $[\mathcal{A}']$ is the description of $\mathcal{A}'$.

In the following, we show that $\mathcal{B}$ breaks OW-KLA security of OW. Let $\mathcal{G}$ be an algorithm that works as follows.

$\mathcal{G}(1^\lambda)$**:** Generate $(\mathsf{ek},d\mathcal{K},\mathsf{vk})\leftarrow\mathsf{OW}.\mathcal{KG}(1^\lambda)$, $st_{\mathcal{A}}\leftarrow\mathcal{A}_0^{O_{\mathcal{V}rfy}}(\mathsf{ek},d\mathcal{K})$, $x\leftarrow\{0,1\}^n$, and $\mathsf{OW.ct}\leftarrow\mathsf{OW.Enc}(\mathsf{ek},x)$. Let $V:=\top$ if the response to $\mathcal{A}_0$'s query (which is assumed to be made once) is $\top$ and $V:=\bot$ otherwise. Output $(V,st_{\mathcal{A}},\mathsf{OW.ct},x)$.

By Equation (3) and a standard averaging argument, for at least $\frac{\epsilon(\lambda)}{2\Pr[V=\top]}$-fraction of $(V,st_{\mathcal{A}},\mathsf{OW.ct},x)$ generated by $\mathcal{G}(1^\lambda)$ conditioned on $V=\top$, we have

$$\Pr\left[\mathcal{A}_1(st_{\mathcal{A}},\mathsf{IND.ct})\to\mathsf{coin}\right]\geq\frac{1}{2}+\frac{\epsilon(\lambda)}{2\Pr[V=\top]}\geq\frac{1}{2}+\frac{\epsilon(\lambda)}{2}$$

where $\mathsf{coin}\leftarrow\{0,1\}$, $r\leftarrow\{0,1\}^n$, $b:=(x\cdot r)\oplus\mathsf{coin}$, and $\mathsf{IND.ct}=(\mathsf{OW.ct},r,b)$.

Therefore, for at least $\frac{\epsilon(\lambda)}{2}$-fraction of $(V,st_{\mathcal{A}},\mathsf{OW.ct},x)$ generated by $\mathcal{G}(1^\lambda)$, we have

$$\Pr\left[V=\top\wedge\mathcal{A}_1(st_{\mathcal{A}},\mathsf{IND.ct})\to\mathsf{coin}\right]\geq\frac{1}{2}+\frac{\epsilon(\lambda)}{2} \tag{4}$$

where $\mathsf{coin}\leftarrow\{0,1\}$, $r\leftarrow\{0,1\}^n$, $b:=(x\cdot r)\oplus\mathsf{coin}$, and $\mathsf{IND.ct}=(\mathsf{OW.ct},r,b)$.

For such $(V,st_{\mathcal{A}},\mathsf{OW.ct},x)$, if we let $aux=(st_{\mathcal{A}},\mathsf{OW.ct})$, Equation (4) directly implies

$$\Pr_{r\leftarrow\{0,1\}^n}\left[\mathcal{A}'(aux,r)\to x\cdot r\right]\geq\frac{1}{2}+\frac{\epsilon(\lambda)}{2}.$$

Therefore, by Lemma 2.22, we have

$$\Pr\left[\mathcal{E}xt([\mathcal{A}'],aux)\to x\right]\geq\epsilon(\lambda)^2. \tag{5}$$

Since Equation (5) *and* $V=\top$ hold at the same time for at least $\frac{\epsilon(\lambda)}{2}$-fraction of $(V,st_{\mathcal{A}},\mathsf{OW.ct},x)$, we have

$$\Pr_{(\mathsf{v},st_{\mathcal{A}},\mathsf{OW.ct},x)\leftarrow\mathcal{G}(1^\lambda)}\left[V=\top\wedge\mathcal{B}_1(st_{\mathcal{A}},\mathsf{OW.ct})\to x\right]\geq\frac{\epsilon(\lambda)^3}{2}.$$

By the definitions of $\mathcal{B}=(\mathcal{B}_0,\mathcal{B}_1)$ and $\mathcal{G}$ and the assumption that $\epsilon(\lambda)$ is non-negligible, this implies that $\mathcal{B}$ breaks OW-KLA security of OW. □

[10]We can assume $\Pr[V=\top]\neq 0$ since otherwise Equation (1) cannot be satisfied.

Remark 3.13 (On Multiple-Query Case). In the above reduction, it is important that $\mathcal{A}_1$ can be assumed to not make any verification query because otherwise we cannot apply the quantum Goldreich-Levin theorem (Lemma 2.22). In the 1-query setting, this can be assumed without loss of generality by Remark 3.5. In the multiple-query setting, we cannot assume it in general. If we assume that the base scheme satisfies OMUR, we can assume it without loss of generality because post-challenge verification queries are useless for such schemes. However, we do not know how to resolve the issue in the multiple-query setting without relying on OMUR.

Lemma 3.14. *If a PKE-SKL scheme is 1-query IND-KLA secure and satisfies OMUR, then it is IND-KLA secure.*

Proof. Let $\mathsf{SKL} = (\mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$ be an IND-KLA secure PKE-SKL scheme that satisfies OMUR. For a QPT adversary $\mathcal{A}$ against IND-KLA security of SKL that makes $Q = \mathrm{poly}(\lambda)$ verification queries and $\mathsf{coin} \in \{0,1\}$, we consider the following sequence of hybrids.

$\mathsf{Hyb}_0^{\mathsf{coin}}$**:** This is identical to $\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda, \mathsf{coin})$.

Note that our goal is to prove

$$\left|\Pr\left[\mathsf{Hyb}_0^0 = 1\right] - \Pr\left[\mathsf{Hyb}_0^1 = 1\right]\right| = \mathsf{negl}(\lambda).$$

$\mathsf{Hyb}_1^{\mathsf{coin}}$**:** This is identical to $\mathsf{Hyb}_0^{\mathsf{coin}}$ except that the verification oracle returns $\bot$ to all queries made after it returns $\top$ once.

By the OMUR of SKL, we have

$$\left|\Pr\left[\mathsf{Hyb}_1^{\mathsf{coin}} \to 1\right] - \Pr\left[\mathsf{Hyb}_0^{\mathsf{coin}} \to 1\right]\right| = \mathsf{negl}(\lambda)$$

for $\mathsf{coin} \in \{0,1\}$.

$\mathsf{Hyb}_2^{\mathsf{coin}}$**:** This is identical to $\mathsf{Hyb}_1^{\mathsf{coin}}$ except that the challenger chooses $i^* \leftarrow [Q]$ at the beginning of the game, the verification oracle just returns $\bot$ without running the verification algorithm to i-th query for $i \neq i^*$, and the experiment returns 0 if the verification oracle returns $\bot$ to i^*-th query.

Note that there is exactly one verification query to be accepted in $\mathsf{Hyb}_1^{\mathsf{coin}}$ whenever it returns 1. If i^* is the correct guess for such query, which occurs with probability $\frac{1}{Q}$, then $\mathsf{Hyb}_2^{\mathsf{coin}}$ is identical to $\mathsf{Hyb}_1^{\mathsf{coin}}$.[11] Moreover, $\mathsf{Hyb}_2^{\mathsf{coin}}$ outputs 0 when the guess is incorrect. Therefore, we have

$$\Pr\left[\mathsf{Hyb}_2^{\mathsf{coin}} \to 1\right] = \frac{1}{Q}\Pr\left[\mathsf{Hyb}_1^{\mathsf{coin}} \to 1\right].$$

Below, we prove

$$\left|\Pr\left[\mathsf{Hyb}_2^0 = 1\right] - \Pr\left[\mathsf{Hyb}_2^1 = 1\right]\right| = \mathsf{negl}(\lambda).$$

To prove this, we consider a QPT adversary $\mathcal{B}$ against 1-query IND-KLA security of SKL that works as follows.

$\mathcal{B}(\mathsf{ek}, \mathit{dk})$**:** Choose $i^* \leftarrow [Q]$ and run $\mathcal{A}(\mathsf{ek}, \mathit{dk})$ where the i^*-th query is forwarded to its own verification oracle and responded according to the response from the oracle while all the other queries are responded by $\bot$. When $\mathcal{A}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*)$, forward it to the external challenger, receive ct^* from the challenger, and forward it to $\mathcal{A}$. Finally, output whatever $\mathcal{A}$ outputs.

By the definitions of $\mathcal{B}$ and $\mathsf{Hyb}_2^{\mathsf{coin}}$, one can see that

$$\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{B}}(1^\lambda, \mathsf{coin}) \to 1\right] = \Pr\left[\mathsf{Hyb}_2^{\mathsf{coin}} = 1\right]$$

[11] A similar remark to Footnote 9 applies here.

for $\mathsf{coin} \in \{0,1\}$. Therefore, we have

$$\begin{aligned}&\left|\Pr\left[\mathsf{Hyb}_2^0 = 1\right] - \Pr\left[\mathsf{Hyb}_2^1 = 1\right]\right| \\ &= \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{B}}(1^\lambda, 0) \to 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{B}}(1^\lambda, 1) \to 1\right]\right| \\ &= \mathsf{negl}(\lambda)\end{aligned}$$

by the 1-query IND-SKL security of SKL.

Combining the above, we have

$$\left|\Pr\left[\mathsf{Hyb}_0^0 = 1\right] - \Pr\left[\mathsf{Hyb}_0^1 = 1\right]\right| = \mathsf{negl}(\lambda).$$

This completes the proof of Lemma 3.14. □

4 Public Key Encryption with CoIC-KLA Security

In this section, we introduce a new security notion called CoIC-KLA security for PKE, and construct a PKE scheme that satisfies it based on any IND-CPA secure PKE scheme. Looking ahead, it is used as a building block of our construction of PKE-SKL in Section 5.

4.1 Tools

We first introduce some tools used in this section.

Measurement Implementation. We review some notions related to measurement implementations used in the definition and the security proof of CoIC-KLA security.

Definition 4.1 (Projective Implementation). *Let:*

- $\mathcal{D}$ *be a finite set of distributions over an index set* $\mathcal{I}$.
- $\mathcal{P} = \{\boldsymbol{P}_i\}_{i\in\mathcal{I}}$ *be a positive operator valued measure (POVM).*
- $\mathcal{E} = \{\boldsymbol{E}_D\}_{D\in\mathcal{D}}$ *be a projective measurement with index set* $\mathcal{D}$.

We consider the following measurement procedure.

1. *Measure under the projective measurement* $\mathcal{E}$ *and obtain a distribution* D.
2. *Output a random sample from the distribution* D.

We say $\mathcal{E}$ *is the projective implementation of* $\mathcal{P}$*, denoted by* $\mathsf{ProjImp}(\mathcal{P})$*, if the measurement process above is equivalent to* $\mathcal{P}$.

Theorem 4.2 ([Zha20, Lemma 1]). *Any binary outcome POVM* $\mathcal{P} = (\boldsymbol{P}, \boldsymbol{I} - \boldsymbol{P})$ *has a unique projective implementation* $\mathsf{ProjImp}(\mathcal{P})$.

Definition 4.3 (Shift Distance). *For two distributions* D_0, D_1*, the shift distance with parameter* ϵ*, denoted by* $\Delta^\epsilon_{\mathsf{Shift}}(D_0, D_1)$*, is the smallest quantity* δ *such that for all* $x \in \mathbb{R}$*:*

$$\begin{aligned}&\Pr[D_0 \le x] \le \Pr[D_1 \le x+\epsilon] + \delta, && \Pr[D_0 \ge x] \le \Pr[D_1 \ge x-\epsilon] + \delta, \\ &\Pr[D_1 \le x] \le \Pr[D_0 \le x+\epsilon] + \delta, && \Pr[D_1 \ge x] \le \Pr[D_0 \ge x-\epsilon] + \delta.\end{aligned}$$

For two real-valued measurements $\mathcal{M}$ *and* $\mathcal{N}$ *over the same quantum system, the shift distance between* $\mathcal{M}$ *and* $\mathcal{N}$ *with parameter* ϵ *is*

$$\Delta^\epsilon_{\mathsf{Shift}}(\mathcal{M},\mathcal{N}) := \sup_{|\psi\rangle} \Delta^\epsilon_{\mathsf{Shift}}(\mathcal{M}(|\psi\rangle), \mathcal{N}(|\psi\rangle)).$$

Definition 4.4 (Mixture of Projetive Measurement [Zha20]). *Let $D : \mathcal{R} \to \mathcal{I}$ where $\mathcal{R}$ and $\mathcal{I}$ are some sets. Let $\{(\boldsymbol{P}_i, \boldsymbol{Q}_i)\}_{i\in\mathcal{I}}$ be a collection of binary projective measurement. The mixture of projective measurements associated to $\mathcal{R}$, $\mathcal{I}$, D, and $\{(\boldsymbol{P}_i, \boldsymbol{Q}_i)\}_{i\in\mathcal{I}}$ is the binary POVM $\mathcal{P}_D = (\boldsymbol{P}_D, \boldsymbol{Q}_D)$ defined as follows.*

$$\boldsymbol{P}_D = \sum_{i\in\mathcal{I}} \Pr[i \leftarrow D(R)]\boldsymbol{P}_i \qquad\qquad \boldsymbol{Q}_D = \sum_{i\in\mathcal{I}} \Pr[i \leftarrow D(R)]\boldsymbol{Q}_i,$$

where R is uniformly distributed in $\mathcal{R}$.

Theorem 4.5 ([Zha20, KN22b]). *Let D be any probability distribution and $\mathcal{P} = \{(\Pi_i, \boldsymbol{I} - \Pi_i)\}_i$ be a collection of binary outcome projective measurements. For any $0 < \epsilon, \delta < 1$, there exists an algorithm of measurement $\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D}$ that satisfies the following.*

- $\Delta^{\epsilon}_{\mathsf{Shift}}(\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D}, \mathsf{ProjImp}(\mathcal{P}_D)) \leq \delta$.
- *$\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D}$ is (ϵ,δ)-almost projective in the following sense. For any quantum state $|\psi\rangle$, we apply $\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D}$ twice in a row to $|\psi\rangle$ and obtain measurement outcomes x and y, respectively. Then, $\Pr[|x - y| \leq \epsilon] \geq 1 - \delta$.*
- *$\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D}$ is (ϵ,δ)-reverse almost projective in the following sense. For any quantum state $|\psi\rangle$, we apply $\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D}$ and $\mathcal{API}^{\epsilon,\delta}_{\mathcal{P}^{\mathtt{rev}},D}$ in a row to $|\psi\rangle$ and obtain measurement outcomes x and y, respectively, where $\mathcal{P}^{\mathtt{rev}} = \{(\boldsymbol{I} - \Pi_i, \Pi_i)\}_i$. Then, $\Pr[|(1 - x) - y| \leq \epsilon] \geq 1 - \delta$.*
- *The expected running time of $\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D}$ is $T_{\mathcal{P},D} \cdot \mathrm{poly}(1/\epsilon, \log(1/\delta))$ where $T_{\mathcal{P},D}$ is the combined running time of D, the procedure mapping $i \to (\boldsymbol{P}_i, \boldsymbol{I} - \boldsymbol{P}_i)$, and the running time of measurement $(\boldsymbol{P}_i, \boldsymbol{I} - \boldsymbol{P}_i)$.*

Theorem 4.6 ([Zha20, Corollary 1]). *Let q be an efficiently constructible, potentially mixed state, and D_0, D_1 efficiently sampleable distributions. If D_0 and D_1 are computationally indistinguishable, for any inverse polynomial ϵ and any function δ, we have $\Delta^{3\epsilon}_{\mathsf{Shift}}(\mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D_0}(q), \mathcal{API}^{\epsilon,\delta}_{\mathcal{P},D_1}(q)) \leq 2\delta + \mathsf{negl}(\lambda)$.*

Definition 4.7 (Quantum Program with Classical Inputs and Outputs [ALL+21]). *A quantum program with classical inputs is a pair of quantum state q and unitaries $\{\boldsymbol{U}_x\}_{x\in[N]}$ where $[N]$ is the domain, such that the state of the program evaluated on input x is equal to $\boldsymbol{U}_x q \boldsymbol{U}_x^\dagger$. We measure the first register of $\boldsymbol{U}_x q \boldsymbol{U}_x^\dagger$ to obtain an output. We say that $\{\boldsymbol{U}_x\}_{x\in[N]}$ has a compact classical description $\boldsymbol{U}$ when applying $\boldsymbol{U}_x$ can be efficiently computed given $\boldsymbol{U}$ and x.*

Ciphertext-Policy Functional Encryption. We review the definition of ciphertext-policy functional encryption (CPFE) that we use as the building block of our CoIC-KLA secure PKE scheme.

Definition 4.8 (Ciphertext-Policy Functional Encryption). *A CPFE scheme for the circuit space $\mathcal{C}$ and the input space $\mathcal{X}$ is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

- *The setup algorithm* Setup *takes as input a security parameter 1^λ, and outputs a master public key* MPK *and master secret key* MSK.
- *The key generation algorithm* KG *takes as input the master secret key* MSK *and $x \in \mathcal{X}$, and outputs a decryption key* sk_x.
- *The encryption algorithm* Enc *takes as input the master public key* MPK *and $C \in \mathcal{C}$, and outputs a ciphertext* ct.
- *The decryption algorithm* Dec *takes as input a functional decryption key* sk_x *and a ciphertext* ct, *and outputs y.*

Decryption Correctness: *We require* $\mathsf{Dec}(\mathsf{KG}(\mathsf{MSK}, x), \mathsf{Enc}(\mathsf{MPK}, C)) = C(x)$ *for every $C \in \mathcal{C}$, $x \in \mathcal{X}$, and* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$.

Next, we introduce 1-bounded security for CPFE schemes.

Definition 4.9 (1-Bounded Security). *Let* CPFE *be a CPFE scheme. We define the game* $\mathsf{Expt}^{\text{1-bounded}}_{\mathcal{A},\mathsf{CPFE}}(\lambda,\mathsf{coin})$ *as follows.*

1. *The challenger generates* $(\mathsf{MPK},\mathsf{MSK})\leftarrow\mathsf{Setup}(1^\lambda)$ *and sends* MPK *to* $\mathcal{A}$*.* $\mathcal{A}$ *sends* $x\in\mathcal{X}$ *to the challenger. The challenger generates* $\mathsf{sk}_x\leftarrow\mathsf{KG}(\mathsf{MSK},x)$ *and sends* sk_x *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *outputs* (C_0,C_1) *such that* $C_0(x)=C_1(x)$ *and* C_0 *and* C_1 *have the same size. The challenger picks* $\mathsf{coin}\leftarrow\{0,1\}$*, generates* $\mathsf{ct}\leftarrow\mathsf{Enc}(\mathsf{MPK},C_{\mathsf{coin}})$*, and sends* ct *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $\mathsf{coin}'\in\{0,1\}$*.*

We say that CPFE *is* 1*-bounded secure if for every QPT* $\mathcal{A}$*, we have*

$$\mathsf{Adv}^{\text{1-bounded}}_{\mathcal{A},\mathsf{CPFE}}(\lambda)=2\left|\Pr\left[\mathsf{Expt}^{\text{1-bounded}}_{\mathcal{A},\mathsf{CPFE}}(\lambda)=1\right]-\frac{1}{2}\right|=\mathsf{negl}(\lambda).$$

Theorem 4.10 ([GVW12]). *If there exists IND-CPA secure PKE, there exists* 1*-bounded secure CPFE for* P/poly*.* [12]

4.2 Definitions of CoIC-KLA Security

We introduce definitions of CoIC-KLA security. In addition to normal CoIC-KLA security needed to realize our PKE-SKL, we also define what we call strong CoIC-KLA security. We can prove that strong CoIC-KLA security implies CoIC-KLA security. The reason we introduce strong CoIC-KLA is that it is more compatible to our construction strategy in Section 4.3 that uses watermarking technique by Kitagawa and Nishimaki [KN22b].

Definition 4.11 (CoIC-KLA Security). *We say that a PKE scheme* PKE *with the message space* $\mathcal{X}$ *is CoIC-KLA secure, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

1. $\mathcal{C}$ *runs* $(\mathsf{ek}_0,\mathsf{dk}_0)\leftarrow\mathsf{KG}(1^\lambda)$ *and* $(\mathsf{ek}_1,\mathsf{dk}_1)\leftarrow\mathsf{KG}(1^\lambda)$*, and generates* $\mathsf{dk}:=\frac{1}{\sqrt{2}}(|0\rangle|\mathsf{dk}_0\rangle+|1\rangle|\mathsf{dk}_1\rangle)$*.* $\mathcal{C}$ *sends* ek_0*,* ek_1*, and* dk *to* $\mathcal{A}$*.* $\mathcal{A}$ *can get access to the following oracle only once.*

   $\mathcal{O}(\widetilde{\mathsf{dk}})$**:** *On input a possibly malformed decryption key* $\widetilde{\mathsf{dk}}$*, it applies a binary-outcome measurement* $(\boldsymbol{I}-\Pi_{\mathrm{vrfy}},\Pi_{\mathrm{vrfy}})$*, where* Π_{vrfy} *is the projection to the right decryption key, i.e.,*

$$\Pi_{\mathrm{vrfy}}:=\left(\frac{1}{\sqrt{2}}(|0\rangle|\mathsf{dk}_0\rangle+|1\rangle|\mathsf{dk}_1\rangle)\right)\left(\frac{1}{\sqrt{2}}(\langle 0|\langle\mathsf{dk}_0|+\langle 1|\langle\mathsf{dk}_1|)\right).$$

   It returns the measurement outcome (indicating whether the state was projected onto Π_{vrfy} *or not).*

2. $\mathcal{A}$ *sends* $(\mathsf{m}^*_0,\mathsf{m}^*_1)\in\mathcal{X}^2$ *to* $\mathcal{C}$*.* $\mathcal{C}$ *generates* $a,b\leftarrow\{0,1\}$ *and generates* $\mathsf{ct}^*_0\leftarrow\mathsf{Enc}(\mathsf{ek}_0,\mathsf{m}^*_a)$ *and* $\mathsf{ct}^*_1\leftarrow\mathsf{Enc}(\mathsf{ek}_1,\mathsf{m}^*_{a\oplus b})$*.* $\mathcal{C}$ *sends* ct^*_0 *and* ct^*_1 *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a guess* b' *for* b*.* $\mathcal{C}$ *outputs* 1 *if* $b=b'$ *and* 0 *otherwise as the final output of the experiment.*

For any QPT $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(\lambda):=2\cdot\left|\Pr\left[\mathsf{Exp}^{\mathsf{coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)\to 1\right]-\frac{1}{2}\right|\le\mathsf{negl}(\lambda).$$

Definition 4.12 (Strong CoIC-KLA Security). *We say that a PKE scheme* PKE *with the message space* $\mathcal{X}$ *is* ϵ*-strong CoIC-KLA secure, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{s\text{-}coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda,\epsilon)$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

[12] Though [GVW12] present their construction as KPFE instead of CPFE, it is easy to see that they implicitly give CPFE.

1. *$\mathcal{C}$ runs* $(\mathsf{ek}_0, \mathsf{dk}_0) \leftarrow \mathsf{KG}(1^\lambda)$ *and* $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{KG}(1^\lambda)$*, and generates* $\mathit{dk} := \frac{1}{\sqrt{2}}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle)$*. $\mathcal{C}$ sends* ek_0*,* ek_1*, and* dk *to* $\mathcal{A}$*.*

2. *$\mathcal{A}$ sends* $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{X}^2$ *and a quantum circuit* $\mathcal{D} = (q, \boldsymbol{U})$*, where $\mathcal{D}$ is a quantum program with classical inputs and one-bit outputs and $\boldsymbol{U}$ is a compact classical description of* $\{\boldsymbol{U}_{\mathsf{ct}_0,\mathsf{ct}_1}\}_{\mathsf{ct}_0,\mathsf{ct}_1}$ *to $\mathcal{C}$.*

3. *Let D be the following distribution.*

   D**:** *Generate* $a, b \leftarrow \{0,1\}$ *and* $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{ek}_0, \mathsf{m}_a)$ *and* $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{ek}_1, \mathsf{m}_{a\oplus b})$*. Output* $(b, \mathsf{ct}_0, \mathsf{ct}_1)$*.*

   We also let $\mathcal{P} = (\boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1}, \boldsymbol{Q}_{b,\mathsf{ct}_0,\mathsf{ct}_1})_{b,\mathsf{ct}_0,\mathsf{ct}_1}$ *be a collection of binary outcome projective measurements, where*

$$\boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1} = \boldsymbol{U}^\dagger_{\mathsf{ct}_0,\mathsf{ct}_1}(|b\rangle\langle b| \otimes \boldsymbol{I})\boldsymbol{U}_{\mathsf{ct}_0,\mathsf{ct}_1} \quad \textit{and} \quad \boldsymbol{Q}_{b,\mathsf{ct}_0,\mathsf{ct}_1} = \boldsymbol{I} - \boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1}.$$

   Moreover, we let $\mathcal{M}_D = (\boldsymbol{P}_D, \boldsymbol{Q}_D)$ *be binary outcome POVMs, where*

$$\boldsymbol{P}_D = \sum_{r\in\mathcal{R}} \frac{1}{|\mathcal{R}|}\boldsymbol{P}_{D(r)} \quad \textit{and} \quad \boldsymbol{Q}_D = \boldsymbol{I} - \boldsymbol{P}_D.$$

   Note that $\mathcal{R}$ is the random coin space of D and $\boldsymbol{P}_{D(r)} = \boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1}$*, where* $(b, \mathsf{ct}_0, \mathsf{ct}_1) \leftarrow D(r)$*.*[13] *$\mathcal{C}$ applies the measurement* $\mathsf{ProjImp}(\mathcal{M}_D)$ *to q, and obtain a value p. $\mathcal{C}$ outputs* 1 *if* $p \geq \frac{1}{2} + \epsilon$ *and* 0 *otherwise.*

For any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{s\text{-}coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{s\text{-}coic\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda, \epsilon) \to 1\right] \leq \mathsf{negl}(\lambda).$$

Theorem 4.13. *If* PKE *is ϵ-strong CoIC-KLA secure for any inverse polynomial ϵ, then* PKE *is CoIC-KLA secure.*

Proof. Assume there exists $\mathcal{A}$ that breaks CoIC-KLA security of PKE. Without loss of generality, we assume that $\mathcal{A}$ correctly guesses the bit b with probability $\frac{1}{2} + \gamma$ for some inverse polynomial γ. Then, consider the following experiment using $\mathcal{A}$.

1. Execute $\mathsf{Exp}^{\mathsf{coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$ until the point $\mathcal{A}$ outputs $(\mathsf{m}_0^*, \mathsf{m}_1^*)$.
2. Construct a quantum program with classical inputs and outputs $\mathcal{D} = (q, \boldsymbol{U})$, where q is the inner quantum state of $\mathcal{A}$ and $\boldsymbol{U}$ is a compact description of $\{\boldsymbol{U}_{\mathsf{ct}_0,\mathsf{ct}_1}\}_{\mathsf{ct}_0,\mathsf{ct}_1}$ and $\boldsymbol{U}_{\mathsf{ct}_0,\mathsf{ct}_1}$ is a unitary that performs the rest of $\mathcal{A}$'s computations on input $(\mathsf{ct}_0, \mathsf{ct}_1)$.
3. Obtain p by applying $\mathsf{ProjImp}(\mathcal{M}_D)$ to q, where the measurement $\mathcal{M}_D$ and the distribution D are defined in Definition 4.12.

Then, from the definition of ProjImp and the fact that $\mathcal{A}$'s advantage is $\frac{1}{2} + \gamma$, we have $E[p] = \frac{1}{2} + \gamma$. By the averaging argument, we obtain $\Pr\left[p \geq \frac{1}{2} + \frac{\gamma}{2}\right] \geq \frac{\gamma}{2}$. Consider the following adversary $\mathcal{B}$ that attacks $\frac{\gamma}{2}$-strong CoIC-KLA security of PKE.

1. Given, ek_0, ek_1, and dk, $\mathcal{B}$ executes $\mathsf{Exp}^{\mathsf{coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$ until the point $\mathcal{A}$ outputs $(\mathsf{m}_0^*, \mathsf{m}_1^*)$. When $\mathcal{A}$ makes a query to $\mathcal{O}$, $\mathcal{B}$ returns a random bit.
2. $\mathcal{B}$ constructs a quantum program with classical inputs and outputs $\mathcal{D} = (q, \boldsymbol{U})$, where q is the inner quantum state of $\mathcal{A}$, $\boldsymbol{U}$ is a compact description of $\{\boldsymbol{U}_{\mathsf{ct}_0,\mathsf{ct}_1}\}_{\mathsf{ct}_0,\mathsf{ct}_1}$, and $\boldsymbol{U}_{\mathsf{ct}_0,\mathsf{ct}_1}$ is a unitary that performs the rest of $\mathcal{A}$'s computations on input $(\mathsf{ct}_0, \mathsf{ct}_1)$. $\mathcal{B}$ outputs $(\mathsf{m}_0^*, \mathsf{m}_1^*)$ and $\mathcal{D}$.

$\mathcal{B}$ correctly answers to $\mathcal{A}$'s query to $\mathcal{O}$ and correctly simulates $\mathsf{Exp}^{\mathsf{coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}$ for $\mathcal{A}$ with probability $\frac{1}{2}$.[14] Moreover, from the above discussion, under the condition that $\mathcal{B}$ correctly answers to $\mathcal{A}$'s query to $\mathcal{O}$, $\mathcal{B}$ wins with probability $\frac{\gamma}{2}$. Overall, $\mathsf{Adv}^{\mathsf{s\text{-}coic\text{-}kla}}_{\mathsf{PKE},\mathcal{B}}(\lambda) \geq \frac{\gamma}{4}$, which contradicts $\frac{\gamma}{2}$-strong CoIC-KLA security of PKE. This completes the proof. □

[13] The random coin r for D consists of random bits a, b and encryption coins of two ciphertexts.

[14] $\mathcal{B}$ does not apply the verification procedure to the queried state differently from $\mathsf{Exp}^{\mathsf{coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}$. This is not a problem since from the view of $\mathcal{A}$, the experiment simulated by $\mathcal{B}$ is the same as the experiment where the verification process is applied to the queried state, but the result is ignored and a random bit is returned.

4.3 Strong CoIC-KLA Secure PKE from CPFE

We construct a strong CoIC-KLA secure PKE $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ using a CPFE scheme $\mathsf{CPFE} = (\mathsf{CPFE.Setup}, \mathsf{CPFE.KG}, \mathsf{CPFE.Enc}, \mathsf{CPFE.Dec})$ as a building block.

$\mathsf{Gen}(1^\lambda)$**:**

- Generate $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{CPFE.Setup}(1^\lambda)$.
- Generate $x \leftarrow \{0,1\}^\lambda$ and $\mathsf{sk}_x \leftarrow \mathsf{CPFE.KG}(\mathsf{MSK}, x)$.
- Output $\mathsf{ek} := \mathsf{MPK}$ and $\mathsf{dk} := \mathsf{sk}_x$.

$\mathsf{Enc}(\mathsf{ek}, \mathsf{m})$**:**

- Parse $\mathsf{ek} = \mathsf{MPK}$.
- Let $C[\mathsf{m}]$ be a constant circuit that outputs m on any input. C is padded so that it has the same size as the circuit C^* appeared in the security proof.
- Output $\mathsf{ct} \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}, C[\mathsf{m}])$.

$\mathsf{Dec}(\mathsf{dk}, \mathsf{ct})$**:**

- Parse $\mathsf{dk} = \mathsf{sk}_x$.
- Output $\mathsf{m}' \leftarrow \mathsf{CPFE.Dec}(\mathsf{sk}_x, \mathsf{ct})$.

The decryption correctness of PKE follows from that of CPFE. We also have the following theorems.

Theorem 4.14. *If* CPFE *is a* 1*-bounded secure CPFE scheme, then* PKE *is a* ϵ*-strong CoIC-KLA secure PKE scheme for any inverse polynomial* ϵ*.*

Proof. We show that if there exists a QPT adversary $\mathcal{A}$ that breaks ϵ-strong CoIC-KLA security for some inverse polynomial ϵ, then we can construct a QPT adversary $\mathcal{B}$ that contradicts the following lemma.

Lemma 4.15. *Consider the following experiment* $\mathsf{Expt}^{\mathsf{BZ}}_{\mathsf{CPFE},\mathcal{B}}(1^\lambda)$ *between an adversary* $\mathcal{B}$ *and a challenger* $\mathcal{C}$*.*

1. $\mathcal{C}$ *generates* $(\mathsf{MPK}_0, \mathsf{MSK}_0) \leftarrow \mathsf{CPFE.Setup}(1^\lambda)$, $(\mathsf{MPK}_1, \mathsf{MSK}_1) \leftarrow \mathsf{CPFE.Setup}(1^\lambda)$, $x_0, x_1 \leftarrow \{0,1\}^\lambda$, $\mathsf{sk}_{x_0} \leftarrow \mathsf{CPFE.KG}(\mathsf{MSK}_0, x_0)$, *and* $\mathsf{sk}_{x_1} \leftarrow \mathsf{CPFE.KG}(\mathsf{MSK}_1, x_1)$. $\mathcal{C}$ *gives* MPK_0, MPK_1, *and* $\frac{1}{\sqrt{2}}(|0\rangle |\mathsf{sk}_{x_0}\rangle + |1\rangle |\mathsf{sk}_{x_1}\rangle)$ *to* $\mathcal{B}$*.*
2. $\mathcal{B}$ *outputs* x'_0 *and* x'_1*.* $\mathcal{C}$ *outputs* 1 *if* $x'_0 = x_0$ *and* $x'_1 = x_1$ *and* 0 *otherwise.*

Then, for any QPT adversary $\mathcal{B}$*, we have* $\mathsf{Adv}^{\mathsf{BZ}}_{\mathsf{CPFE},\mathcal{B}}(1^\lambda) = \Pr\left[\mathsf{Expt}^{\mathsf{BZ}}_{\mathsf{CPFE},\mathcal{B}}(1^\lambda) = 1\right] = \mathsf{negl}(\lambda)$.

Proof. This lemma directly follows from Lemma 2.21. □

Let ϵ be some inverse polynomial. Assume there exits a QPT $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{s\text{-}coic\text{-}kla}}_{\mathsf{PKE},\mathcal{A}}(\lambda, \epsilon) = \gamma$ for some inverse polynomial γ. We construct the following adversary $\mathcal{B}$.

1. Given MPK_0, MPK_1, and $d\mathcal{K}$, $\mathcal{B}$ sets $\mathsf{ek}_0 := \mathsf{MPK}_0$ and $\mathsf{ek}_1 := \mathsf{MPK}_1$. $\mathcal{B}$ sends ek_0, ek_1, and $d\mathcal{K}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs $(\mathsf{m}^*_0, \mathsf{m}^*_1)$ and $\mathcal{D} = (q, \boldsymbol{U})$, $\mathcal{B}$ outputs $(x'_0, x'_1) \leftarrow \mathcal{E}\mathit{xtract}(\mathsf{MPK}_0, \mathsf{MPK}_1, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathcal{D}, \epsilon)$, where $\mathcal{E}\mathit{xtract}$ is described below.

$\mathcal{E}\mathit{xtract}(\mathsf{MPK}_0, \mathsf{MPK}_1, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathcal{D}, \epsilon)$**:**

- Let $\epsilon' = \epsilon/8\lambda$ and $\delta' = 2^{-\lambda}$.
- Parse $(q, \boldsymbol{U}) \leftarrow \mathcal{D}$.

- Let $\mathcal{P}$ be defined in the same way as that in Definition 4.12 and $D_{0,i}$ and $D_{1,i}$ be the following distributions for every $i \in [\lambda]$.

  $D_{0,i}$**:** Generate $a, b \leftarrow \{0,1\}$. Generate $\mathsf{ct}_0 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_0, C^*[a, b, \mathsf{m}_0, \mathsf{m}_1, i])$, where $C^*[a, b, \mathsf{m}_0, \mathsf{m}_1, i]$ is a circuit that takes x as input and outputs $\mathsf{m}_{a \oplus b \oplus x[i]}$. Generate $\mathsf{ct}_1 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_1, C[\mathsf{m}_a])$. Output $(b, \mathsf{ct}_0, \mathsf{ct}_1)$.

  $D_{1,i}$**:** Generate $a, b \leftarrow \{0,1\}$. Generate $\mathsf{ct}_1 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_0, C^*[a, b, \mathsf{m}_0, \mathsf{m}_1, i])$, where $C^*[a, b, \mathsf{m}_0, \mathsf{m}_1, i]$ is a circuit that takes x as input and outputs $\mathsf{m}_{a \oplus b \oplus x[i]}$. Generate $\mathsf{ct}_0 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_1, C[\mathsf{m}_a])$. Output $(b, \mathsf{ct}_0, \mathsf{ct}_1)$.

- Let D be the distribution defined in the same way as that in Definition 4.12. Compute $\tilde{p}_0 \leftarrow \mathcal{API}^{\epsilon', \delta'}_{\mathcal{P}, D}(q)$. If $\tilde{p}_0 < \frac{1}{2} + \epsilon - 4\epsilon'$, return $\bot$. Otherwise, let $q_{0,0}$ be the post-measurement state, go to the next step.
- For all $i \in [\lambda]$, do the following.
  1. Compute $\tilde{p}_{0,i} \leftarrow \mathcal{API}^{\epsilon', \delta'}_{\mathcal{P}, D_{0,i}}(q_{0,i-1})$. Let $q_{0,i}$ be the post-measurement state.
  2. If $\tilde{p}_{0,i} > \frac{1}{2} + \epsilon - 4(i+1)\epsilon'$, set $x'_0[i] = 0$. If $\tilde{p}_{0,i} < \frac{1}{2} - \epsilon + 4(i+1)\epsilon'$, set $x'_0[i] = 1$. Otherwise, exit the loop and output $\bot$.
- Let $q_{1,0}$ be $q_{0,\lambda}$. For all $i \in [\lambda]$, do the following.
  1. Compute $\tilde{p}_{1,i} \leftarrow \mathcal{API}^{\epsilon', \delta'}_{\mathcal{P}, D_{1,i}}(q_{1,i-1})$. Let $q_{1,i}$ be the post-measurement state.
  2. If $\tilde{p}_{1,i} > \frac{1}{2} + \epsilon - 4(\lambda + i + 1)\epsilon'$, set $x'_1[i] = 0$. If $\tilde{p}_{1,i} < \frac{1}{2} - \epsilon + 4(\lambda + i + 1)\epsilon'$, set $x'_1[i] = 1$. Otherwise, exit the loop and output $\bot$.
- Output $x'_0 = x'_0[1] \| \cdots \| x'_0[\lambda]$ and $x'_1 = x'_1[1] \| \cdots \| x'_1[\lambda]$.

We will estimate $\mathsf{Adv}^{\mathsf{BZ}}_{\mathsf{CPFE}, \mathcal{B}}(1^\lambda)$. We define the events BadDec, and $\mathsf{BadExt}_{0,i}$ and $\mathsf{BadExt}_{1,i}$ for every $i \in [\lambda]$.

BadDec**:** When $\mathcal{B}$ runs $\mathcal{E}\mathit{xtract}(\mathsf{MPK}_0, \mathsf{MPK}_1, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathcal{D}, \epsilon)$, $\tilde{p}_0 < \frac{1}{2} + \epsilon - 4\epsilon'$ holds.

$\mathsf{BadExt}_{0,i}$**:** When $\mathcal{B}$ runs $\mathcal{E}\mathit{xtract}(\mathsf{MPK}_0, \mathsf{MPK}_1, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathcal{D}, \epsilon)$, the following conditions hold.

- $\tilde{p}_0 \geq \frac{1}{2} + \epsilon - 4\epsilon'$ holds.
- $x'_0[j] = x_0[j]$ holds for every $j \in [i-1]$.
- $x'_0[i] \neq x_0[i]$ holds.

$\mathsf{BadExt}_{1,i}$**:** When $\mathcal{B}$ runs $\mathcal{E}\mathit{xtract}(\mathsf{MPK}_0, \mathsf{MPK}_1, \mathsf{m}^*_0, \mathsf{m}^*_1, \mathcal{D}, \epsilon)$, the following conditions hold.

- $\tilde{p}_0 \geq \frac{1}{2} + \epsilon - 4\epsilon'$ holds.
- $x'_0[j] = x_0[j]$ holds for every $j \in [\lambda]$.
- $x'_1[j] = x_1[j]$ holds for every $j \in [i-1]$.
- $x'_1[i] \neq x_1[i]$ holds.

From the assumption that $\mathsf{Adv}^{\mathsf{s\text{-}coic\text{-}kla}}_{\mathsf{PKE}, \mathcal{A}}(\lambda) = \gamma$, for $\tilde{p}_0$ computed in $\mathcal{E}\mathit{xtract}$, $\tilde{p}_0 \geq \frac{1}{2} + \epsilon - \epsilon'$ holds with probability $\gamma - \mathsf{negl}(\lambda)$ due to the first item of Theorem 4.5. This means that $\Pr[\mathsf{BadDec}] \leq 1 - \gamma + \mathsf{negl}(\lambda)$. Then, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{BZ}}_{\mathsf{CPFE}, \mathcal{B}}(1^\lambda) &\geq 1 - \left(\Pr[\mathsf{BadDec}] + \sum_{i \in [\lambda]} \Pr[\mathsf{BadExt}_{0,i}] + \sum_{i \in [\lambda]} \Pr[\mathsf{BadExt}_{1,i}] \right) \\
&\geq \gamma - \mathsf{negl}(\lambda) - \left(\sum_{i \in [\lambda]} \Pr[\mathsf{BadExt}_{0,i}] + \sum_{i \in [\lambda]} \Pr[\mathsf{BadExt}_{1,i}] \right).
\end{aligned}$$

Estimation of $\Pr[\mathsf{BadExt}_{0,i}]$ for every $i \in [\lambda]$. We first estimate $\Pr[\mathsf{BadExt}_{0,1}]$. We first consider the case of $x_0[1] = 0$. From the first item of the event, we have $\tilde{p}_0 > \frac{1}{2} + \epsilon - 4\epsilon'$. Let $\tilde{p}'_0 \leftarrow \mathcal{API}^{\epsilon',\delta'}_{\mathcal{P},D}(q_{0,0})$. From the almost-projective property of $\mathcal{API}$, we have

$$\Pr\left[\tilde{p}'_0 > \frac{1}{2} + \epsilon - 4\epsilon' - \epsilon'\right] \geq 1 - \delta'.$$

Lemma 4.16. *When $x_0[1] = 0$, $D_{0,1}$ is computationally indistinguishable from D.*

Proof. The difference between $D_{0,1}$ and D is that ct_0 is generated as $\mathsf{ct}_0 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_0, C^*[a,b,\mathsf{m}_0,\mathsf{m}_1,1])$ in $D_{0,1}$ and it is generated as $\mathsf{ct}_0 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_0, C[\mathsf{m}_{a\oplus b}])$ in D. From the condition that $x_0[1] = 0$, we have $C^*[a,b,\mathsf{m}_0,\mathsf{m}_1,1](x_0) = C[\mathsf{m}_{a\oplus b}](x_0) = \mathsf{m}_{a\oplus b}$. Thus, from the 1-bounded security of CPFE, $D_{0,1}$ and D are computationally indistinguishable when $x_0[1] = 0$. □

Thus, from Theorem 4.6 and Lemma 4.16, we have

$$1 - \delta' \leq \Pr\left[\tilde{p}'_0 > \frac{1}{2} + \epsilon - 5\epsilon'\right] \leq \Pr\left[\tilde{p}_{0,1} > \frac{1}{2} + \epsilon - 8\epsilon'\right] + \mathsf{negl}(\lambda).$$

This means that $\Pr[\mathsf{BadExt}_{0,1}] = \mathsf{negl}(\lambda)$ when $x_0[1] = 0$. We next consider the case of $x_0[1] = 1$. We define the following distribution $D^{\mathtt{rev}}$.

$D^{\mathtt{rev}}$**:** Generate $(b, \mathsf{ct}_0, \mathsf{ct}_1) \leftarrow D$. Output $(1 \oplus b, \mathsf{ct}_0, \mathsf{ct}_1)$.

That is, the first bit of the output is flipped from D. Then, for any random coin r, we have $(\boldsymbol{P}_{D^{\mathtt{rev}}(r)}, \boldsymbol{Q}_{D^{\mathtt{rev}}(r)}) = (\boldsymbol{Q}_{D(r)}, \boldsymbol{P}_{D(r)})$. This is because we have $\boldsymbol{Q}_{b,\mathsf{ct}_0,\mathsf{ct}_1} = \boldsymbol{I} - \boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1} = \boldsymbol{P}_{1\oplus b,\mathsf{ct}_0,\mathsf{ct}_1}$ for any tuple $(b, \mathsf{ct}_0, \mathsf{ct}_1)$. Therefore, $\mathcal{API}^{\epsilon',\delta'}_{\mathcal{P},D^{\mathtt{rev}}}$ is exactly the same process as $\mathcal{API}^{\epsilon',\delta'}_{\mathcal{P}^{\mathtt{rev}},D}$, where $\mathcal{P}^{\mathtt{rev}} = (\boldsymbol{Q}_{b,\mathsf{ct}_0,\mathsf{ct}_1}, \boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1})_{b,\mathsf{ct}_0,\mathsf{ct}_1}$. Let $\tilde{p}'_0 \leftarrow \mathcal{API}^{\epsilon',\delta'}_{\mathcal{P},D^{\mathtt{rev}}}(q_{0,0})$. From, the reverse-almost-projective property of $\mathcal{API}$, we have

$$\Pr\left[\tilde{p}'_0 < \frac{1}{2} - \epsilon + 4\epsilon' + \epsilon'\right] \geq 1 - \delta'.$$

Lemma 4.17. *When $x_0[1] = 1$, $D_{0,1}$ is computationally indistinguishable from $D^{\mathtt{rev}}$.*

Proof. We see that $D^{\mathtt{rev}}$ is identical to the following distribution.

- Generate $a, b \leftarrow \{0,1\}$ and $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{ek}_0, \mathsf{m}_a)$ and $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}_{a\oplus 1\oplus b})$. Output $(b, \mathsf{ct}_0, \mathsf{ct}_1)$.

Then, the difference between $D_{0,1}$ and $D^{\mathtt{rev}}$ is that ct_0 is generated as $\mathsf{ct}_0 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_0, C^*[a,b,\mathsf{m}_0,\mathsf{m}_1,1])$ in $D_{0,1}$ and it is generated as $\mathsf{ct}_0 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_0, C[\mathsf{m}_{a\oplus 1\oplus b}])$ in $D^{\mathtt{rev}}$. From the condition that $x_0[1] = 1$, we have $C^*[a,b,\mathsf{m}_0,\mathsf{m}_1,1](x_0) = C[\mathsf{m}_{a\oplus 1\oplus b}](x_0) = \mathsf{m}_{a\oplus 1\oplus b}$. Thus, from the 1-bounded security of CPFE, $D_{0,1}$ and $D^{\mathtt{rev}}$ are computationally indistinguishable when $x_0[1] = 1$. □

Thus, from Theorem 4.6 and Lemma 4.17, we have

$$1 - \delta' \leq \Pr\left[\tilde{p}'_0 < \frac{1}{2} - \epsilon + 5\epsilon'\right] \leq \Pr\left[\tilde{p}_{0,1} < \frac{1}{2} - \epsilon + 8\epsilon'\right] + \mathsf{negl}(\lambda).$$

This means that $\Pr[\mathsf{BadExt}_{0,1}] = \mathsf{negl}(\lambda)$ when $x_0[1] = 1$.

Overall, $\Pr[\mathsf{BadExt}_{0,1}] = \mathsf{negl}(\lambda)$ regardless of the value of x_0. We can similarly show that $\Pr[\mathsf{BadExt}_{0,i}] = \mathsf{negl}(\lambda)$ for $i \in \{2, \cdots, \lambda\}$ using the fact that $D_{0,i}$ is computationally indistinguishable from D if $x_0[i] = 0$ and it is computationally indistinguishable from $D^{\mathtt{rev}}$ if $x_0[i] = 1$. We omit the details.

Estimation of $\Pr[\mathsf{BadExt}_{1,i}]$ for every $i \in [\lambda]$. We estimate $\Pr[\mathsf{BadExt}_{1,1}]$. We first consider the case of $x_0[\lambda] = 0$ and $x_1[1] = 0$. From the second item of the event, we have $\tilde{p}_{0,\lambda} > \frac{1}{2} + \epsilon - 4(\lambda+1)\epsilon'$. Let $\tilde{p}'_{0,\lambda} \leftarrow \mathcal{API}^{\epsilon',\delta'}_{\mathcal{P},D_{0,\lambda}}(q_{0,\lambda})$. From, the almost-projective property of $\mathcal{API}$, we have

$$\Pr\left[\tilde{p}'_{0,\lambda} > \frac{1}{2} + \epsilon - 4(\lambda+1)\epsilon' - \epsilon'\right] \geq 1 - \delta'.$$

Lemma 4.18. *When $x_0[\lambda] = x_1[1] = 0$, $D_{0,\lambda}$ and $D_{1,1}$ are computationally indistinguishable.*

Proof. We can show that $D_{0,\lambda}$ is computationally indistinguishable from D when $x_0[\lambda] = 0$ similarly to Lemma 4.16. We see that D is identical to the following distribution.

- Generate $a, b \leftarrow \{0,1\}$ and $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{ek}_0, \mathsf{m}_{a\oplus b})$ and $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}_a)$. Output $(b, \mathsf{ct}_0, \mathsf{ct}_1)$.

Then, the difference between $D_{1,1}$ and D is that ct_1 is generated as $\mathsf{ct}_1 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_1, C^*[a, b, \mathsf{m}_0, \mathsf{m}_1, 1])$ in $D_{1,1}$ and it is generated as $\mathsf{ct}_1 \leftarrow \mathsf{CPFE.Enc}(\mathsf{MPK}_1, C[\mathsf{m}_{a\oplus b}])$ in D. From the condition that $x_1[1] = 0$, we have $C^*[a, b, \mathsf{m}_0, \mathsf{m}_1, 1](x_1) = C[\mathsf{m}_{a\oplus b}](x_1) = \mathsf{m}_{a\oplus b}$. Thus, from the 1-bounded security of CPFE, $D_{1,1}$ and D are computationally indistinguishable when $x_0[1] = 0$. This means that $D_{0,\lambda}$ and $D_{1,1}$ are computationally indistinguishable when $x_0[\lambda] = x_1[1] = 0$. □

Thus, from Theorem 4.6 and Lemma 4.18, we have

$$1 - \delta' \leq \Pr\left[\tilde{p}'_{0,\lambda} > \frac{1}{2} + \epsilon - (4\lambda+5)\epsilon'\right] \leq \Pr\left[\tilde{p}_{1,1} > \frac{1}{2} + \epsilon - 4(\lambda+2)\epsilon'\right] + \mathsf{negl}(\lambda).$$

This means that $\Pr[\mathsf{BadExt}_{1,1}] = \mathsf{negl}(\lambda)$ when $x_0[\lambda] = 0$ and $x_1[1] = 0$. We next consider the case of $x_0[\lambda] = 0$ and $x_1[1] = 1$. We define the following distribution $D^{\mathtt{rev}}_{0,\lambda}$.

$D^{\mathtt{rev}}_{0,\lambda}$**:** Generate $(b, \mathsf{ct}_0, \mathsf{ct}_1) \leftarrow D_{0,\lambda}$. Output $(1 \oplus b, \mathsf{ct}_0, \mathsf{ct}_1)$.

That is, the first bit of the output is flipped from $D_{0,\lambda}$. Then, for any random coin r, we have $(\boldsymbol{P}_{D^{\mathtt{rev}}_{0,\lambda}(r)}, \boldsymbol{Q}_{D^{\mathtt{rev}}_{0,\lambda}(r)}) = (\boldsymbol{Q}_{D_{0,\lambda}(r)}, \boldsymbol{P}_{D_{0,\lambda}(r)})$. (Again, this is because we have $\boldsymbol{Q}_{b,\mathsf{ct}_0,\mathsf{ct}_1} = \boldsymbol{I} - \boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1} = \boldsymbol{P}_{1\oplus b,\mathsf{ct}_0,\mathsf{ct}_1}$ for any tuple $(b, \mathsf{ct}_0, \mathsf{ct}_1)$.) Therefore, $\mathcal{API}^{\epsilon',\delta'}_{\mathcal{P},D^{\mathtt{rev}}_{0,\lambda}}$ is exactly the same process as $\mathcal{API}^{\epsilon',\delta'}_{\mathcal{P}^{\mathtt{rev}},D_{0,\lambda}}$, where $\mathcal{P}^{\mathtt{rev}} = (\boldsymbol{Q}_{b,\mathsf{ct}_0,\mathsf{ct}_1}, \boldsymbol{P}_{b,\mathsf{ct}_0,\mathsf{ct}_1})_{b,\mathsf{ct}_0,\mathsf{ct}_1}$. Let $\tilde{p}'_{0,\lambda} \leftarrow \mathcal{API}^{\epsilon',\delta'}_{\mathcal{P},D^{\mathtt{rev}}_{0,\lambda}}(q_{0,\lambda})$. From, the reverse-almost-projective property of $\mathcal{API}$, we have

$$\Pr\left[\tilde{p}'_{0,\lambda} < \frac{1}{2} - \epsilon + 4(\lambda+1)\epsilon' + \epsilon'\right] \geq 1 - \delta'.$$

Lemma 4.19. *When $x_0[\lambda] = 0$ and $x_1[1] = 1$, $D^{\mathtt{rev}}_{0,\lambda}$ and $D_{1,1}$ are computationally indistinguishable.*

Proof. We can show that both $D^{\mathtt{rev}}_{0,\lambda}$ and $D_{1,1}$ are computationally indistinguishable from $D^{\mathtt{rev}}$ when $x_0[\lambda] = 0$ and $x_1[1] = 1$. The proof is similarly to those for Lemmata 4.16 to 4.18, thus we omit the details. □

Thus, from Theorem 4.6 and Lemma 4.19, we have

$$1 - \delta' \leq \Pr\left[\tilde{p}'_{0,\lambda} < \frac{1}{2} - \epsilon + (4\lambda+5)\epsilon'\right] \leq \Pr\left[\tilde{p}_{1,1} < \frac{1}{2} - \epsilon + 4(\lambda+2)\epsilon'\right] + \mathsf{negl}(\lambda).$$

This means that $\Pr[\mathsf{BadExt}_{1,1}] = \mathsf{negl}(\lambda)$ when $x_0[\lambda] = 0$ and $x_1[1] = 1$.

Similarly, we can show that $\Pr[\mathsf{BadExt}_{1,1}] = \mathsf{negl}(\lambda)$ holds when $(x_0[\lambda], x_1[1]) = (1,0)$ and $(x_0[\lambda], x_1[1]) = (1,1)$. Moreover, we can show that $\Pr[\mathsf{BadExt}_{1,i}] = \mathsf{negl}(\lambda)$ holds for $i \in \{2, \cdots, \lambda\}$.

From the above discussion, we have $\mathsf{Adv}^{\mathsf{BZ}}_{\mathsf{CPFE},\mathcal{B}}(1^\lambda) \geq \gamma - \mathsf{negl}(\lambda)$ for some inverse polynomial γ, which contradicts Lemma 4.15. This completes the proof of Theorem 4.14. □

5 Construction of PKE with Secure Key Leasing

In this section, we prove the following theorem:

Theorem 5.1. *If there is an IND-CPA secure PKE scheme, then there is an IND-KLA secure PKE-SKL scheme.*

By Theorem 3.9, it suffices to construct 1-query OW-KLA secure PKE-SKL scheme. In the rest of this section, we construct such a scheme. To build our scheme, we rely on a PKE scheme satisfying CoIC-KLA security, which is constructed from any IND-CPA secure PKE scheme in Section 4.

Let $\mathsf{cPKE} = (\mathsf{cPKE.KG}, \mathsf{cPKE.Enc}, \mathsf{cPKE.Dec})$ be a PKE scheme satisfying CoIC-KLA security with message space $\{0,1\}^\ell$ where $\ell = \omega(\log \lambda)$. We note that CoIC-KLA security implies OW-CPA security when $\ell = \omega(\log \lambda)$. (See Appendix B for the proof.) Then, we construct a PKE-SKL scheme $(\mathsf{SKL}.\mathcal{KG}, \mathsf{SKL.Enc}, \mathsf{SKL}.\mathcal{D}ec, \mathsf{SKL}.\mathcal{V}rfy)$ with message space $\{0,1\}^{\lambda\ell}$ as follows.

$\mathsf{SKL}.\mathcal{KG}(1^\lambda)$**:**

- Generate $(\mathsf{cPKE.ek}_{i,b}, \mathsf{cPKE.dk}_{i,b}) \leftarrow \mathsf{cPKE.KG}(1^\lambda)$ for $i \in [\lambda]$ and $b \in \{0,1\}$.
- Output an encryption key
$$\mathsf{ek} := \{\mathsf{cPKE.ek}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}},$$
a decryption key
$$\mathsf{dk} := \bigotimes_{i\in[\lambda]} \frac{1}{\sqrt{2}} \left(|0\rangle |\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle |\mathsf{cPKE.dk}_{i,1}\rangle\right),$$
and a verification key
$$\mathsf{vk} := \{\mathsf{cPKE.dk}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}.$$
For convenience, we write DK_i to mean the registers of dk that contains $\frac{1}{\sqrt{2}} \left(|0\rangle |\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle |\mathsf{cPKE.dk}_{i,1}\rangle\right)$ for $i \in [\lambda]$.

$\mathsf{SKL.Enc}(\mathsf{ek}, \mathsf{m})$**:**

- Parse $\mathsf{ek} = \{\mathsf{cPKE.ek}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and $\mathsf{m} = \mathsf{m}_1 \| \dots \| \mathsf{m}_\lambda$ where $\mathsf{m}_i \in \{0,1\}^\ell$ for each $i \in [\lambda]$.
- Generate $\mathsf{cPKE.ct}_{i,b} \leftarrow \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}_i)$ for $i \in [\lambda]$ and $b \in \{0,1\}$.
- Output $\mathsf{ct} := \{\mathsf{cPKE.ct}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$.

$\mathsf{SKL}.\mathcal{D}ec(\mathsf{dk}, \mathsf{ct})$**:**

- Parse $\mathsf{dk} = \bigotimes_{i\in[\lambda]} \mathsf{dk}_i$ and $\mathsf{ct} = \{\mathsf{cPKE.ct}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$.
- Let U_{dec} be a unitary such that for all $\mathsf{cPKE.dk}'$, $\mathsf{cPKE.ct}'_0$, and $\mathsf{cPKE.ct}'_1$:
$$\begin{aligned} &|b\rangle \left|\mathsf{cPKE.dk}'\right\rangle \left|\mathsf{cPKE.ct}'_0, \mathsf{cPKE.ct}'_1\right\rangle |0\rangle \\ \xrightarrow{U_{\mathrm{dec}}} &|b\rangle \left|\mathsf{cPKE.dk}'\right\rangle \left|\mathsf{cPKE.ct}'_0, \mathsf{cPKE.ct}'_1\right\rangle \left|\mathsf{cPKE.Dec}(\mathsf{cPKE.dk}', \mathsf{cPKE.ct}'_b)\right\rangle \end{aligned}$$
Note that such a unitary can be computed in quantum polynomial-time since we assume that $\mathsf{cPKE.Dec}$ is a deterministic classical polynomial-time algorithm.
- For all $i \in [\lambda]$, generate
$$U_{\mathrm{dec}} \left(\mathsf{dk}_i \otimes |\mathsf{cPKE.ct}_{i,0}, \mathsf{cPKE.ct}_{i,1}\rangle \langle \mathsf{cPKE.ct}_{i,0}, \mathsf{cPKE.ct}_{i,1}| \otimes |0\rangle \langle 0|\right) U_{\mathrm{dec}}^\dagger,$$
measure the rightmost register, and let m'_i be the measurement outcome.
- Output $\mathsf{m}' := \mathsf{m}'_1 \| \dots \| \mathsf{m}'_\lambda$.

$\mathsf{SKL}.\mathcal{V}rfy(\mathsf{vk}, \widetilde{d\mathcal{K}})$:

- Parse $\mathsf{vk} = \{\mathsf{cPKE.dk}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$.
- Apply a binary-outcome measurement $(\boldsymbol{I} - \Pi^{\mathsf{vk}}_{\mathrm{vrfy}}, \Pi^{\mathsf{vk}}_{\mathrm{vrfy}})$ on $\widetilde{d\mathcal{K}}$ where $\Pi^{\mathsf{vk}}_{\mathrm{vrfy}}$ is the projection onto the right decryption key, i.e.,

$$\Pi^{\mathsf{vk}}_{\mathrm{vrfy}} := \bigotimes_{i\in[\lambda]} \left(\frac{1}{\sqrt{2}}\left(|0\rangle|\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle|\mathsf{cPKE.dk}_{i,1}\rangle\right)\right)\left(\frac{1}{\sqrt{2}}\left(\langle 0|\langle \mathsf{cPKE.dk}_{i,0}| + \langle 1|\langle \mathsf{cPKE.dk}_{i,1}|\right)\right).$$

  If the measurement outcome is 1 (indicating that the state was projected onto $\Pi^{\mathsf{vk}}_{\mathrm{vrfy}}$), output $\top$ and otherwise output $\bot$.

The correctness of SKL easily follows from that of cPKE. Below, we show that SKL is 1-query OW-KLA secure.

Theorem 5.2. *If* cPKE *is CoIC-KLA secure, then* SKL *is 1-query OW-KLA secure.*

Proof. Let $\mathcal{A}$ be a QPT adversary against 1-query OW-KLA security of SKL. By Remark 3.5, we assume that $\mathcal{A}$ makes the verification query before receiving the challenge ciphertext without loss of generality. We consider the following sequence of hybrids.

Hyb_0: This is the same as $\mathsf{Exp}^{\mathsf{ow\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda)$. More specifically, it works as follows.

1. The challenger generates $(\mathsf{cPKE.ek}_{i,b}, \mathsf{cPKE.dk}_{i,b}) \leftarrow \mathsf{cPKE.KG}(1^\lambda)$ for $i \in [\lambda]$ and $b \in \{0,1\}$, sets $\mathsf{ek} := \{\mathsf{cPKE.ek}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and $d\mathcal{K} := \bigotimes_{i\in[\lambda]} \frac{1}{\sqrt{2}}\left(|0\rangle|\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle|\mathsf{cPKE.dk}_{i,1}\rangle\right)$, and sends ek and $d\mathcal{K}$ to $\mathcal{A}$.
2. $\mathcal{A}$ queries $\widetilde{d\mathcal{K}}$ to the verification oracle. The challenger applies a binary-outcome measurement $(\boldsymbol{I} - \Pi^{\mathsf{vk}}_{\mathrm{vrfy}}, \Pi^{\mathsf{vk}}_{\mathrm{vrfy}})$ on $\widetilde{d\mathcal{K}}$where $\Pi^{\mathsf{vk}}_{\mathrm{vrfy}}$ is the projection defined in the description of $\mathsf{SKL}.\mathcal{V}rfy$. If the measurement outcome is 0 (indicating that the state was projected onto $\boldsymbol{I} - \Pi^{\mathsf{vk}}_{\mathrm{vrfy}}$), the challenger outputs 0 as the final outcome of this experiment.[15] Otherwise, the challenger returns $\top$ to $\mathcal{A}$ as the response from the oracle.
3. The challenger chooses $\mathsf{m}^*_i \leftarrow \{0,1\}^\ell$ for $i \in [\lambda]$, generates $\mathsf{cPKE.ct}^*_{i,b} \leftarrow \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_i)$ for $i \in [\lambda]$ and $b \in \{0,1\}$, and sends $\mathsf{ct}^* := \{\mathsf{cPKE.ct}^*_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ to $\mathcal{A}$.[16]
4. $\mathcal{A}$ outputs $\mathsf{m}' = \mathsf{m}'_1\|\dots\|\mathsf{m}'_\lambda$. The challenger outputs 1 if $\mathsf{m}'_i = \mathsf{m}^*_i$ for all $i \in [\lambda]$ and otherwise 0 as the final outcome of this experiment.

Note that we have $\Pr[\mathsf{Hyb}_0 = 1] = \mathsf{Adv}^{\mathsf{ow\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(1^\lambda)$. Our goal is to prove $\Pr[\mathsf{Hyb}_0 = 1] = \mathsf{negl}(\lambda)$.

Hyb_1: This is identical to Hyb_0 except for the following modifications:

- The challenger chooses $\mathsf{m}^*_{i,b} \leftarrow \{0,1\}^\ell$ for $i \in [\lambda]$ and $b \in \{0,1\}$ (instead of choosing $\mathsf{m}^*_i \leftarrow \{0,1\}^\ell$ for $i \in [\lambda]$) and $a_i \leftarrow \{0,1\}$ for $i \in [\lambda]$.
- $\mathsf{cPKE.ct}^*_{i,b}$ is generated as $\mathsf{cPKE.ct}^*_{i,b} \leftarrow \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,a_i})$ for $i \in [\lambda]$ and $b \in \{0,1\}$. We emphasize that m^*_{i,a_i} is encrypted for both cases of $b = 0$ and $b = 1$ and $\mathsf{m}^*_{i,a_i\oplus 1}$ is not used in this step.
- In Step 4, the challenger outputs 1 if $\mathsf{m}'_i \in \{\mathsf{m}^*_{i,0}, \mathsf{m}^*_{i,1}\}$ for all $i \in [\lambda]$.

By considering m^*_{i,a_i} in Hyb_1 as m^*_i in Hyb_0, these hybrids are identical from the view of $\mathcal{A}$ except that the winning condition (i.e., the condition that the challenger returns 1) is just relaxed in Hyb_1. Therefore, we trivially have $\Pr[\mathsf{Hyb}_0 = 1] \le \Pr[\mathsf{Hyb}_1 = 1]$.

[15] In the description of the OW-KLA experiment in Definition 3.7, the oracle returns $\bot$ even if the decryption key does not pass the verification. However, in the 1-query setting, if the first (and only) query is rejected, the experiment finally outputs 0. Thus, we terminate the experiment at this point when the query is rejected.

[16] Since $\mathcal{A}$ makes only one verification query, we can assume that $\mathcal{A}$ requests the challenge ciphertext immediately after finishing the first verification query without loss of generality.

Hyb_2: This is identical to Hyb_1 except that $\mathsf{cPKE.ct}^*_{i,b}$ is generated as $\mathsf{cPKE.ct}^*_{i,b} \leftarrow \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,a_i \oplus b})$ for $i \in [\lambda]$ and $b \in \{0,1\}$. We remark that the way of generating $\mathsf{cPKE.ct}^*_{i,1}$ is changed but that of $\mathsf{cPKE.ct}^*_{i,0}$ is unchanged (because $a_i \oplus 0 = a_i$).

By the CoIC-KLA security of cPKE and a standard hybrid argument, we have $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| = \mathsf{negl}(\lambda)$. See Lemma 5.3 for the detail.

Hyb_3: This is identical to Hyb_2 except that the challenger quits choosing $a_i \leftarrow \{0,1\}$ for $i \in [\lambda]$ and $\mathsf{cPKE.ct}^*_{i,b}$ is generated as $\mathsf{cPKE.ct}^*_{i,b} \leftarrow \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,b})$ for $i \in [\lambda]$ and $b \in \{0,1\}$.

This modification is just conceptual and we have $\Pr[\mathsf{Hyb}_2 = 1] = \Pr[\mathsf{Hyb}_3 = 1]$.

Hyb_4: This is identical to Hyb_3 except for a conceptual modification that the measurement of the returned key $\widetilde{dk}$ is deferred until the end of the experiment. For clarity, we give the full description of this experiment.

1. The challenger generates $(\mathsf{cPKE.ek}_{i,b}, \mathsf{cPKE.dk}_{i,b}) \leftarrow \mathsf{cPKE.KG}(1^\lambda)$ for $i \in [\lambda]$ and $b \in \{0,1\}$, sets $\mathsf{ek} := \{\mathsf{cPKE.ek}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and $dk := \bigotimes_{i\in[\lambda]} \frac{1}{\sqrt{2}}(|0\rangle |\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle |\mathsf{cPKE.dk}_{i,1}\rangle)$, and sends ek and dk to $\mathcal{A}$.
2. $\mathcal{A}$ queries $\widetilde{dk}$ to the verification oracle. The challenger returns $\top$ to $\mathcal{A}$ as the response from the oracle.
3. The challenger chooses $\mathsf{m}^*_{i,b} \leftarrow \{0,1\}^\ell$ for $i \in [\lambda]$ and $b \in \{0,1\}$ generates $\mathsf{cPKE.ct}^*_{i,b} \leftarrow \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,b})$ for $i \in [\lambda]$ and $b \in \{0,1\}$, and sends $\mathsf{ct}^* := \{\mathsf{cPKE.ct}^*_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{m}' = \mathsf{m}'_1 \| \dots \| \mathsf{m}'_\lambda$. The challenger outputs 0 as the final outcome of this experiment if $\mathsf{m}'_i \notin \{\mathsf{m}^*_{i,0}, \mathsf{m}^*_{i,1}\}$ for some $i \in [\lambda]$.
5. Otherwise, the challenger applies a binary-outcome measurement $(\boldsymbol{I} - \Pi^{\mathsf{vk}}_{\mathrm{vrfy}}, \Pi^{\mathsf{vk}}_{\mathrm{vrfy}})$ on $\widetilde{dk}$where $\Pi^{\mathsf{vk}}_{\mathrm{vrfy}}$ is the projection defined in the description of $\mathsf{SKL}.\mathcal{V}rfy$. the challenger outputs the outcome of the measurement as the final outcome of this experiment.

By the deferred measurement principle, we have $\Pr[\mathsf{Hyb}_3 = 1] = \Pr[\mathsf{Hyb}_4 = 1]$.

Hyb_5: This is identical to Hyb_4 except that the challenger measures the returned key $\widetilde{dk}$ in the *computational basis* instead of applying the projective measurement $(\boldsymbol{I} - \Pi^{\mathsf{vk}}_{\mathrm{vrfy}}, \Pi^{\mathsf{vk}}_{\mathrm{vrfy}})$ in Step 5, and the condition to output 1 is modified as follows:

- Let $\{\widetilde{b}_i, \mathsf{cPKE.}\widetilde{\mathsf{dk}}_i\}_{i\in[\lambda]}$ be the outcome of the measurement of $\widetilde{dk}$ in the computational basis. If there is $i \in [\lambda]$ such that $\mathsf{cPKE.}\widetilde{\mathsf{dk}}_i \neq \mathsf{cPKE.dk}_{i,\widetilde{b}_i}$, the challenger outputs 0 as the final outcome of this experiment. Otherwise, define $\mathbf{b} = b_1 \| \dots \| b_\lambda \in \{0,1\}^\lambda$ in such a way that $\mathsf{m}'_i = \mathsf{m}^*_{i,b_i}$ for $i \in [\lambda]$. Note that such $\mathbf{b}$ must exist since this step is invoked only when the challenger does not output 0 in Step 4.[17] If there is $i \in [\lambda]$ such that $\widetilde{b}_i \neq b_i$, the challenger outputs 1 and otherwise 0 as the final output of the experiment.

We prove that if $\Pr[\mathsf{Hyb}_5 = 1] = \mathsf{negl}(\lambda)$, then it holds that $\Pr[\mathsf{Hyb}_4 = 1] = \mathsf{negl}(\lambda)$. The intuition is as follows: If we have $\widetilde{b}_i = b_i$ with overwhelming probability, then $\widetilde{dk}$ has a negligible amplitude on $\bigotimes_{i\in[\lambda]} |b'_i\rangle \left|\mathsf{cPKE.}\widetilde{\mathsf{dk}}_{b'_i}\right\rangle$ for all $\mathbf{b}' \neq \mathbf{b}$. In this case, the probability that $\widetilde{dk}$ is projected onto $\Pi^{\mathsf{vk}}_{\mathrm{vrfy}}$ is negligible since the right key dk has an exponentially small amplitude on $\bigotimes_{i\in[\lambda]} |b_i\rangle \left|\mathsf{cPKE.}\widetilde{\mathsf{dk}}_{b_i}\right\rangle$. See Lemma 5.4 for the detail.

Hyb_6: This is identical to Hyb_5 except that the challenger chooses $i^* \leftarrow [\lambda]$ at the beginning of the experiment and the condition to output 1 is modified to that $\widetilde{b}_{i^*} \neq b_{i^*}$ holds for the a priori chosen i^* instead of for some $i \in [\lambda]$.

Whenever there is i such that $\widetilde{b}_i \neq b_i$, the probability that $i^* \leftarrow [\lambda]$ satisfies $\widetilde{b}_{i^*} \neq b_{i^*}$ is at least $\frac{1}{\lambda}$. Thus, we have $\Pr[\mathsf{Hyb}_6 = 1] \geq \frac{1}{\lambda}\Pr[\mathsf{Hyb}_5 = 1]$.

[17] If $\mathsf{m}^*_{i,0} = \mathsf{m}^*_{i,1}$ (which happens with a negligible probability), then we set $b_i := 0$.

Hyb_7**:** This is identical to Hyb_6 except that challenger measures the register DK_{i^*} of the decryption key $\widetilde{dk}$ in the computational basis before giving $\widetilde{dk}$ to $\mathcal{A}$. (See the description of $\mathsf{SKL}.\mathcal{KG}$ for the definition of register DK_{i^*}.)

Note that the measurement of DK_{i^*} in the computational basis yields either $(0, \mathsf{cPKE.dk}_{i^*,0})$ or $(1, \mathsf{cPKE.dk}_{i^*,1})$. In particular, there are only two possible outcomes. Thus, by Lemma 2.21, we have $\Pr[\mathsf{Hyb}_7 = 1] \geq \frac{1}{2}\Pr[\mathsf{Hyb}_6 = 1]$.

Hyb_8**:** This is identical to Hyb_7 except that the collapsing caused by measuring DK_{i^*} is simulated by classical randomness. That is, the challenger chooses $b^* \leftarrow \{0,1\}$ at the beginning and sets

$$\widetilde{dk} := \bigotimes_{i\in[\lambda]\setminus\{i^*\}} \frac{1}{\sqrt{2}}\left(|0\rangle|\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle|\mathsf{cPKE.dk}_{i,1}\rangle\right)_{\mathsf{DK}_i} \otimes \left(|b^*\rangle|\mathsf{cPKE.dk}_{i^*,b^*}\rangle\right)_{\mathsf{DK}_{i^*}}.$$

It is easy to see that Hyb_7 and Hyb_8 are identical from the view of $\mathcal{A}$, and thus we have $\Pr[\mathsf{Hyb}_7 = 1] = \Pr[\mathsf{Hyb}_8 = 1]$. In Lemma 5.6, we prove that $\Pr[\mathsf{Hyb}_8 = 1] = \mathsf{negl}(\lambda)$ by using the OW-CPA security (which is implied by CoIC-KLA security) of SKL.

By combining the above, we have $\Pr[\mathsf{Hyb}_0 = 1] = \mathsf{negl}(\lambda)$. This means that SKL is OW-KLA secure. We are left to prove Lemmata 5.3, 5.4 and 5.6

Lemma 5.3. *It holds that* $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| = \mathsf{negl}(\lambda)$ *if* cPKE *is CoIC-KLA secure.*

Proof. We define additional hybrids $\mathsf{Hyb}_{1.j}$ for $j \in [\lambda+1]$ as follows.

$\mathsf{Hyb}_{1.j}$**:** This is identical to Hyb_1 except that $\mathsf{cPKE.ct}^*_{i,b}$ is generated as

$$\mathsf{cPKE.ct}^*_{i,b} \leftarrow \begin{cases} \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,a_i\oplus b}) & i < j \\ \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,a_i}) & i \geq j \end{cases}$$

for $i \in [\lambda]$.

Clearly, we have $\mathsf{Hyb}_1 = \mathsf{Hyb}_{1.1}$ and $\mathsf{Hyb}_2 = \mathsf{Hyb}_{1.\lambda+1}$. Thus, it suffices to prove that $\left|\Pr\left[\mathsf{Hyb}_{1.j+1} = 1\right] - \Pr\left[\mathsf{Hyb}_{1.j} = 1\right]\right| = \mathsf{negl}(\lambda)$. Remark that the only difference between $\mathsf{Hyb}_{1.j+1}$ and $\mathsf{Hyb}_{1.j}$ is the way of generating $\mathsf{cPKE.ct}^*_{j,1}$. To show that $\left|\Pr\left[\mathsf{Hyb}_{1.j+1} = 1\right] - \Pr\left[\mathsf{Hyb}_{1.j} = 1\right]\right| = \mathsf{negl}(\lambda)$, we construct $\mathcal{B}$ against CoIC-KLA security of cPKE as follows.

$\mathcal{B}(\mathsf{cPKE.ek}^*_0, \mathsf{cPKE.ek}^*_1, \widetilde{dk}^*)$**:** It works as follows.

1. Generate $(\mathsf{cPKE.ek}_{i,b}, \mathsf{cPKE.dk}_{i,b}) \leftarrow \mathsf{cPKE.KG}(1^\lambda)$ for $i \in [\lambda]\setminus\{j\}$ and $b \in \{0,1\}$ and set $\mathsf{cPKE.ek}_{j,b} := \mathsf{cPKE.ek}^*_b$ for $b \in \{0,1\}$. Set $\mathsf{ek} := \{\mathsf{cPKE.ek}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and
$$\widetilde{dk} := \bigotimes_{i\in[\lambda]\setminus\{j\}} \frac{1}{\sqrt{2}}\left(|0\rangle|\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle|\mathsf{cPKE.dk}_{i,1}\rangle\right)_{\mathsf{DK}_i} \otimes \widetilde{dk}^*_{\mathsf{DK}_j}.$$
This implicitly defines $\mathsf{vk} := \{\mathsf{cPKE.dk}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ where $\mathsf{cPKE.dk}_{j,b}$ is the decryption key corresponding to $\mathsf{cPKE.dk}_{j,b}$ chosen by the external challenger for $b \in \{0,1\}$ (but $\mathcal{B}$ cannot know vk).
2. Send ek and $\widetilde{dk}$ to $\mathcal{A}$ and receives the verification query $\widetilde{dk}$ from $\mathcal{A}$.
3. Apply a binary-outcome measurement $(\boldsymbol{I} - \Pi^{\mathsf{vk}}_{\mathrm{vrfy}}, \Pi^{\mathsf{vk}}_{\mathrm{vrfy}})$ on $\widetilde{dk}$. This is possible by simulating the projection on $\{\mathsf{DK}_i\}_{i\neq j}$ by itself while forwarding DK_j to its own verification oracle. If the outcome is 0, output 0. Otherwise, return $\top$ to $\mathcal{A}$ as the response from the oracle.
4. Choose $\mathsf{m}^*_{i,b} \leftarrow \{0,1\}^\ell$ for $i \in [\lambda]$ and $b \in \{0,1\}$ and $a_i \leftarrow \{0,1\}$ for $i \in [\lambda]\setminus\{j\}$, send $(\mathsf{m}^*_{j,0}, \mathsf{m}^*_{j,1})$ to the external challenger, and receive $(\mathsf{cPKE.ct}^*_0, \mathsf{cPKE.ct}^*_1)$ from the challenger. This implicitly defines $a_j \leftarrow \{0,1\}$ and $\beta \leftarrow \{0,1\}$ where the challenger generates $\mathsf{cPKE.ct}^*_0 := \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_j, m^*_{j,a_j})$ and $\mathsf{cPKE.ct}^*_1 := \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_j, m^*_{j,a_j\oplus\beta})$ (but $\mathcal{B}$ cannot know a_j or β).[18]

[18]Here, β plays the role of b in the experiment $\mathsf{Exp}^{\mathsf{coic\text{-}kla}}_{\mathsf{cPKE},\mathcal{B}}(1^\lambda)$ in Definition 4.11. This is because b is used in another meaning in this section.

5. Generate $\mathsf{cPKE.ct}^*_{i,b} \leftarrow, \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,a_i \oplus b})$ for $i \in [\lambda] \setminus \{j\}$ and $b \in \{0,1\}$, set $\mathsf{cPKE.ct}^*_{j,b} := \mathsf{cPKE.ct}^*_b$ for $b \in \{0,1\}$, send $\mathsf{ct}^* := \{\mathsf{cPKE.ct}^*_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ to $\mathcal{A}$, and receive $\mathsf{m}' = \mathsf{m}'_1 \| \dots \| \mathsf{m}'_\lambda$ from $\mathcal{A}$.
6. Output 1 if $\mathsf{m}'_i \in \{\mathsf{m}^*_{i,0}, \mathsf{m}^*_{i,1}\}$ for all $i \in [\lambda]$ and otherwise output 0.

We have

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{coic\text{-}kla}}_{\mathsf{cPKE},\mathcal{B}}(\lambda) \\
&= 2\left|\Pr[\mathcal{B}(\mathsf{cPKE.ek}^*_0, \mathsf{cPKE.ek}^*_1, \widetilde{dk}^*) = \beta] - \frac{1}{2}\right| \\
&= |\Pr[\mathcal{B}(\mathsf{cPKE.ek}^*_0, \mathsf{cPKE.ek}^*_1, \widetilde{dk}^*) = 1|\beta = 0] - \Pr[\mathcal{B}(\mathsf{cPKE.ek}^*_0, \mathsf{cPKE.ek}^*_1, \widetilde{dk}^*) = 1|\beta = 1]| \\
&= \left|\Pr\left[\mathsf{Hyb}_{1.j+1} = 1\right] - \Pr\left[\mathsf{Hyb}_{1.j} = 1\right]\right|
\end{aligned}$$

where $(\mathsf{cPKE.ek}^*_0, \mathsf{cPKE.dk}^*_0) \leftarrow \mathsf{cPKE.KG}(1^\lambda)$, $(\mathsf{cPKE.ek}^*_1, \mathsf{cPKE.dk}^*_1) \leftarrow \mathsf{cPKE.KG}(1^\lambda)$, and $\widetilde{dk}^* := \frac{1}{\sqrt{2}}(|\mathsf{cPKE.dk}^*_0\rangle + |\mathsf{cPKE.dk}^*_1\rangle)$. Thus, $\left|\Pr\left[\mathsf{Hyb}_{1.j+1} = 1\right] - \Pr\left[\mathsf{Hyb}_{1.j} = 1\right]\right| = \mathsf{negl}(\lambda)$ by the CoIC-KLA security of cPKE. This completes the proof of Lemma 5.3. □

Lemma 5.4. *If* $\Pr[\mathsf{Hyb}_5 = 1] = \mathsf{negl}(\lambda)$, *then it holds that* $\Pr[\mathsf{Hyb}_4 = 1] = \mathsf{negl}(\lambda)$.

Proof. Let $\epsilon := \Pr[\mathsf{Hyb}_4 = 1]$. For $\mathsf{vk} = \{\mathsf{cPKE.dk}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and $\mathbf{b} = b_1\| \dots \|b_\lambda \in \{0,1\}^\lambda$, let $\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}$ be the event that vk is chosen as a verification key and $\mathsf{m}'_i = \mathsf{m}^*_{i,b_i}$ for all $i \in [\lambda]$. Let $\widetilde{dk}^{\mathsf{vk}}_{\mathbf{b}}$ be the state of the returned key conditioned on $\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}$. Clearly, we have

$$\sum_{\mathsf{vk},\mathbf{b}} \Pr\left[\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}\right] \cdot \mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\mathrm{vrfy}} \widetilde{dk}^{\mathsf{vk}}_{\mathbf{b}}\right) = \epsilon.$$

Let $\mathtt{Good}$ be a subset defined as

$$\mathtt{Good} := \left\{(\mathsf{vk}, \mathbf{b}) : \mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\mathrm{vrfy}} \widetilde{dk}^{\mathsf{vk}}_{\mathbf{b}}\right) \geq \frac{\epsilon}{2}\right\}.$$

Then, by a standard averaging argument, it holds that

$$\sum_{(\mathsf{vk},\mathbf{b})\in\mathtt{Good}} \Pr\left[\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}\right] \geq \frac{\epsilon}{2}. \tag{6}$$

For $\mathsf{vk} = \{\mathsf{cPKE.dk}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and $\mathbf{b} = b_1\| \dots \|b_\lambda$, let $\Pi^{\mathsf{vk}}_{\neq\mathbf{b}}$ be a projection defined as follows:

$$\Pi^{\mathsf{vk}}_{\neq\mathbf{b}} := \sum_{\mathbf{b}'\in\{0,1\}^\lambda\setminus\{\mathbf{b}\}} \bigotimes_{i\in[\lambda]} |b'_i\rangle \left|\mathsf{cPKE.dk}_{i,b'_i}\right\rangle \langle b'_i| \left\langle \mathsf{cPKE.dk}_{i,b'_i}\right|.$$

Then, by the definition of Hyb_5, one can see that

$$\Pr[\mathsf{Hyb}_5 = 1] = \sum_{\mathsf{vk},\mathbf{b}} \Pr\left[\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}\right] \cdot \mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\neq\mathbf{b}} \widetilde{dk}^{\mathsf{vk}}_{\mathbf{b}}\right).$$

Then, we show the following proposition.

Proposition 5.5. *For any* $(\mathsf{vk}, \mathbf{b}) \in \mathtt{Good}$, *it holds that*

$$\mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\neq\mathbf{b}} \widetilde{dk}^{\mathsf{vk}}_{\mathbf{b}}\right) \geq \frac{\epsilon}{4} - 2^{-\lambda}.$$

Proof of Proposition 5.5. By diagonalization, we can write

$$\widetilde{d\mathcal{K}}^{\mathsf{vk}}_{\mathbf{b}} = \sum_{j=1}^{N} p_j \left|\psi_j\right\rangle \left\langle\psi_j\right|$$

where $0 < p_j \leq 1$, $\sum_{j=1}^{N} p_j = 1$, and $\left|\left\langle\psi_j|\psi_j\right\rangle\right| = 1$. For each $j \in [N]$, it holds that

$$\begin{aligned}
&\mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\mathrm{vrfy}} \left|\psi_j\right\rangle \left\langle\psi_j\right|\right) \\
&= 2^{-\lambda} \left\| \left(\sum_{\mathbf{b}' \in \{0,1\}^\lambda} \bigotimes_{i\in[\lambda]} \left\langle b'_i\right| \left\langle \mathsf{cPKE.dk}_{i,b'_i}\right| \right) \left|\psi_j\right\rangle \right\|^2 \\
&\leq 2^{-\lambda+1} \left(\left\| \left(\sum_{\mathbf{b}' \in \{0,1\}^\lambda \setminus \{\mathbf{b}\}} \bigotimes_{i\in[\lambda]} \left\langle b'_i\right| \left\langle \mathsf{cPKE.dk}_{i,b'_i}\right| \right) \left|\psi_j\right\rangle \right\|^2 + \left\| \left(\bigotimes_{i\in[\lambda]} \left\langle b_i\right| \left\langle \mathsf{cPKE.dk}_{i,b_i}\right| \right) \left|\psi_j\right\rangle \right\|^2 \right) \\
&\leq 2^{-\lambda+1} \left(\left(2^\lambda - 1\right) \sum_{\mathbf{b}' \in \{0,1\}^\lambda \setminus \{\mathbf{b}\}} \left\| \left(\bigotimes_{i\in[\lambda]} \left\langle b'_i\right| \left\langle \mathsf{cPKE.dk}_{i,b'_i}\right| \right) \left|\psi_j\right\rangle \right\|^2 + 1 \right) \\
&\leq 2 \left\| \Pi^{\mathsf{vk}}_{\neq \mathbf{b}} \left|\psi_j\right\rangle \right\|^2 + 2^{-\lambda+1}
\end{aligned}$$

where the inequalities in the third and fourth lines follow from Cauchy–Schwarz inequality.

Then, it holds that

$$\begin{aligned}
&\mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\mathrm{vrfy}} \widetilde{d\mathcal{K}}^{\mathsf{vk}}_{\mathbf{b}}\right) \\
&= \sum_{j=1}^{N} p_j \,\mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\mathrm{vrfy}} \left|\psi_j\right\rangle \left\langle\psi_j\right|\right) \\
&\leq \sum_{j=1}^{N} p_j \left(2 \left\| \Pi^{\mathsf{vk}}_{\neq \mathbf{b}} \left|\psi_j\right\rangle \right\|^2 + 2^{-\lambda+1} \right) \\
&= 2\,\mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\neq \mathbf{b}} \widetilde{d\mathcal{K}}^{\mathsf{vk}}_{\mathbf{b}}\right) + 2^{-\lambda+1}.
\end{aligned}$$

Since we assume $(\mathsf{vk}, \mathbf{b}) \in \mathtt{Good}$, it holds that $\mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\mathrm{vrfy}} \widetilde{d\mathcal{K}}^{\mathsf{vk}}_{\mathbf{b}}\right) \geq \frac{\epsilon}{2}$. By combining the above, Proposition 5.5 is proven. □

Then, we have

$$\begin{aligned}
\Pr[\mathsf{Hyb}_5 = 1] &= \sum_{\mathsf{vk},\mathbf{b}} \Pr\left[\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}\right] \cdot \mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\neq \mathbf{b}} \widetilde{d\mathcal{K}}^{\mathsf{vk}}_{\mathbf{b}}\right) \\
&\geq \sum_{(\mathsf{vk},\mathbf{b}) \in \mathtt{Good}} \Pr\left[\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}\right] \cdot \mathrm{Tr}\left(\Pi^{\mathsf{vk}}_{\neq \mathbf{b}} \widetilde{d\mathcal{K}}^{\mathsf{vk}}_{\mathbf{b}}\right) \\
&\geq \sum_{(\mathsf{vk},\mathbf{b}) \in \mathtt{Good}} \Pr\left[\mathsf{E}^{\mathsf{vk}}_{\mathbf{b}}\right] \cdot \left(\frac{\epsilon}{4} - 2^{-\lambda}\right) \\
&\geq \frac{\epsilon}{2} \cdot \left(\frac{\epsilon}{4} - 2^{-\lambda}\right)
\end{aligned}$$

where the second inequality follows from Proposition 5.5 and the third inequality follows from Eq. 6. Recalling that $\epsilon = \Pr[\mathsf{Hyb}_4 = 1]$, the above inequality implies Lemma 5.4. □

Lemma 5.6. *It holds that* $\Pr[\mathsf{Hyb}_8 = 1] = \mathsf{negl}(\lambda)$ *if* cPKE *is OW-CPA secure.*

Proof. For clarity, we give the full description of Hyb_8 below.

Hyb_8**:** It works as follows:

1. The challenger chooses $i^* \leftarrow [\lambda]$ and $b^* \in \{0,1\}$, generates $(\mathsf{cPKE.ek}_{i,b}, \mathsf{cPKE.dk}_{i,b}) \leftarrow \mathsf{cPKE.KG}(1^\lambda)$ for $i \in [\lambda]$ and $b \in \{0,1\}$, sets $\mathsf{ek} := \{\mathsf{cPKE.ek}_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and
$$\mathsf{dk} := \bigotimes_{i\in[\lambda]\setminus\{i^*\}} \frac{1}{\sqrt{2}}\left(|0\rangle\,|\mathsf{cPKE.dk}_{i,0}\rangle + |1\rangle\,|\mathsf{cPKE.dk}_{i,1}\rangle\right)_{\mathsf{DK}_i} \otimes \left(|b^*\rangle\,\left|\mathsf{cPKE.dk}_{i^*,b^*}\right\rangle\right)_{\mathsf{DK}_{i^*}}$$
and sends ek and dk to $\mathcal{A}$.
2. $\mathcal{A}$ queries $\widetilde{\mathsf{dk}}$ to the verification oracle. The challenger returns $\top$ to $\mathcal{A}$ as the response from the oracle.
3. The challenger chooses $\mathsf{m}^*_{i,b} \leftarrow \{0,1\}^\ell$ for $i \in [\lambda]$ and $b \in \{0,1\}$, generates $\mathsf{cPKE.ct}^*_{i,b} \leftarrow \mathsf{cPKE.Enc}(\mathsf{cPKE.ek}_{i,b}, \mathsf{m}^*_{i,b})$ for $i \in [\lambda]$ and $b \in \{0,1\}$, and sends $\mathsf{ct}^* := \{\mathsf{cPKE.ct}^*_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{m}' = \mathsf{m}'_1\|\ldots\|\mathsf{m}'_\lambda$. The challenger outputs 0 as the final output of the experiment if $\mathsf{m}'_i \notin \{\mathsf{m}^*_{i,0}, \mathsf{m}^*_{i,1}\}$ for some $i \in [\lambda]$.
5. Otherwise, the challenger measures $\widetilde{\mathsf{dk}}$ in the computational basis, and let $\{\widetilde{b}_i, \mathsf{cPKE.}\widetilde{\mathsf{dk}}_i\}_{i\in[\lambda]}$ be the outcome. If there is $i \in [\lambda]$ such that $\mathsf{cPKE.}\widetilde{\mathsf{dk}}_i \neq \mathsf{cPKE.dk}_{i,\widetilde{b}_i}$, the challenger outputs 0 as the final outcome of this experiment. Otherwise, define $\mathbf{b} = b_1\|\ldots\|b_\lambda \in \{0,1\}^\lambda$ in such a way that $\mathsf{m}'_i = \mathsf{m}^*_{i,b_i}$ for $i \in [\lambda]$. Note that such $\mathbf{b}$ must exist since this step is invoked only when the challenger does not output 0 in Step 4.[19] If $\widetilde{b}_{i^*} \neq b_{i^*}$, the challenger outputs 1 and otherwise 0 as the final output of the experiment.

Suppose that we simulate Hyb_8 for $\mathcal{A}$ while embedding a problem instance of the OW-CPA security of cPKE into $\mathsf{cPKE.ek}_{i^*,b^*\oplus1}$ and $\mathsf{cPKE.ct}^*_{i^*,b^*\oplus1}$. Remark that this is possible without knowing $\mathsf{cPKE.dk}_{i^*,b^*\oplus1}$. Suppose that $\mathsf{Hyb}_8 = 1$ occurs in the simulated execution. Then, we in particular have $\mathsf{m}'_{i^*} = \mathsf{m}^*_{i^*,b_{i^*}}$, $\mathsf{cPKE.}\widetilde{\mathsf{dk}}_{i^*} = \mathsf{cPKE.dk}_{i^*,\widetilde{b}_{i^*}}$, and $\widetilde{b}_{i^*} \neq b_{i^*}$. We consider the following two sub-cases.

1. If $b_{i^*} = b^*$, then we have $\widetilde{b}_{i^*} = b^* \oplus 1$. This implies $\mathsf{cPKE.}\widetilde{\mathsf{dk}}_{i^*} = \mathsf{cPKE.dk}_{i^*,b^*\oplus1}$. Then we can decrypt $\mathsf{cPKE.ct}_{i^*,b^*\oplus1}$ by honestly running the decryption algorithm with $\mathsf{cPKE.dk}_{i^*,b^*\oplus1}$. This contradicts the OW-CPA security of cPKE.
2. If $b_{i^*} \neq b^*$, then we have $\mathsf{m}'_{i^*} = \mathsf{m}^*_{i^*,b^*\oplus1}$, which is the message encrypted in $\mathsf{cPKE.ek}_{i^*,b^*\oplus1}$. This means that we can break the OW-CPA security of cPKE.

Neither of them occurs with a non-negligible probability assuming the OW-CPA security of cPKE. Thus, $\Pr[\mathsf{Hyb}_8 = 1] = \mathsf{negl}(\lambda)$. This completes the proof of Lemma 5.6. □

This completes the proof of Theorem 5.2. □

Remark 5.7 (On OMUR). We can show that SKL constructed above also satisfies OMUR. Since there is a generic conversion to add OMUR as shown in Lemma 3.10 anyway, we only give a proof sketch.

We reduce OMUR to 1-key OW-KLA security. Suppose that there is an adversary that breaks OMUR, i.e., passes the verification twice. Then roughly speaking, we can use it to break 1-key OW-KLA security by sending one of them to the verification oracle and using the other one to decrypt the challenge message. There is an issue that the reduction algorithm may make only one verification query while the adversary against OMUR may make arbitrarily many verification queries. To resolve this issue, we can use a similar idea to that used in the proof of Lemma 3.10. The reduction algorithm guesses the first two queries to be accepted. Conditioned on that the guess is correct, the reduction algorithm can simulate the verification oracle by simply returning $\bot$ to all queries except for the two queries that are guessed to be accepted until the adversary make the second guessed query. The guess is correct with probability $\binom{Q}{2}^{-1}$ where Q is the number of queries. Thus, the reduction works with a polynomial security loss. Since we already proved that SKL is 1-query OW-KLA secure (Theorem 5.2), the above reduction shows that it satisfies OMUR.

[19] If $\mathsf{m}^*_{i,0} = \mathsf{m}^*_{i,1}$ (which happens with a negligible probability), then we set $b_i := 0$.

6 Attribute-Based Encryption with Secure Key Leasing

6.1 Definitions

Definition 6.1 (ABE with Secure Key Leasing). *An ABE-SKL scheme* ABE-SKL *is a tuple of six algorithms* $(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Cert}, \mathsf{Vrfy})$. *Below, let* $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_\lambda$, *and* $R = \{R_\lambda : \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}\}_\lambda$ *be the ciphertext space, the key attribute space, and the associated relation of* ABE-SKL, *respectively.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{msk})$**:** *The setup algorithm takes a security parameter* 1^λ, *and outputs a public key* pk *and master secret key* msk.

$\mathcal{KG}(\mathsf{msk}, y) \to (\mathit{usk}, \mathsf{vk})$**:** *The key generation algorithm takes a master secret key* msk *and a key attribute* $y \in \mathcal{Y}$, *and outputs a user secret key* usk *and a verification key* vk.

$\mathsf{Enc}(\mathsf{pk}, x, m) \to \mathsf{ct}$**:** *The encryption algorithm takes a public key* pk, *a ciphertext attribute* $x \in \mathcal{X}$, *and a plaintext* m, *and outputs a ciphertext* ct.

$\mathcal{Dec}(\mathit{usk}, x, \mathsf{ct}) \to z$**:** *The decryption algorithm takes a user secret key* usk, *a ciphertext attribute* x, *and a ciphertext* ct *and outputs a value* $z \in \{\bot\} \cup \{0,1\}^\ell$.

$\mathcal{Vrfy}(\mathsf{vk}, \mathit{usk}') \to \top / \bot$**:** *The verification algorithm takes a verification key* vk *and a quantum state* usk', *and outputs* $\top$ *or* $\bot$.

Decryption correctness: *For every* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *satisfying* $R(x,y) = 1$, *we have*

$$\Pr\left[\mathcal{Dec}(\mathit{usk}, x, \mathsf{ct}) = m \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathit{usk}, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x, m)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Verification correctness: *For every* $y \in \mathcal{Y}$, *we have*

$$\Pr\left[\mathcal{Vrfy}(\mathsf{vk}, \mathit{usk}) = \top \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathit{usk}, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 6.2 (Adaptive Indistinguishability against Key Leasing Attacks). *We say that an ABE-SKL scheme* ABE-SKL *for relation* $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ *is secure against adaptive indistinguishability against key leasing attacks (Ada-IND-KLA), if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathcal{A},\mathsf{ABE\text{-}SKL}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger:*

1. *At the beginning, the challenger runs* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and initialize the list* $L_{\mathcal{KG}}$ *to be an empty set. Throughout the experiment,* $\mathcal{A}$ *can access the following oracles.*

   $O_{\mathcal{KG}}(y)$**:** *Given* y, *it finds an entry of the form* (y, vk, V) *from* $L_{\mathcal{KG}}$. *If there is such an entry, it returns* $\bot$. *Otherwise, it generates* $(\mathit{usk}, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y)$, *sends* usk *to* $\mathcal{A}$, *and adds* $(y, \mathsf{vk}, \bot)$ *to* $L_{\mathcal{KG}}$.

   $O_{\mathcal{Vrfy}}(y, \mathit{usk}')$**:** *Given* (y, usk'), *it finds an entry* (y, vk, V) *from* $L_{\mathcal{KG}}$. *(If there is no such entry, it returns* $\bot$.*) It then runs* $d \coloneqq \mathcal{Vrfy}(\mathsf{vk}, \mathit{usk}')$ *and returns* d *to* $\mathcal{A}$. *If* $V = \bot$, *it updates the entry into* (y, vk, d).

2. *When* $\mathcal{A}$ *sends* (x^*, m_0, m_1) *to the challenger, the challenger checks if for any entry* (y, vk, V) *in* $L_{\mathcal{KG}}$ *such that* $R(x^*, y) = 1$, *it holds that* $V = \top$. *If so, the challenger generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x^*, m_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$. *Otherwise, the challenger outputs* 0.

3. $\mathcal{A}$ *continues to make queries to* $O_{\mathcal{KG}}(\cdot)$ *and* $O_{\mathcal{Vrfy}}(\cdot,\cdot)$. *However,* $\mathcal{A}$ *is not allowed to send a key attribute* y *such that* $R(x^*, y) = 1$ *to* $O_{\mathcal{KG}}$.

4. $\mathcal{A}$ *outputs a guess* coin' *for* coin. *The challenger outputs* coin' *as the final output of the experiment.*

For any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(1^\lambda,0)\to 1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(1^\lambda,1)\to 1\right]\right| \le \mathsf{negl}(\lambda).$$

Remark 6.3. In Definition 6.2, the key generation oracle returns $\bot$ if the same y is queried more than once. To handle the situation where multiple keys for the same attribute y are generated, we need to manage indices for y such as $(y,1,vk_1,V_1),(y,2,vk_2,V_2)$. Although we can reflect the index management in the definition, it complicates the definition and prevents readers from understanding the essential idea. Thus, we use the simplified definition above.

We also consider relaxed versions of the above security notion.

Definition 6.4 (Selective indistinguishability against key leasing attacks). *We consider selective indistinguishability against key leasing attacks (Sel-IND-KLA). For doing so, we consider the same security game as that for Ada-IND-KLA except that the adversary $\mathcal{A}$ should declare its target x^* at the beginning of the game (even before it is given* pk*). We then define the advantage* $\mathsf{Adv}^{\mathsf{sel\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(\lambda)$ *for the selective case similarly. We say* $\mathsf{ABE\text{-}SKL}$ *is secure against selective indistinguishability against key leasing attack if for any QPT adversary $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{sel\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(\lambda)$ *is negligible.*

We also consider the following security notion where we introduce additional restriction that the number of distinguishing keys that are issued (and eventually returned) before ct^* is generated is bounded by some predetermined parameter q. Here, distinguishing key refers to a key that can decrypt the challenge ciphertext if it is not returned.

Definition 6.5 (Bounded Distinguishing Key Ada-IND-KLA/Sel-IND-KLA for ABE). *For defining bounded distinguishing key Ada-IND-KLA security, we consider the same security game as that for Ada-IND-KLA (i.e.,* $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathcal{A},\mathsf{ABE\text{-}SKL}}(1^\lambda,\mathsf{coin})$*) except that we change the step 2 in Definition 6.2 with the following:*

2' *When $\mathcal{A}$ sends (x^*,m_0,m_1) to the challenger, the challenger checks if there are at most q entries (y,vk,V) in $L_{\mathcal{KG}}$ such that $R(x^*,y)=1$ and for all these entries, $V=\top$. If so, the challenger generates $\mathsf{ct}^*\leftarrow\mathsf{Enc}(\mathsf{pk},x^*,m_{\mathsf{coin}})$ and sends ct^* to $\mathcal{A}$. Otherwise, the challenger outputs 0.*

We then define the advantage $\mathsf{Adv}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A},q}(\lambda)$ *similarly to* $\mathsf{Adv}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(\lambda)$*. We say* $\mathsf{ABE\text{-}SKL}$ *is q-bounded distinguishing key Ada-IND-KLA secure if for any QPT adversary $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}SKL},\mathcal{A},q}(\lambda)$ *is negligible. We also define q-bounded distinguishing key Sel-IND-KLA security analogously by enforcing the adversary to output its target x^* at the beginning of the game.*

We emphasize that while the number of distinguishing keys that the adversary can obtain in the game is bounded by a fixed polynomial, the number of non-distinguishing keys (i.e., keys for y with $R(x^*,y)=0$) can be unbounded.

6.2 1-Bounded Distinguishing Key Construction

We construct an ABE-SKL scheme $\mathsf{1ABE}=(\mathsf{Setup},\mathcal{KG},\mathsf{Enc},\mathcal{Dec},\mathcal{Vrfy})$ for relation $R:\mathcal{X}\times\mathcal{Y}\to\{0,1\}$ with 1-bounded distinguishing key Ada-IND-KLA/Sel-IND-KLA security whose message space is $\{0,1\}^\ell$ by using the following building blocks.

- IND-KLA secure PKE-SKL $\mathsf{SKL}.(\mathcal{KG},\mathsf{Enc},\mathcal{Dec},\mathcal{Vrfy})$. Without loss of generality, we assume that $\mathsf{skl.ek}\in\{0,1\}^{\ell_{\mathsf{ek}}}$ and the randomness space used by $\mathsf{SKL.Enc}$ is $\{0,1\}^{\ell_{\mathsf{rand}}}$ for some $\ell_{\mathsf{ek}}(\lambda)$ and $\ell_{\mathsf{rand}}(\lambda)$. We also assume that the message space of SKL is $\{0,1\}^\ell$.
- Adaptively/Selectively secure ABE $\mathsf{ABE}.(\mathsf{Setup},\mathsf{KG},\mathsf{Enc},\mathsf{Dec})$ for relation R with message space $\{0,1\}^\lambda$.
- A garbling scheme $\mathsf{GC}=(\mathsf{Grbl},\mathsf{GCEval})$. Without loss of generality, we assume that the labels of GC are in $\{0,1\}^\lambda$.

$\mathsf{Setup}(1^\lambda)$**:**

- For $i\in[\ell_{\mathsf{ek}}]$ and $b\in\{0,1\}$, run $(\mathsf{abe.pk}_{i,b},\mathsf{abe.msk}_{i,b})\leftarrow\mathsf{ABE.Setup}(1^\lambda)$.

- Output $(\mathsf{pk}, \mathsf{msk}) := (\{\mathsf{abe.pk}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}}, \{\mathsf{abe.msk}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}})$.

$\mathcal{KG}(\mathsf{msk}, y)$**:**

- Generate $(\mathsf{skl.ek}, \mathsf{skl}.d\mathcal{K}, \mathsf{skl.vk}) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$.
- Run $\mathsf{abe.sk}_i \leftarrow \mathsf{ABE.KG}(\mathsf{ABE.msk}_{i,\mathsf{skl.ek}[i]}, y)$ for $i \in [\ell_{\mathsf{ek}}]$, where $\mathsf{skl.ek}[i]$ denotes the i-th bit of the binary string skl.ek.
- Output $us\mathcal{K} := (\{\mathsf{abe.sk}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \mathsf{skl.ek}, \mathsf{skl}.d\mathcal{K})$ and $\mathsf{vk} := \mathsf{skl.vk}$.

$\mathsf{Enc}(\mathsf{pk}, x, m)$**:**

- Choose $\mathsf{R} \leftarrow \{0,1\}^{\ell_{\mathsf{rand}}}$.
- Construct circuit $E[m, \mathsf{R}]$, which is a circuit that takes as input an encryption key skl.ek of SKL and outputs $\mathsf{SKL.Enc}(\mathsf{skl.ek}, m; \mathsf{R})$.
- Compute $(\{\mathsf{lab}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}}, \widetilde{E}) \leftarrow \mathsf{Grbl}(1^\lambda, E[m, \mathsf{R}])$.
- Run $\mathsf{abe.ct}_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}_{i,b}, x, \mathsf{lab}_{i,b})$ for $i \in [\ell_{\mathsf{ek}}]$ and $b \in \{0,1\}$.
- Output $\mathsf{ct} := (\{\mathsf{abe.ct}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}}, \widetilde{E})$.

$\mathcal{D}ec(us\mathcal{K}, x, \mathsf{ct})$**:**

- Parse $us\mathcal{K} = (\{\mathsf{abe.sk}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \mathsf{skl.ek}, \mathsf{skl}.d\mathcal{K})$ and $\mathsf{ct} = (\{\mathsf{abe.ct}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}}, \widetilde{E})$.
- Compute $\mathsf{lab}_i \leftarrow \mathsf{ABE.Dec}(\mathsf{ABE.sk}_i, x, \mathsf{abe.ct}_{i,\mathsf{skl.ek}[i]})$ for $i \in [\ell_{\mathsf{ek}}]$.
- Compute $\mathsf{skl.ct} = \mathsf{GCEval}(\widetilde{E}, \{\mathsf{lab}_i\}_{i\in[\ell_{\mathsf{ek}}]})$.
- Compute and output $m' \leftarrow \mathsf{SKL}.\mathcal{D}ec(\mathsf{skl}.d\mathcal{K}, \mathsf{skl.ct})$.

$\mathcal{V}rfy(\mathsf{vk}, us\mathcal{K}')$**:**

- Parse $\mathsf{vk} = \mathsf{skl.vk}$ and $us\mathcal{K}' = (\{\mathsf{abe.sk}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \mathsf{skl.ek}', \mathsf{skl}.d\mathcal{K}')$.
- Compute and output $\mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}, \mathsf{skl}.d\mathcal{K}')$.

We show that the scheme satisfies decryption correctness. To see this, we first observe that the decryption algorithm correctly recovers labels of $\widetilde{E}$ corresponding to the input skl.ek by the correctness of ABE. Therefore, skl.ct recovered by the garbled circuit evaluation equals to $\mathsf{SKL.Enc}(\mathsf{skl.ek}, m; \mathsf{R})$ by the correctness of GC. Then, the message m is recovered in the last step by the correctness of SKL. We can also see that the verification correctness follows from that of SKL.

Theorem 6.6. *If* ABE *is adaptively (resp., selectively) secure,* GC *is secure, and* SKL *is IND-KLA secure, then* 1ABE *above is* 1*-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) secure.*

Proof of Theorem 6.6. Here, we first focus on the proof for the case of Ada-IND-KLA and later mention the necessary modifications for the case of Sel-IND-KLA. Let Q be the upper bound on the number of key queries to $O_{\mathcal{KG}}$ before the challenge phase. We define a sequence of hybrid games.

Hyb_0**:** This is the same as $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{1ABE},\mathcal{A},1}(1^\lambda, 0)$. More specifically, it is as follows.

1. The challenger generates $(\mathsf{abe.pk}_{i,b}, \mathsf{abe.msk}_{i,b}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ for $i \in [\ell_{\mathsf{ek}}]$ and $b \in \{0,1\}$ and sends $\mathsf{pk} := \{\mathsf{abe.pk}_{i,b}\}_{i,b}$ to the adversary $\mathcal{A}$. The challenger then initializes the list $L_{\mathcal{KG}}$ to be an empty set. $\mathcal{A}$ can access the following oracles.

   $O_{\mathcal{KG}}(y^{(j)})$**:** Given the j-th query $y^{(j)}$ with $j \in [Q]$, if there is an entry of the form $(y^{(j)}, \mathsf{vk}, V)$, it outputs $\bot$. Otherwise, it generates $(\mathsf{skl.ek}^{(j)}, \mathsf{skl}.d\mathcal{K}^{(j)}, \mathsf{skl.vk}^{(j)}) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$ and $\mathsf{abe.sk}_i^{(j)} \leftarrow \mathsf{ABE.KG}(\mathsf{abe.msk}_{i,\mathsf{skl.ek}^{(j)}[i]}, y^{(j)})$ for $i \in [\ell_{\mathsf{ek}}]$, where $\mathsf{skl.ek}^{(j)}[i]$ is the i-th bit of the binary string $\mathsf{skl.ek}^{(j)}$. It then sends $us\mathcal{K}^{(j)} := (\{\mathsf{abe.sk}_i^{(j)}\}_i, \mathsf{skl}.d\mathcal{K}^{(j)})$ to $\mathcal{A}$ and adds $(y^{(j)}, \mathsf{skl.vk}^{(j)}, \bot)$ to $L_{\mathcal{KG}}$.

$O_{\mathcal{V}rfy}(y, usk')$: Given (y, usk'), it finds an entry (y, vk, V) from $L_{\mathcal{KG}}$ and parse $usk' = (\{\mathsf{abe.sk}'_i\}_i, \mathsf{skl}.usk')$. (If there is no such entry, it returns $\perp$.) It then parses $\mathsf{vk} = \mathsf{skl.vk}$ and returns $d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}, \mathsf{skl}.dk')$ to $\mathcal{A}$. It finally updates the entry into (y, vk, d) if $V = \perp$.

2. When $\mathcal{A}$ sends (x^*, m_0, m_1) to the challenger, the challenger checks whether there is at most one entry (y, vk, V) in $L_{\mathcal{KG}}$ such that $R(x^*, y) = 1$ and for that entry $V = \top$ holds. If so, the challenger generates $(\{\mathsf{lab}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}}, \widetilde{E}) \leftarrow \mathsf{Grbl}(1^\lambda, E[m_0, \mathsf{R}])$ and computes $\mathsf{abe.ct}_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}_{i,b}, x^*, \mathsf{lab}_{i,b})$ for $i \in [\ell_{\mathsf{ek}}]$ and $b \in \{0,1\}$. It then sends $\mathsf{ct}^* := (\{\mathsf{abe.ct}_{i,b}\}_{i,b}, \widetilde{E})$ to $\mathcal{A}$. Otherwise (i.e., if there are multiple entries with $R(x^*, y) = 1$ or if there is an entry with $R(x^*, y) = 1$ and $V = \perp$), it aborts the game and outputs 0.
3. $\mathcal{A}$ continues to make queries to $O_{\mathcal{KG}}(\cdot)$ and $O_{\mathcal{V}rfy}(\cdot,\cdot)$. However, $\mathcal{A}$ is not allowed to send a key attribute y such that $R(x^*, y) = 1$ to $O_{\mathcal{KG}}$.
4. $\mathcal{A}$ outputs a guess coin' for coin. The challenger outputs coin' as the final output of the experiment.

Hyb_1: This game is the same as Hyb_0 except that the challenger chooses random $\tilde{j} \leftarrow [Q]$ at the beginning of the game. Then, right before it computes the challenge ciphertext, the challenger finds an index $j^* \in [Q]$ such that $R(x^*, y^{(j^*)}) = 1$. If there is no such a query, we define $j^* := 1$.[20] The challenger then checks whether $\tilde{j} = j^*$. If so, the challenger continues the game until $\mathcal{A}$ outputs its guess. Otherwise, it aborts the game and outputs 0 as the outcome of the game.

Since the choice of $\tilde{j}$ is independent from the view of $\mathcal{A}$ and the outcome of the game is 1 only when $\tilde{j} = j^*$, we can easily see that $\Pr[\mathsf{Hyb}_1 = 1] = \Pr[\mathsf{Hyb}_0 = 1]/Q$.

Hyb_2: This game is the same as Hyb_1 except for the way $\{\mathsf{abe.ct}_{i,b}\}_{i,b}$ is generated. Namely, we generate $\mathsf{abe.ct}_{i,b}$ as $\mathsf{abe.ct}_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}_{i,b}, \mathsf{lab}_{i,\mathsf{skl.ek}^{(j^*)}[i]})$ for $i \in [\ell_{\mathsf{ek}}]$ and $b \in \{0,1\}$.

We observe that the labels being encrypted are changed only for positions of the form $(i, 1 \oplus \mathsf{lab}_{i,\mathsf{skl.ek}^{(j^*)}[i]})$. The adversary $\mathcal{A}$ cannot notice the change since it is not given any secret key that can decrypt the ABE ciphertexts for these positions. To check this, recall that there is at most one index j^* such that $R(x^*, y^{(j^*)}) = 1$ and for the corresponding key query, the adversary is given ABE secret keys for positions of the form $(i, \mathsf{lab}_{i,\mathsf{skl.ek}^{(j^*)}[i]})$, but not for $(i, \mathsf{lab}_{i,1\oplus\mathsf{skl.ek}^{(j^*)}[i]})$. Hence, we obtain $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| = \mathsf{negl}(\lambda)$ by the adaptive security of ABE. See Lemma 6.7 for the detail.

Hyb_3: This game is the same as Hyb_2 except for the way ct^* is generated. In particular, to generate ct^*, we first run $(\{\mathsf{lab}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \widetilde{E}) \leftarrow \mathsf{Sim.GC}(1^\lambda, \mathsf{SKL.Enc}(\mathsf{skl.ek}^{(j^*)}, m_0; \mathsf{R}))$ and then compute $\mathsf{abe.ct}_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}_{i,b}, \mathsf{lab}_i)$ for $i \in [\ell_{\mathsf{ek}}]$ and $b \in \{0,1\}$.

We claim that this game is indistinguishable from the previous one. To see this, it suffices to show that $(\{\mathsf{lab}_{i,\mathsf{skl.ek}^{(j^*)}[i]}\}_i, \widetilde{E})$ computed by $(\{\mathsf{lab}_{i,b}\}_{i,b}, \widetilde{E}) \leftarrow \mathsf{Grbl}(1^\lambda, E[m_0, \mathsf{R}])$ and $(\{\mathsf{lab}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \widetilde{E})$ computed by $(\{\mathsf{lab}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \widetilde{E}) \leftarrow \mathsf{Sim.GC}(1^\lambda, \mathsf{SKL.Enc}(\mathsf{skl.ek}^{(j^*)}, m_0; \mathsf{R}))$ are computationally indistinguishable. This immediately follows from the security of the garbled circuit, since we have

$$E[m_0, \mathsf{R}](\mathsf{skl.ek}^{(j^*)}) = \mathsf{SKL.Enc}(\mathsf{skl.ek}^{(j^*)}, m_0; \mathsf{R})$$

by the definition of E. Hence, we obtain $|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| = \mathsf{negl}(\lambda)$.

Hyb_4: This game is the same as Hyb_3 except that the challenger chooses $(\{\mathsf{lab}_i\}_i, \widetilde{E})$ by $(\{\mathsf{lab}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \widetilde{E}) \leftarrow \mathsf{Sim.GC}(1^\lambda, \mathsf{SKL.Enc}(\mathsf{skl.ek}^{(j^*)}, m_1; \mathsf{R}))$ instead of $(\{\mathsf{lab}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \widetilde{E}) \leftarrow \mathsf{Sim.GC}(1^\lambda, \mathsf{SKL.Enc}(\mathsf{skl.ek}^{(j^*)}, m_0; \mathsf{R}))$.

To show that $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| = \mathsf{negl}(\lambda)$, it suffices to show that $\mathsf{SKL.Enc}(\mathsf{skl.ek}^{(j^*)}, m_0; \mathsf{R})$ is indistinguishable from $\mathsf{SKL.Enc}(\mathsf{skl.ek}^{(j^*)}, m_1; \mathsf{R})$ for $\mathcal{A}$, if it makes $O_{\mathcal{V}rfy}$ output $\top$ on input $(y^{(j^*)}, usk')$

[20] Note that if there are multiple indices j^* satisfying the above, the challenger aborts and outputs 0 as specified in the previous game. Therefore, there is at most one such j^*.

for some usk' before the challenge ciphertext is given to $\mathcal{A}$. The indistinguishability follows from the security of SKL, since the fact that $\mathcal{A}$ passes the verification $O_{\mathcal{V}rfy}$ implies that $\mathcal{A}$ submitted $\mathsf{skl}.dk'$ such that $\mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}^{(j^*)}, \mathsf{skl}.dk') = \top$ before it is given the challenge ciphertext and therefore it has no longer the ability to decrypt the ciphertext. To turn this intuition into a formal reduction, we have to embed the public key of SKL into the answer to the j^*-th key generation query. Since the reduction algorithm does not know j^* until $\mathcal{A}$ submits (x^*, m_0, m_1), it can only guess it. The change in Hyb_1 is introduced in order to incorporate the guess into the game so that the reduction is possible. We refer to Lemma 6.8 for the formal proof for $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| = \mathsf{negl}(\lambda)$.

Hyb_5**:** This is the same as $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{1ABE},\mathcal{A},1}(1^\lambda, 1)$.

From the above discussion, we have

$$\begin{aligned} |Q\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_0 = 1]| &= Q|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \\ &\leq Q\sum_{i\in[0,2]} |\Pr[\mathsf{Hyb}_{i+1} = 1] - \Pr[\mathsf{Hyb}_i = 1]| \leq \mathsf{negl}(\lambda). \end{aligned} \tag{7}$$

We then observe that Hyb_5 (resp., Hyb_4) is the same as Hyb_0 (resp., Hyb_3) except that m_1 is used for the encryption instead of m_0. Therefore, we obtain $|Q\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_5 = 1]| \leq \mathsf{negl}(\lambda)$ analogously to Eq. (7) by considering similar sequence of the games with m_0 being replaced by m_1 in reverse order. We therefore have

$$\begin{aligned} &\left|\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{1ABE},\mathcal{A},1}(1^\lambda, 0) - \mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{1ABE},\mathcal{A},1}(1^\lambda, 1)\right| \\ &= |\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_5 = 1]| \\ &\leq |\Pr[\mathsf{Hyb}_0 = 1] - Q\Pr[\mathsf{Hyb}_3 = 1]| + Q\cdot|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| + |\Pr[\mathsf{Hyb}_5 = 1] - Q\Pr[\mathsf{Hyb}_4 = 1]| \\ &\leq \mathsf{negl}(\lambda) \end{aligned}$$

as desired. It remains to prove Lemmata 6.7 and 6.8.

Lemma 6.7. $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| = \mathsf{negl}(\lambda)$ *if* ABE *is adaptively secure.*

Proof. This can be reduced to the adaptive security of ABE by a standard hybrid argument where we modify the way of generating $\mathsf{ABE.ct}_{i,1\oplus\mathsf{skl.ek}^{(j^*)}[i]}$ for each $i \in [\ell_{\mathsf{ek}}]$ one by one. More precisely, the reduction works as follows.

We define additional hybrids $\mathsf{Hyb}_{1.k}$ for $k \in [\ell_{\mathsf{ek}}]$ as follows.

$\mathsf{Hyb}_{1.k}$**:** This is identical to Hyb_1 except that $\mathsf{abe.ct}_{i,1\oplus\mathsf{skl.ek}^{(j^*)}[i]}$ is generated as

$$\mathsf{abe.ct}_{i,1\oplus\mathsf{skl.ek}^{(j^*)}[i]} \leftarrow \begin{cases} \mathsf{ABE.Enc}(\mathsf{abe.pk}_{i,1\oplus\mathsf{skl.ek}^{(j^*)}[i]}, \mathsf{lab}_{i,\mathsf{skl.ek}^{(j^*)}[i]}) & i < k \\ \mathsf{ABE.Enc}(\mathsf{abe.pk}_{i,1\oplus\mathsf{skl.ek}^{(j^*)}[i]}, \mathsf{lab}_{i,1\oplus\mathsf{skl.ek}^{(j^*)}[i]}) & i \geq k \end{cases} \tag{8}$$

for $i \in [\lambda]$.

Clearly, we have $\mathsf{Hyb}_1 = \mathsf{Hyb}_{1.1}$ and $\mathsf{Hyb}_2 = \mathsf{Hyb}_{1.\ell_{\mathsf{ek}}+1}$. Thus, it suffices to prove that $|\Pr[\mathsf{Hyb}_{1.k+1} = 1] - \Pr[\mathsf{Hyb}_{1.k} = 1]| = \mathsf{negl}(\lambda)$ for all $k \in [\ell_{\mathsf{ek}}]$. Remark that the only difference between $\mathsf{Hyb}_{1.k+1}$ and $\mathsf{Hyb}_{1.k}$ is the way of generating $\mathsf{abe.ct}_{k,1\oplus\mathsf{skl.ek}^{(j^*)}[i]}$. To show that $|\Pr[\mathsf{Hyb}_{1.k+1} = 1] - \Pr[\mathsf{Hyb}_{1.k} = 1]| = \mathsf{negl}(\lambda)$, we construct $\mathcal{B}$ against adaptive security of ABE as follows.

$\mathcal{B}(\mathsf{abe.pk})$**:** It works as follows.

1. It chooses $\tilde{j} \leftarrow [Q]$ and $(\mathsf{skl.ek}^{(j)}, \mathsf{skl}.dk^{(j)}, \mathsf{skl.vk}^{(j)}) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$ for $j \in [Q]$.
2. Generate $(\mathsf{abe.pk}_{i,b}, \mathsf{abe.msk}_{i,b}) \leftarrow \mathsf{ABE.KG}(1^\lambda)$ for $(i,b) \in [\ell_{\mathsf{ek}}] \times \{0,1\}\backslash\{(k, 1\oplus\mathsf{skl.ek}^{(\tilde{j})}[k])\}$. Set $\mathsf{abe.pk}_{k,1\oplus\mathsf{skl.ek}^{(\tilde{j})}[k]} := \mathsf{abe.pk}$ and send $\mathsf{pk} := \{\mathsf{abe.pk}_{i,b}\}_{i,b}$ to $\mathcal{A}$.

3. $\mathcal{B}$ initializes the list $L_{\mathcal{KG}}$ to be an empty set and simulates the following oracles for $\mathcal{A}$.

   $O_{\mathcal{KG}}(y^{(j)})$**:** Given the j-th query $y^{(j)}$ with $j \in [Q]$, if there is an entry of the form $(y^{(j)}, \mathsf{vk}, V)$, it outputs $\bot$. Otherwise, it generates $\mathsf{abe.sk}_i^{(j)} \leftarrow \mathsf{ABE.KG}(\mathsf{abe.msk}_{i,\mathsf{skl.ek}^{(j)}[i]}, y^{(j)})$ for $i \in [\ell_{\mathsf{ek}}]\backslash\{k\}$. To simulate $\mathsf{abe.sk}_k^{(j)}$, $\mathcal{B}$ preceeds as follows. If $\mathsf{skl.ek}^{(j)}[k] = 1 \oplus \mathsf{skl.ek}^{(\tilde{j})}[k]$, it queries $y^{(j)}$ to its challenger. The challenger runs

   $$\mathsf{abe.sk} \leftarrow \mathsf{ABE.KG}(\mathsf{abe.msk}, y^{(j)})$$

   and returns it to $\mathcal{B}$. $\mathcal{B}$ then sets $\mathsf{abe.sk}_k^{(j)} := \mathsf{abe.sk}$. Otherwise (i.e., if $\mathsf{skl.ek}^{(j)}[k] = \mathsf{skl.ek}^{(\tilde{j})}[k]$), it runs $\mathsf{abe.sk}_k^{(j)} \leftarrow \mathsf{ABE.KG}(\mathsf{abe.msk}_{k,\mathsf{skl.ek}^{(j)}[k]}, y^{(j)})$. It then sends $us\mathcal{K}^{(j)} := (\{\mathsf{abe.sk}_i^{(j)}\}_i, \mathsf{skl}.d\mathcal{K}^{(j)})$ to $\mathcal{A}$ and adds $(y^{(j)}, \mathsf{skl.vk}^{(j)}, \bot)$ to $L_{\mathcal{KG}}$.

   $O_{\mathcal{V}rfy}(y, us\mathcal{K}')$**:** Given $(y, us\mathcal{K}')$, it finds an entry (y, vk, V) from $L_{\mathcal{KG}}$ and parse $us\mathcal{K}' = (\{\mathsf{abe.sk}_i'\}_i, \mathsf{skl}.us\mathcal{K}')$. (If there is no such entry, it returns $\bot$.) It then parses $\mathsf{vk} = \mathsf{skl.vk}$ and returns $d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}, \mathsf{skl}.d\mathcal{K}')$ to $\mathcal{A}$. It finally updates the entry into (y, vk, d) if $V = \bot$.

4. When $\mathcal{A}$ sends (x^*, m_0, m_1) to the challenger, $\mathcal{B}$ checks whether there are multiple entries (y, vk, V) in $L_{\mathcal{KG}}$ such that $R(x^*, y) = 1$ or there is an entry (y, vk, V) in $L_{\mathcal{KG}}$ with $R(x^*, y) = 1$ and $V = \top$. If so, $\mathcal{B}$ aborts the game and outputs 0 as its guess. Otherwise, $\mathcal{B}$ defines $j^* \in [Q]$ as in Hyb_1. It then aborts and outputs 0 if $j^* \neq \tilde{j}$. Otherwise, $\mathcal{B}$ computes ct^* as follows. It first chooses $\mathsf{R} \leftarrow \{0,1\}^{\ell_{\mathsf{rand}}}$ and computes $(\{\mathsf{lab}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}}, \widetilde{E}) \leftarrow \mathsf{Grbl}(1^\lambda, E[m_0, \mathsf{R}])$. It then computes $\mathsf{abe.ct}_{i,b}$ for $(i,b) \in [\ell_{\mathsf{ek}}] \times \{0,1\}\backslash\{(k, 1 \oplus \mathsf{skl.ek}^{(j^*)}[k])\}$ as in Equation (8). $\mathcal{B}$ then submits $(\mathsf{lab}_{k,\mathsf{skl.ek}^{(j^*)}[k]}, \mathsf{lab}_{k,1\oplus\mathsf{skl.ek}^{(j^*)}[k]})$ to its challenger. Then, the challenger runs

   $$\mathsf{abe.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, \mathsf{lab}_{\overline{\mathsf{coin}}\oplus\mathsf{skl.ek}^{(j^*)}[k]})$$

   and gives $\mathsf{abe.ct}$ to $\mathcal{B}$, where $\overline{\mathsf{coin}} \in \{0,1\}$ is the coin chosen by the challenger. Then, $\mathcal{B}$ sets $\mathsf{abe.ct}_{k,1\oplus\mathsf{skl.ek}^{(j^*)}[k]} := \mathsf{abe.ct}$ and gives $\mathsf{ct}^* := (\{\mathsf{abe.ct}_{i,b}\}_{i,b}, \widetilde{E})$ to $\mathcal{A}$.

5. $\mathcal{A}$ then continues to make queries to $O_{\mathcal{KG}}(\cdot)$ and $O_{\mathcal{V}rfy}(\cdot,\cdot)$. $\mathcal{B}$ answers the queries in the same manner as before the challenge query.

6. $\mathcal{A}$ finally outputs its guess. $\mathcal{B}$ outputs the same bit as its guess.

We first argue that $\mathcal{B}$ does not make any prohibited key query. To see this, we first observe that for every key query y that $\mathcal{B}$ makes, there exists j such that $y = y^{(j)}$. We then observe that $R(x^*, y^{(j)}) = 0$ for $j \neq j^*$ and $\mathcal{B}$ does not make a key query for $y^{(j^*)}$ in the above simulation.

We have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{ada\text{-}ind}}_{\mathsf{ABE},\mathcal{B}}(\lambda) &= 2\left|\Pr\left[\ \mathcal{B} \text{ outputs } \overline{\mathsf{coin}}\ \right] - \frac{1}{2}\right| \\
&= \left|\Pr\left[\ \mathcal{B} \text{ outputs } 1\ |\overline{\mathsf{coin}} = 0\right] - \Pr\left[\ \mathcal{B} \text{ outputs } 1\ |\overline{\mathsf{coin}} = 1\right]\right| \\
&= |\Pr[\mathsf{Hyb}_{1.k+1} = 1] - \Pr[\mathsf{Hyb}_{1.k} = 1]|
\end{aligned}$$

where the probabilities are taken over the randomness used in the respective games. Thus, $|\Pr[\mathsf{Hyb}_{1.k+1} = 1] - \Pr[\mathsf{Hyb}_{1.k} = 1]| = \mathsf{negl}(\lambda)$ by the adaptive security of ABE. This completes the proof of Lemma 6.7.

□

Lemma 6.8. $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| = \mathsf{negl}(\lambda)$ *if* SKL *is IND-KLA secure.*

Proof. This can be reduced to the IND-KLA security security of SKL. To do so, we construct an adversary $\mathcal{B}$ against IND-KLA security of the scheme with advantage $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]|$ as follows.

$\mathcal{B}(\mathsf{skl.ek}, \mathsf{skl}.d\mathcal{K})$**:** It works as follows.

1. It chooses $\tilde{j} \leftarrow [Q]$ and $(\mathsf{skl.ek}^{(j)}, \mathsf{skl.}d\mathcal{K}^{(j)}, \mathsf{skl.vk}^{(j)}) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$ for $j \in [Q]\backslash\{\tilde{j}\}$. It then sets $(\mathsf{skl.ek}^{(\tilde{j})}, \mathsf{skl.}d\mathcal{K}^{(\tilde{j})}) := (\mathsf{skl.ek}, \mathsf{skl.}d\mathcal{K})$. It then generates $(\mathsf{abe.pk}_{i,b}, \mathsf{abe.msk}_{i,b}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ for $i \in [\ell_{\mathsf{ek}}]$ and $b \in \{0,1\}$ and sends $\mathsf{pk} := \{\mathsf{abe.pk}_{i,b}\}_{i,b}$ to the adversary $\mathcal{A}$.
2. $\mathcal{B}$ initializes the list $L_{\mathcal{KG}}$ to be an empty set and simulates the following oracles for $\mathcal{A}$.

   $O_{\mathcal{KG}}(y^{(j)})$**:** Given the j-th query $y^{(j)}$ with $j \in [Q]$, if there is an entry of the form $(y^{(j)}, \mathsf{vk}, V)$, it outputs $\bot$. Otherwise, it generates $\mathsf{abe.sk}_i^{(j)} \leftarrow \mathsf{ABE.KG}(\mathsf{abe.msk}_{i,\mathsf{skl.ek}^{(j)}[i]}, y^{(j)})$ for $i \in [\ell_{\mathsf{ek}}]$. It then returns $usk^{(j)} := (\{\mathsf{abe.sk}_i^{(j)}\}_i, \mathsf{skl.}d\mathcal{K}^{(j)})$ to $\mathcal{A}$ and adds $(y^{(j)}, \mathsf{skl.vk}^{(j)}, \bot)$ to $L_{\mathcal{KG}}$.

   $O_{\mathcal{V}rfy}(y, usk')$**:** Given (y, usk'), it finds an entry (y, vk, V) from $L_{\mathcal{KG}}$ and parses $usk' = (\{\mathsf{abe.sk}'_i\}_i, \mathsf{skl.}d\mathcal{K}')$. (If there is no such entry, it returns $\bot$.) If $y = y^{(j)}$ for $j \neq \tilde{j}$, $\mathcal{B}$ returns $d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}^{(j)}, \mathsf{skl.}d\mathcal{K}')$ to $\mathcal{A}$. Otherwise (i.e., if $y = y^{(\tilde{j})}$), $\mathcal{B}$ submits $\mathsf{skl.}d\mathcal{K}'$ to its verification oracle. Then,

   $$d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}, \mathsf{skl.}d\mathcal{K}')$$

   is computed and returned to $\mathcal{B}$. $\mathcal{B}$ then returns d to $\mathcal{A}$. It finally updates the entry into (y, vk, d) if $V = \bot$.
3. When $\mathcal{A}$ sends (x^*, m_0, m_1) to the challenger, $\mathcal{B}$ checks whether there are multiple entries (y, vk, V) in $L_{\mathcal{KG}}$ such that $R(x^*, y) = 1$ or there is an entry (y, vk, V) in $L_{\mathcal{KG}}$ with $R(x^*, y) = 1$ and $V = \top$. If so, $\mathcal{B}$ aborts the game and outputs 0 as its guess. Otherwise, $\mathcal{B}$ defines $j^* \in [Q]$ as in Hyb_1. It then aborts and outputs 0 if $j^* \neq \tilde{j}$. Otherwise, $\mathcal{B}$ computes ct^* as follows. It first submits (m_0, m_1) to its challenger. Then, the challenger runs
   $$\mathsf{skl.ct} \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}, m_{\overline{\mathsf{coin}}})$$
   and returns it to $\mathcal{B}$, where $\overline{\mathsf{coin}} \in \{0,1\}$ is the coin chosen by the challenger. $\mathcal{B}$ then runs $(\{\mathsf{lab}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \widetilde{E}) \leftarrow \mathsf{Sim.GC}(1^\lambda, \mathsf{skl.ct})$ and computes $\mathsf{abe.ct}_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}_{i,b}, \mathsf{lab}_i)$ for $i \in [\ell_{\mathsf{ek}}]$ and $b \in \{0,1\}$. Then, $\mathcal{B}$ sets $\mathsf{ct}^* := (\{\mathsf{abe.ct}_{i,b}\}_{i,b}, \widetilde{E})$ and gives it to $\mathcal{A}$.
4. $\mathcal{A}$ then continues to make queries to $O_{\mathcal{KG}}(\cdot)$ and $O_{\mathcal{V}rfy}(\cdot,\cdot)$. $\mathcal{B}$ answers the queries in the same manner as before the challenge query.
5. $\mathcal{A}$ finally outputs its guess. $\mathcal{B}$ outputs the same bit as its guess.

We then have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}kla}}_{\mathsf{SKL},\mathcal{B}}(\lambda) &= \left|\Pr\left[\ \mathcal{B} \text{ outputs } 1 \ |\overline{\mathsf{coin}} = 0\right] - \Pr\left[\ \mathcal{B} \text{ outputs } 1 \ |\overline{\mathsf{coin}} = 1\right]\right| \\ &= |\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]|\end{aligned}$$

where the probabilities are taken over the randomness used in the respective games. Thus, $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| = \mathsf{negl}(\lambda)$ by the security of SKL. This completes the proof of Lemma 6.8. □

This completes the proof of Theorem 6.6 for the case of adaptive security.

The proof for selective security. The statement for selective security can be obtained immediately by considering the same sequence of games as adaptive security case with natural adaptations. In particular, we modify the reduction algorithm in Lemma 6.7 so that it outputs x^* at the beginning of the game right after given x^* from $\mathcal{A}$.

An alternative option is to consider a simpler proof that is tailored to selective setting. This is possible because the proof obtained by adapting the adaptive setting to the selective setting includes a redundant step. In particular, we consider a sequence of games without Hyb_1. The reason why Hyb_1 is not necessary is that in the selective setting, the reduction algorithm obtains x^* at the beginning of the game and can use this information throughout the game. In particular, whenever $\mathcal{A}$ makes a key query $y^{(j)}$, the reduction algorithm can check whether $j^* = j$ holds or not by computing the value of $R(x^*, y^{(j)})$ and there is no need to guess it. By introducing this change, we can improve the reduction cost to be independent of Q. □

6.3 Q-Bounded Distinguishing Key Construction

We construct an ABE-SKL scheme $\mathsf{qABE} = (\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$ for relation $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ with q-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) security from an ABE-SKL scheme $\mathsf{1ABE} = \mathsf{1ABE}.(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$ for the same relation R with 1-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) security. We note that the construction here is essentially the same as [ISV+17], which converts a single collusion secure ABE scheme into a q-bounded collusion secure ABE. However, our proof is more complex reflecting the fact that the adversary is allowed to make unbounded number of key queries (though the number of distinguishing keys is bounded).

The following construction uses parameters $v := v(\lambda)$ and $w := w(\lambda)$. We will set the parameters in Theorem 6.9.

$\mathsf{Setup}(1^\lambda)$:

- For $i \in [v]$ and $j \in [w]$, run $(\mathsf{1abe.pk}_{i,j}, \mathsf{1abe.msk}_{i,j}) \leftarrow \mathsf{1ABE.Setup}(1^\lambda)$.
- Output $(\mathsf{pk}, \mathsf{msk}) := (\{\mathsf{1abe.pk}_{i,j}\}_{i\in[v],j\in[w]}, \{\mathsf{1abe.msk}_{i,j}\}_{i\in[v],j\in[w]})$.

$\mathcal{KG}(\mathsf{msk}, y)$:

- For $i \in [v]$, choose $j_i \leftarrow [w]$.
- Run $(\mathsf{1abe.vk}_i, \mathsf{1abe.}usk_i) \leftarrow \mathsf{1ABE.}\mathcal{KG}(\mathsf{1abe.msk}_{i,j_i}, y)$ for $i \in [v]$.
- Output $usk := \{j_i, \mathsf{1abe.}usk_i\}_{i\in[v]}$ and $\mathsf{vk} := \{\mathsf{1abe.vk}_i\}_{i\in[v]}$.

$\mathsf{Enc}(\mathsf{pk}, x, m)$:

- Choose $\mu_1, \ldots, \mu_{v-1} \leftarrow \{0,1\}^\ell$ and set $\mu_v := (\oplus_{i\in[v-1]}\mu_i) \oplus m$, where $\oplus$ denotes bit-wise XOR here.
- Run $\mathsf{1abe.ct}_{i,j} \leftarrow \mathsf{1ABE.Enc}(\mathsf{1abe.pk}_{i,j}, x, \mu_i)$ for $i \in [v]$ and $j \in [w]$.
- Output $\mathsf{ct} := \{\mathsf{1abe.ct}_{i,j}\}_{i\in[v],j\in[w]}$.

$\mathcal{Dec}(usk, x, \mathsf{ct})$:

- Parse $usk := \{j_i, \mathsf{1abe.}usk_i\}_{i\in[v]}$ and $\mathsf{ct} := \{\mathsf{1abe.ct}_{i,j}\}_{i\in[v],j\in[w]}$.
- Compute $\mu'_i \leftarrow \mathsf{1ABE.Dec}(\mathsf{1abe.}usk_i, x, \mathsf{1abe.ct}_{i,j_i})$ for $i \in [v]$.
- Compute and output $m' := \oplus_{i\in[v]}\mu'_i$.

$\mathcal{Vrfy}(\mathsf{vk}, usk')$:

- Parse $\mathsf{vk} = \{\mathsf{1abe.vk}_i\}_{i\in[v]}$ and $usk' := \{j_i, \mathsf{1abe.}usk'_i\}_{i\in[v]}$.
- Compute $d_i \leftarrow \mathsf{1ABE.}\mathcal{Vrfy}(\mathsf{1abe.vk}_i, \mathsf{1abe.}usk'_i)$ for $i \in [v]$.
- If $d_i = \top$ for all $i \in [v]$, output $\top$. Otherwise, output $\bot$.

It is straightforward to see that the decryption correctness and the verification correctness of the above scheme follow from those of 1ABE.

Theorem 6.9. *Assuming* 1ABE *is* 1*-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) secure,* qABE *is q-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) secure if we set the parameters as follows:*

- *For the adaptive case, we assume that the size of the ciphertext attribute space $|\mathcal{X}_\lambda|$ is bounded by $2^{n(\lambda)}$ for some polynomial function $n(\lambda)$. We then set $v = 2(\lambda + n)$ and $w = q^2$.*
- *For the selective case, we set $v = \lambda$ and $w = q^2$.*

Proof of Theorem 6.9. Here, we first focus on the proof for the case of q-bounded distinguishing key Ada-IND-KLA and later mention the difference for the case of q-bounded distinguishing key Sel-IND-KLA. We define a sequence of hybrid games.

Hyb_0**:** This is the same as $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{qABE},\mathcal{A},q}(1^\lambda,0)$. More specifically, it is as follows.

1. The challenger generates $(\mathsf{1abe.pk}_{i,j},\mathsf{1abe.msk}_{i,j}) \leftarrow \mathsf{1ABE.Setup}(1^\lambda)$ for $i \in [v]$ and $j \in [w]$ and sends $\mathsf{pk} := \{\mathsf{1abe.pk}_{i,j}\}_{i,j}$ to the adversary $\mathcal{A}$. The challenger then initializes the list $L_{\mathcal{KG}}$ to be an empty set. $\mathcal{A}$ can access the following oracles.

   $O_{\mathcal{KG}}(y^{(k)})$**:** Given the k-th query $y^{(k)}$ with $k \in [Q]$, if there is an entry of the form $(y^{(k)},\mathsf{vk},V)$, it outputs $\bot$. Otherwise, it chooses $j_i^{(k)} \leftarrow [w]$ for $i \in [v]$ and runs $(\mathsf{1abe.vk}_i^{(k)},\mathsf{1abe.}usk_i^{(k)}) \leftarrow \mathsf{1ABE.}\mathcal{KG}(\mathsf{1abe.msk}_{i,j_i},y^{(k)})$ for $i \in [v]$. It then returns $usk^{(k)} := \{j_i^{(k)},\mathsf{1abe.}usk_i^{(k)}\}_{i\in[v]}$ and $\mathsf{vk}^{(k)} := \{\mathsf{1abe.vk}_i^{(k)}\}_{i\in[v]}$ to $\mathcal{A}$ and adds $(y^{(k)},\mathsf{vk}^{(k)},\bot)$ to $L_{\mathcal{KG}}$.

   $O_{\mathcal{V}rfy}(y,usk')$**:** Given (y,usk'), it finds an entry (y,vk,V) from $L_{\mathcal{KG}}$ and parses $usk' = \{j_i,usk'_i\}_i$. (If there is no such entry, it returns $\bot$.) It then computes $d_i := \mathsf{1ABE.}\mathcal{V}rfy(\mathsf{1abe.vk}_i,usk'_i)$ for $i \in [v]$ and checks if $d_i = \top$ for all $i \in [v]$. If so, it returns $d := \top$ to $\mathcal{A}$. Otherwise, it returns $d := \bot$ to $\mathcal{A}$. It finally updates the entry into (y,vk,d) if $V = \bot$.

2. When $\mathcal{A}$ sends (x^*,m_0,m_1) to the challenger, the challenger computes the set $K_{x^*} := \{k \in [Q_1] : R(x^*,y^{(k)}) = 1\}$, where $Q_1 \le Q$ is the number of key queries made by $\mathcal{A}$ so far. If we have $V = \top$ for all entries of the form $(y^{(k)},\mathsf{vk},V)$ in $L_{\mathcal{KG}}$ with $k \in K_{x^*}$ and $|K_{x^*}| \le q$, the challenger chooses $\mu_1,\dots,\mu_{v-1} \leftarrow \{0,1\}^\ell$, sets $\mu_v := (\oplus_{i\in[v-1]}\mu_i) \oplus m_0$, and computes $\mathsf{1abe.ct}_{i,j} \leftarrow \mathsf{1ABE.Enc}(\mathsf{1abe.pk}_{i,j},x^*,\mu_i)$ for $i \in [v]$ and $j \in [w]$. It then sends $\mathsf{ct}^* := \{\mathsf{1abe.ct}_{i,j}\}_{i,j}$ to $\mathcal{A}$. Otherwise (i.e., if $|K_{x^*}| > q$ or if there is an entry of the form $(y^{(k)},\mathsf{vk},\bot)$ for some $k \in K_{x^*}$), it aborts the game and outputs 0.
3. $\mathcal{A}$ continues to make queries to $O_{\mathcal{KG}}(\cdot)$ and $O_{\mathcal{V}rfy}(\cdot,\cdot)$. However, $\mathcal{A}$ is not allowed to send a key attribute y such that $R(x^*,y) = 1$ to $O_{\mathcal{KG}}$.
4. $\mathcal{A}$ outputs a guess coin' for coin. The challenger outputs coin' as the final output of the experiment.

Hyb_1**:** This game is the same as Hyb_0 except for the way ct^* is generated. In particular, when $\mathcal{A}$ submits (x^*,m_0,m_1), the challenger aborts the game and outputs 0 as the outcome of the game if there is no i^* such that $\{j_{i^*}^{(k)}\}_{k\in K_{x^*}}$ are all distinct. Otherwise, the challenger continues the game as specified in Hyb_0 .

We observe that unless there is no such i^*, the game is the same as the previous one. We bound the probability of this occuring. Let us first consider the case where $\mathcal{A}$ fixes its target x^* at the beginning of the game (i.e., selective security setting). In this case, by simple probability calculation, we can show that the probability that i^* does not exist is exponentially small in the parameter v. However, in the adaptive case, the adversary can adaptively choose x^* dependent on the values of $\{j_i^{(k)}\}_{i\in[v],k\in[Q]}$ and the proof for the selective case no longer works. To deal with the added flexibility given to the adversary, we use the union bound over all $x \in \mathcal{X}$ and then use the above bound for each fixed x. This requires the parameter v to grow dependent on the size of $\log|\mathcal{X}_\lambda|$ so that the sum of the probabilities is still small enough even after taking the union bound. Based on the above discussion, we can prove $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| = \mathsf{negl}(\lambda)$. We refer to Lemma 6.10 for the detail.

Hyb_2**:** This game is the same as Hyb_1 except that the challenger chooses random $\tilde{i} \leftarrow [v]$ at the beginning of the game. Then, right before the challenger computes ct^*, it checks whether $\tilde{i} = i^*$, where i^* is the smallest index such that $\{j_{i^*}^{(k)}\}_{k\in K_{x^*}}$ are all distinct. [21] If so, the challenger continues the game until $\mathcal{A}$ outputs its guess. Otherwise, it aborts the game and outputs 0 as the outcome of the game.

Since the choice of $\tilde{i}$ is independent from the view of $\mathcal{A}$ and the outcome of the game is 1 only when $\tilde{i} = i^*$, we can easily see that $\Pr[\mathsf{Hyb}_2 = 1] = \Pr[\mathsf{Hyb}_1 = 1]/v$.

Hyb_3**:** This is the same as Hyb_2 except for how $\mu_1,\dots,\mu_v$ are generated. In particular, $\mathcal{A}$ first chooses $\mu_1,\dots,\mu_v \leftarrow \{0,1\}^\ell$ and discards μ_{i^*}. It then sets $\mu_{i^*} := (\oplus_{i\in[v]\setminus\{i^*\}}) \oplus m_0$. It can be easily seen that the distribution of $\mu_1,\dots,\mu_v$ is unchanged from the previous game and thus we have $\Pr[\mathsf{Hyb}_2 = 1] = \Pr[\mathsf{Hyb}_3 = 1]$.

[21] Note that i^* is not defined until $\mathcal{A}$ chooses x^*.

Hyb_4**:** This is the same as Hyb_3 except that μ_{i^*} is set as $\mu_{i^*} := (\oplus_{i\in[v]\setminus\{i^*\}}\mu_i)\oplus m_1$.

We claim that this change is not noticed by $\mathcal{A}$ by the security of the underlying 1ABE. To show this, we first observe that the game differs from the previous one only in how $\{\mathsf{1abe.ct}_{i^*,j}\}_{j\in[w]}$ are generated. We then change each plaintext encrypted in $\{\mathsf{1abe.ct}_{i^*,j}\}_j$ one by one by using the security of the underlying 1ABE. This is possible since for each 1ABE instance with index (i^*,j), $\mathcal{A}$ is given only at most one distinguishing key by the change we introduced in Hyb_1 and thus we can use the security of 1ABE for such instances. We therefore have $|\Pr[\mathsf{Hyb}_3=1]-\Pr[\mathsf{Hyb}_4=1]|=\mathsf{negl}(\lambda)$. We refer to Lemma 6.11 for the detail.

Hyb_5**:** This is the same as $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{qABE},\mathcal{A},q}(1^\lambda,1)$.

From the above discussion, we have

$$|v\Pr[\mathsf{Hyb}_3=1]-\Pr[\mathsf{Hyb}_0=1]|=|\Pr[\mathsf{Hyb}_1=1]-\Pr[\mathsf{Hyb}_0=1]|\le\mathsf{negl}(\lambda). \tag{9}$$

We then observe that Hyb_5 (resp., Hyb_4) is the same as Hyb_0 (resp., Hyb_3) except that m_1 is used for the encryption instead of m_0. Therefore, we obtain $|v\Pr[\mathsf{Hyb}_4=1]-\Pr[\mathsf{Hyb}_5=1]|\le\mathsf{negl}(\lambda)$ analogously to Eq. (9) by considering similar sequence of games with m_0 being replaced by m_1 in a reverse order. We therefore have

$$\begin{aligned}
&\left|\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{qABE},\mathcal{A},q}(1^\lambda,0)-\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{qABE},\mathcal{A},q}(1^\lambda,1)\right|\\
&=|\Pr[\mathsf{Hyb}_0=1]-\Pr[\mathsf{Hyb}_5=1]|\\
&\le|\Pr[\mathsf{Hyb}_0=1]-v\Pr[\mathsf{Hyb}_3=1]|+v\cdot|\Pr[\mathsf{Hyb}_3=1]-\Pr[\mathsf{Hyb}_4=1]|+|\Pr[\mathsf{Hyb}_5=1]-v\Pr[\mathsf{Hyb}_4=1]|\\
&\le\mathsf{negl}(\lambda)
\end{aligned}$$

as desired. It remains to prove Lemmata 6.10 and 6.11.

Lemma 6.10. $|\Pr[\mathsf{Hyb}_0=1]-\Pr[\mathsf{Hyb}_1=1]|=\mathsf{negl}(\lambda)$ *holds both for selective and adaptive settings.*

Proof. We first show the statement for the selective case. The proof for this case is the same as [ISV+17, Lemma 1], but we provide the proof here for completeness. In the selective case, the probability that $\{j_i^{(k)}\}_{k\in K_{x^*}}$ are not all distinct for some fixed i is

$$1-\frac{w(w-1)\cdots(w-q+1)}{w^q}\le 1-\left(1-\frac{q-1}{w}\right)^q.$$

Therefore, the probability that there is no i^* satisfying the requirement is at most

$$\left(1-\left(1-\frac{q-1}{w}\right)^q\right)^v$$

which is negligible when $v=\lambda$ and $w=q^2$ since

$$\left(1-\left(1-\frac{q-1}{w}\right)^q\right)^v\le\left(1-e^{-1}\right)^\lambda=2^{-O(\lambda)}.$$

We then consider the adaptive case. We have

$$\begin{aligned}
\Pr\left[\{j_i^{(k)}\}_{k\in K_{x^*}}\text{ are not all distinct}\right] &= \sum_{x\in\mathcal{X}_\lambda}\Pr\left[x^*=x\wedge\{j_i^{(k)}\}_{k\in K_x}\text{ are not all distinct}\right]\\
&\le \sum_{x\in\mathcal{X}_\lambda}\Pr\left[\{j_i^{(k)}\}_{k\in K_x}\text{ are not all distinct}\right]\\
&\le \sum_{x\in\mathcal{X}_\lambda}\left(1-\left(1-\frac{q-1}{w}\right)^q\right)^v\\
&\le |\mathcal{X}_\lambda|\left(1-e^{-1}\right)^v\\
&\le 2^{-\lambda},
\end{aligned}$$

where the probabilities are taken over all randomness used in the game. In the above, third line follows from the same analysis as the selective case and the forth and the fifth lines follow from our parameter setting. □

Lemma 6.11. *If* 1ABE *is* 1*-bounded distinguishing key Ada-IND-KLA,* $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| = \mathsf{negl}(\lambda)$.

Proof. This can be reduced to the 1-bounded distinguishing key Ada-IND-KLA security of 1ABE by a standard hybrid argument, where we modify the plaintext encrypted in $\mathsf{1abe.ct}_{i^*,j}$ for each $j \in [w]$ one by one. More precisely, the reduction works as follows.

We define additional hybrids $\mathsf{Hyb}_{3.k}$ for $k \in [w]$ as follows. In the following, let $\xi_b := (\oplus_{i\in[v]\setminus\{i^*\}}\mu_i) \oplus m_b$ for $b \in \{0,1\}$.

$\mathsf{Hyb}_{3.\tau}$**:** This is identical to Hyb_3 except that $\mathsf{1abe.ct}_{i^*,j}$ is generated as

$$\mathsf{1abe.ct}_{i^*,j} \leftarrow \begin{cases} \mathsf{1ABE.Enc}(\mathsf{1abe.pk}_{i^*,j}, \xi_1) & j < \tau \\ \mathsf{1ABE.Enc}(\mathsf{1abe.pk}_{i^*,j}, \xi_0) & j \geq \tau \end{cases} \tag{10}$$

for $j \in [\lambda]$.

Clearly, we have $\mathsf{Hyb}_3 = \mathsf{Hyb}_{3.1}$ and $\mathsf{Hyb}_4 = \mathsf{Hyb}_{3.w+1}$. Thus, it suffices to prove that $|\Pr[\mathsf{Hyb}_{3.\tau+1} = 1] - \Pr[\mathsf{Hyb}_{3.\tau} = 1]| = \mathsf{negl}(\lambda)$ for all $\tau \in [w]$. Remark that the only difference between $\mathsf{Hyb}_{3.\tau+1}$ and $\mathsf{Hyb}_{3.\tau}$ is the way of generating $\mathsf{1abe.ct}_{i^*,\tau}$. To show that $|\Pr[\mathsf{Hyb}_{3.\tau+1} = 1] - \Pr[\mathsf{Hyb}_{3.\tau} = 1]| = \mathsf{negl}(\lambda)$, we construct $\mathcal{B}$ against the security of 1ABE as follows.

$\mathcal{B}(\mathsf{1abe.pk})$**:** It works as follows.

1. It first chooses random $\tilde{i} \leftarrow [v]$.
2. The challenger generates $(\mathsf{1abe.pk}_{i,j}, \mathsf{1abe.msk}_{i,j}) \leftarrow \mathsf{1ABE.Setup}(1^\lambda)$ for $(i,j) \in ([v] \times [w])\setminus\{(\tilde{i},k)\}$. It then sets $\mathsf{1abe.pk}_{\tilde{i},k} := \mathsf{1abe.pk}$ and sends $\mathsf{pk} := \{\mathsf{1abe.pk}_{i,j}\}_{i,j}$ to the adversary $\mathcal{A}$. It then initializes the list $L_{\mathcal{KG}}$ to be an empty set. $\mathcal{B}$ then simulates the following oracles for $\mathcal{A}$.

   $O_{\mathcal{KG}}(y^{(k)})$**:** Given the k-th query $y^{(k)}$ with $k \in [Q]$ from $\mathcal{A}$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ if there is an entry of the form $(y^{(k)}, \mathsf{vk}, V)$. Otherwise, it chooses $j_i^{(k)} \leftarrow [w]$ for $i \in [v]$ and runs $(\mathsf{1abe.vk}_i^{(k)}, \mathsf{1abe.}usk_i^{(k)}) \leftarrow \mathsf{1ABE.}\mathcal{KG}(\mathsf{1ABE.msk}_{i,j_i}, y^{(k)})$ for $i \in [v]\setminus\{\tilde{i}\}$. If $j_{\tilde{i}}^{(k)} = \tau$, it sends $y^{(k)}$ to its key generation oracle and is given

   $$\mathsf{1abe.}usk \leftarrow \mathsf{1ABE.}\mathcal{KG}(\mathsf{1abe.msk}, y^{(k)}).$$

   Then, it sets $\mathsf{1abe.}usk_{\tilde{i}}^{(k)} := \mathsf{1abe.}usk$. Otherwise (i.e., if $j_{\tilde{i}}^{(k)} \neq \tau$), it runs $(\mathsf{1abe.vk}_{\tilde{i}}^{(k)}, \mathsf{1abe.}usk_{\tilde{i}}^{(k)}) \leftarrow \mathsf{ABE.}\mathcal{KG}(\mathsf{1ABE.msk}_{\tilde{i},j_{\tilde{i}}}, y^{(k)})$ by itself. Finally, $\mathcal{B}$ returns $usk^{(k)} := \{j_i^{(k)}, \mathsf{1abe.}usk_i^{(k)}\}_{i\in[v]}$ to $\mathcal{A}$ and adds $(y^{(k)}, \mathsf{vk}^{(k)}, \perp)$ to $L_{\mathcal{KG}}$, where $\mathsf{vk}^{(k)} := \{\mathsf{1abe.vk}_i^{(k)}\}_{i\in[v]}$.

   $O_{\mathcal{V}rfy}(y, usk')$**:** Given (y, usk'), it finds an entry (y, vk, V) from $L_{\mathcal{KG}}$ and parses $usk' = \{j_i, usk_i'\}_i$. (If there is no such entry, it returns $\perp$.) It then computes $d_i := \mathsf{1ABE.}\mathcal{V}rfy(\mathsf{1abe.vk}_i, usk_i')$ for $i \in [v]$. If $j_{\tilde{i}} = \tau$, $\mathcal{B}$ makes a query to its own verification oracle to obtain

   $$d_{\tilde{i}} := \mathsf{1ABE.}\mathcal{V}rfy(\mathsf{1abe.vk}, usk_{\tilde{i}}').$$

   Otherwise, $\mathcal{B}$ runs $d_{\tilde{i}} := \mathsf{1ABE.}\mathcal{V}rfy(\mathsf{1abe.vk}_{\tilde{i},\tau}, usk_{\tilde{i}}')$ by itself. Finally, it checks if $d_i = \top$ for all $i \in [v]$. If so, it returns $d := \top$ to $\mathcal{A}$. Otherwise, it returns $d := \perp$ to $\mathcal{A}$. It finally updates the entry into (y, vk, d) if $V = \perp$.
3. When $\mathcal{A}$ sends (x^*, m_0, m_1) to the challenger, $\mathcal{B}$ aborts and outputs 0 if either $|K_{x^*}| > q$ or there is an entry of the form $(y^{(k)}, \mathsf{vk}, \perp)$ for some $k \in K_{x^*}$. It also aborts and outputs 0 if $i^* \neq \tilde{i}$, which includes the case that there is no i^* satisfying the properties we defined in Hyb_1. Otherwise, it chooses

$\mu_1, \ldots, \mu_v \leftarrow \{0,1\}^\ell$ and sets $\xi_0 := (\oplus_{i\in[v]\setminus\{i^*\}}\mu_i) \oplus m_0$ and $\xi_1 := (\oplus_{i\in[v]\setminus\{i^*\}}\mu_i) \oplus m_1$. It then computes $\mathsf{1abe.ct}_{i,j} \leftarrow \mathsf{1ABE.Enc}(\mathsf{1abe.pk}_{i,j}, x^*, \mu_i)$ for $i \in [v]\setminus\{i^*\}$ and $j \in [w]$ and $\mathsf{1abe.ct}_{i^*,j}$ for $j \in [w]\setminus\{\tau\}$ as Equation (10). It then submits (ξ_0, ξ_1) to its challenger. Then,

$$\mathsf{1abe.ct} \leftarrow \mathsf{Enc}(\mathsf{1abe.pk}, \xi_{\overline{\mathsf{coin}}})$$

is run and 1abe.ct is returned to $\mathcal{B}$, where $\overline{\mathsf{coin}}$ is the random bit chosen by $\mathcal{B}$'s challenger. Finally, $\mathcal{B}$ sets $\mathsf{1abe.ct}_{i^*,\tau} := \mathsf{1abe.ct}$ and sends $\mathsf{ct}^* := \{\mathsf{1abe.ct}_{i,j}\}_{i,j}$ to $\mathcal{A}$.

4. $\mathcal{A}$ continues to make queries to $O_{\mathcal{KG}}(\cdot)$ and $O_{\mathcal{V}rfy}(\cdot,\cdot)$. However, $\mathcal{A}$ is not allowed to send a key attribute y such that $R(x^*, y) = 1$ to $O_{\mathcal{KG}}$.
5. $\mathcal{A}$ outputs a guess coin' for coin. The challenger outputs coin' as the final output of the experiment.

We first argue that $\mathcal{B}$ does not make more than two distinguishing key queries. This is because $\mathcal{B}$ aborts and outputs 0 before it makes a challenge query if there is no i^* with the required conditions. For such i^*, we have that $\{j_{i^*}^{(k)}\}_{k\in K_{x^*}}$ are all distinct and thus in particular, $\mathcal{B}$ needs to simulate only single distinguishing key for the (i^*, τ)-th instance, to which the reduction algorithm embeds the 1ABE instance.

We then have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{1ABE},\mathcal{B},1}(\lambda) &= \left|\Pr\left[\ \mathcal{B} \text{ outputs } 1 \mid \overline{\mathsf{coin}} = 0\right] - \Pr\left[\ \mathcal{B} \text{ outputs } 1 \mid \overline{\mathsf{coin}} = 1\right]\right| \\ &= |\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]|\end{aligned}$$

where the probabilities are taken over the randomness used in the respective games. Thus, $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| = \mathsf{negl}(\lambda)$ by the adaptive security of 1ABE. This completes the proof of Lemma 6.11.

□

This completes the proof of Theorem 6.9 for the case of adaptive security.

The proof for selective security. The proof for selective security can be obtained immediately by considering the same sequence of games as adaptive security case with natural adaptations. There are two main differences. The proof for Lemma 6.10 requires different parameters for selective and adaptive cases. We refer to the proof of the lemma for the detail. Another difference is that we modify the reduction algorithm in Lemma 6.11 so that it outputs x^* at the beginning of the game right after given x^* from $\mathcal{A}$. □

6.4 Instantiations

Here, we explain new schemes that can be obtained by applying the conversions that we showed in Sections 6.2 and 6.3 to existing IBE/ABE schemes. Our constructions are fully generic and can upgrade almost all ABE schemes[22] into the one with the security against key leasing attacks with the help of IND-KLA secure PKE-SKL scheme, which can be instantiated from any (post quantum) PKE. Here, we mention some instantiations, all of which are obtained from the standard LWE assumption.

- If we start from selectively secure ABE scheme for circuits [GVW13, BGG+14] and apply the conversions in Sections 6.2 and 6.3, we obtain an ABE-SKL scheme for circuits with q-bounded distinguishing key Sel-IND-KLA security for any $q = \mathrm{poly}(\lambda)$.
- If we start from adaptively secure ABE for inner products over the integer [KNYY20] and apply the conversions in Sections 6.2 and 6.3, we obtain an ABE-SKL scheme for the same predicate with q-bounded distinguishing key Ada-IND-KLA security for any $q = \mathrm{poly}(\lambda)$. We note that the conversion in Section 6.3 for adaptive security

[22] Our conversion in Section 6.3 for the adaptive security case poses the restriction that the size of the cipheretxt attribute space of the ABE should be bounded by $2^{\mathrm{poly}(\lambda)}$ for some polynomial $\mathrm{poly}(\lambda)$. This means that we cannot apply the conversion for adaptively secure ABE for DFA for example, since the ciphertext attribute is of unbounded length and there is no such bound for the size of the ciphertext attribute space. However, we do not know any concrete ABE scheme from standard assumptions for which we cannot apply our conversion.

case can be applied for the scheme, since the size of the ciphertext attribute space is bounded by $2^{\mathrm{poly}(\lambda)}$ for the primitive. Similar implications can be obtained for adaptively secure t-CNF formulae for $t = O(1)$ [Tsa19] and fuzzy IBE for small universe [KNYY20].

- If we start from adaptively (resp., selectively) secure IBE [ABB10, CHKP10] and apply the conversion in Section 6.2, we obtain IBE-SKL scheme with 1-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) security. We note that 1-bounded distinguishing key security for the case of IBE is a more natural security notion than that for the case of ABE with other relations since there is only one attribute that is eligible for decrypting a ciphertext in the case of IBE (i.e., the identity that is associated with the ciphertext), whereas there can be exponentially many such attributes in general.

7 Public-Key Functional Encryption with Secure Key Leasing

7.1 Definitions

Definition 7.1 (PKFE with Secure Key Leasing). *A PKFE-SKL scheme* PKFE-SKL *is a tuple of six algorithms* $(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Cert}, \mathsf{Vrfy})$. *Below, let* $\mathcal{X}$, $\mathcal{Y}$, *and* $\mathcal{F}$ *be the plaintext, output, and function spaces of* PKFE-SKL, *respectively.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{msk})$**:** *The setup algorithm takes a security parameter* 1^λ, *and outputs a public key* pk *and master secret key* msk.

$\mathcal{KG}(\mathsf{msk}, f) \to (\mathit{fsk}, \mathsf{vk})$**:** *The key generation algorithm takes a master secret key* msk *and a function* $f \in \mathcal{F}$, *and outputs a functional decryption key* fsk *and a verification key* vk.

$\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$**:** *The encryption algorithm takes a public key* pk *and a plaintext* $x \in \mathcal{X}$, *and outputs a ciphertext* ct.

$\mathcal{Dec}(\mathit{fsk}, \mathsf{ct}) \to \tilde{x}$**:** *The decryption algorithm takes a functional decryption key* fsk *and a ciphertext* ct, *and outputs a value* $\tilde{x}$.

$\mathcal{Vrfy}(\mathsf{vk}, \mathit{fsk}') \to \top / \bot$**:** *The verification algorithm takes a verification key* vk *and a quantum state* fsk', *and outputs* $\top$ *or* $\bot$.

Decryption correctness: *For every* $x \in \mathcal{X}$ *and* $f \in \mathcal{F}$, *we have*

$$\Pr\left[\mathcal{Dec}(\mathit{fsk}, \mathsf{ct}) = f(x) \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathit{fsk}, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, f) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Verification correctness: *For every* $f \in \mathcal{F}$, *we have*

$$\Pr\left[\mathcal{Vrfy}(\mathsf{vk}, \mathit{fsk}) = \top \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathit{fsk}, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, f)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Remark 7.2. Although Kitagawa and Nishimaki [KN22a] require SKFE-SKL to have classical certificate generation algorithm for deletion, we do not since it is optional. If there exists a PKE-SKL scheme that has a classical certificate generation algorithm, our PKFE-SKL scheme in Section 7.2 also has a classical certificate generation algorithm.

Definition 7.3 (Adaptive Indistinguishability against Key Leasing Attacks). *We say that a PKFE-SKL scheme* PKFE-SKL *for* $\mathcal{X}, \mathcal{Y}$, *and* $\mathcal{F}$ *is an adaptively indistinguishable secure against key leasing attacks (Ada-IND-KLA), if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathcal{A},\mathsf{PKFE\text{-}SKL}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger:*

1. *At the beginning, the challenger runs* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$. *Throughout the experiment,* $\mathcal{A}$ *can access the following oracles.*

$O_{\mathcal{KG}}(f)$: *Given f, it finds an entry (f, vk, V) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\perp$. Otherwise, it generates $(fsk, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, f)$, sends fsk to $\mathcal{A}$, and adds $(f, \mathsf{vk}, \perp)$ to $L_{\mathcal{KG}}$.*

$O_{\mathcal{Vrfy}}(f, fsk')$: *Given (f, fsk'), it finds an entry (f, vk, V) from $L_{\mathcal{KG}}$. (If there is no such entry, it returns $\perp$.) It computes $d \leftarrow \mathcal{Vrfy}(\mathsf{vk}, fsk')$ and sends d to $\mathcal{A}$. If $V = \top$, it does not update $L_{\mathcal{KG}}$. Else if $V = \perp$, it updates the entry by setting $V := d$.*

2. *When $\mathcal{A}$ sends (x_0^*, x_1^*) to the challenger, the challenger checks if for any entry (f, vk, V) in $L_{\mathcal{KG}}$ such that $f(x_0^*) \neq f(x_1^*)$, it holds that $V = \top$. If so, the challenger generates $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x^*_{\mathsf{coin}})$ and sends ct^* to $\mathcal{A}$. Otherwise, the challenger outputs 0. Hereafter, $\mathcal{A}$ is not allowed to send a function f such that $f(x_0^*) \neq f(x_1^*)$ to $O_{\mathcal{KG}}$.*

3. *$\mathcal{A}$ outputs a guess coin' for coin. The challenger outputs coin' as the final output of the experiment.*

For any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{PKFE\text{-}SKL},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{PKFE\text{-}SKL},\mathcal{A}}(1^\lambda, 0) \to 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{PKFE\text{-}SKL},\mathcal{A}}(1^\lambda, 1) \to 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Remark 7.4. Definition 7.3 assumes that the adversary does not get more than one decryption key for the same f for simplification as Remark 6.3.

7.2 Constructions

We describe our PKFE-SKL scheme in this section. We construct a PKFE-SKL scheme $\mathsf{PKFE\text{-}SKL} = (\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$ by using the following building blocks.

- IND-KLA secure PKE-SKL $\mathsf{SKL} = \mathsf{SKL}.(\mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$.
- Adaptively secure PKFE $\mathsf{FE} = \mathsf{FE}.(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.
- Adaptively single-ciphertext function private SKFE $\mathsf{SKFE} = \mathsf{SKFE}.(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.
- Pseudorandom-secure SKE $\mathsf{SKE} = \mathsf{SKE}.(\mathsf{Enc}, \mathsf{Dec})$.
- Puncturable PRF $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{F}, \mathsf{Puncture})$.

We set $\ell_{\mathsf{pad}} := |\mathsf{skfe.ct}| - |x|$ and $\ell_{\mathsf{ske}} := |\mathsf{ske.ct}|$, where $|x|$ is the input length of PKFE-SKL, $|\mathsf{skfe.ct}|$ is the ciphertext length of SKFE, and $|\mathsf{ske.ct}|$ is the ciphertext length of SKE.

$\mathsf{Setup}(1^\lambda)$:

- Generate $(\mathsf{fe.pk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
- Output $(\mathsf{pk}, \mathsf{msk}) := (\mathsf{fe.pk}, \mathsf{fe.msk})$.

$\mathcal{KG}(\mathsf{msk}, f)$:

- Generate $(\mathsf{skl.ek}, \mathsf{skl}.sk, \mathsf{skl.vk}) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$.
- Choose $\mathsf{ske.ct} \leftarrow \{0,1\}^{\ell_{\mathsf{ske}}}$.
- Construct a circuit $W[f, \mathsf{skl.ek}, \mathsf{ske.ct}]$, which is described in Figure 1.
- Generate $\mathsf{fe.sk}_W \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f, \mathsf{skl.ek}, \mathsf{ske.ct}])$.
- Output $fsk := (\mathsf{fe.sk}_W, \mathsf{skl}.sk)$ and $\mathsf{vk} := \mathsf{skl.vk}$.

$\mathsf{Enc}(\mathsf{pk}, x)$:

- Choose $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Compute $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.pk}, (x \| 0^{\ell_{\mathsf{pad}}}, \perp, \mathsf{K}))$.

- Output $\mathsf{ct} := \mathsf{fe.ct}$.

$\mathcal{D}ec(\mathit{fsk}, \mathsf{ct})$:

- Parse $\mathit{fsk} = (\mathsf{fe.sk}, \mathsf{skl}.\mathit{sk})$ and $\mathsf{ct} = \mathsf{fe.ct}$.
- Compute $\mathsf{skl.ct} \leftarrow \mathsf{FE.Dec}(\mathsf{fe.sk}, \mathsf{fe.ct})$.
- Compute and output $y \leftarrow \mathsf{SKL}.\mathcal{D}ec(\mathsf{skl}.\mathit{sk}, \mathsf{skl.ct})$.

$\mathcal{V}rfy(\mathsf{vk}, \mathit{fsk}')$:

- Parse $\mathsf{vk} = \mathsf{skl.vk}$ and $\mathit{fsk}' = (\mathsf{fe.sk}', \mathsf{skl}.\mathit{sk}')$.
- Compute and output $\mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}, \mathsf{skl}.\mathit{sk}')$.

Function $W[f, \mathsf{skl.ek}, \mathsf{ske.ct}](x', \mathsf{ske.sk}, \mathsf{K})$

Constants: Function f, PKE-SKL encryption key skl.ek, SKE ciphertext ske.ct.

Input: Plaintext x', SKE key ske.sk, PRF key K.

1. If $\mathsf{ske.sk} = \bot$, do the following:
   - Parse $x' = x \| \overline{x}$ such that $|\overline{x}| = \ell_{\mathsf{pad}}$.
   - Compute and output $\mathsf{skl.ct} := \mathsf{SKL.Enc}(\mathsf{skl.ek}, f(x); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}))$.
2. If $\mathsf{ske.sk} \neq \bot$, do the following:
   - Compute $\mathsf{skfe.sk} \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct})$.
   - Compute and output $z := \mathsf{SKFE.Dec}(\mathsf{skfe.sk}, x')$.

Figure 1: The description of $W[f, \mathsf{skl.ek}, \mathsf{ske.ct}]$

Correctness. The decryption correctness of PKFE-SKL follows from the correctness of FE and the decryption correctness of SKL. The verification correcntess of PKFE-SKL follows from the verification correcntess of SKL.

7.3 Security Proofs

We prove the security of PKFE-SKL.

Theorem 7.5. *If* PKFE *is adaptively secure,* SKFE *is adaptively single-ciphertext function private,* PRF *is a secure punctured PRF, and* SKE *has the ciphertext pseudorandomness, then* PKFE-SKL *above is Ada-IND-KLA.*

Theorem 7.6. *If* PKFE *is q-bounded adaptively secure,* SKFE *is adaptively single-ciphertext function private,* PRF *is a secure punctured PRF, and* SKE *has the ciphertext pseudorandomness, then* PKFE-SKL *above is q-bounded Ada-IND-KLA.*

The proof of Theorem 7.6 is almost the same as that of Theorem 7.5. Hence, we focus on the proof of Theorem 7.5. We can also consider a simulation-based security for q-bounded security as Kitagawa and Nishimaki [KN22a] and believe that we can achieve it using a similar technique. However, it is out of scope of this work.

Proof of Theorem 7.5. In the proof, we embed an SKFE ciphertext $\mathsf{skfe.ct} \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x, \bot, \mathsf{K}, 0, \bot))$ into the challenge ciphertext. More specifically, we generate $\mathsf{fe.ct} \leftarrow \mathsf{PKFE.Enc}(\mathsf{fe.pk}, (\mathsf{skfe.ct}, \mathsf{ske.sk}, \bot))$ and $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T[f, \mathsf{skl.ek}]))$, where $T[f, \mathsf{skl.ek}]$ is described in Figure 2. By using this embedding, we can use the function privacy of SKFE and can alter both plaintexts and functions in the proof.

Let q be the total number of key queries to $O_{\mathcal{KG}}$. In the collusion-resistant setting, q is an unbounded polynomial. Note that even if q is an unbounded polynomial, we need only $\mathrm{poly}(\lambda)$ bits to describe q as an integer. We assume that the adversary does not send the same f to $O_{\mathcal{KG}}$ more than once without loss of generality. We define a sequence of hybrid games.

Function $T[f, \mathsf{skl.ek}](x, x_1, \mathsf{K}, j, \mathsf{skl.ct}^*)$

Constants: Function f, encryption key of PKE-SKL $\mathsf{skl.ek}$.

Input: Plaintext x and x_1, PRF key K, index j, a PKE-SKL ciphertext $\mathsf{skl.ct}^*$.[a]

1. Compute and output $\mathsf{skl.ct} := \mathsf{SKL.Enc}(\mathsf{skl.ek}, f(x); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}))$.

[a] Although inputs $(x_1, j, \mathsf{skl.ct}^*)$ are not used in this function, we need them in hybrid games later.

Figure 2: The description of $T[f, \mathsf{skl.ek}]$

Hyb_0**:** This is the same as $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{PKFE\text{-}SKL},\mathcal{A}}(1^\lambda, 0)$. More specifically, it is as follows.

1. The challenger generates $(\mathsf{fe.pk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and sends $\mathsf{pk} := \mathsf{fe.pk}$ to $\mathcal{A}$. $\mathcal{A}$ can access the following oracles.
   - $O_{\mathcal{KG}}(f_i)$**:** Given f_i, it generates $(\mathsf{skl.ek}_i, \mathsf{skl}.s\mathcal{K}_i, \mathsf{skl.vk}_i) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$, $\mathsf{ske.ct}_i \leftarrow \{0,1\}^{\ell_{\mathsf{ske}}}$, and $\mathsf{fe.sk}_{W,i} \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i])$, sends $fs\mathcal{K}_i := (\mathsf{fe.sk}_{W,i}, \mathsf{skl}.s\mathcal{K}_i)$ to $\mathcal{A}$, and adds $(f_i, \mathsf{vk}_i, \bot)$ to $L_{\mathcal{KG}}$.
   - $O_{\mathcal{V}rfy}(f_i, fs\mathcal{K}'_i)$**:** Given $(f_i, fs\mathcal{K}'_i)$, it finds an entry $(f_i, \mathsf{vk}_i, V_i)$ from $L_{\mathcal{KG}}$ and parse $fs\mathcal{K}'_i = (\mathsf{fe.sk}'_i, \mathsf{skl}.s\mathcal{K}'_i)$. (If there is no such entry, it returns $\bot$.) It returns $d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}, \mathsf{skl}.s\mathcal{K}')$. If $V_i = \top$, it does not update the entry. Otherwise, it updates the entry by setting $V_i := d$.
2. When $\mathcal{A}$ sends (x_0^*, x_1^*) to the challenger, the challenger checks if for any entry (f, vk, V) in $L_{\mathcal{KG}}$ such that $f(x_0^*) \neq f(x_1^*)$, it holds that $V = \top$. If so, the challenger generates $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.pk}, (x^*_{\mathsf{coin}} \| 0^{\ell_{\mathsf{pad}}}, \bot, \mathsf{K}))$ and sends $\mathsf{ct}^* := \mathsf{fe.ct}^*$ to $\mathcal{A}$. Otherwise, the challenger outputs 0. Hereafter, $\mathcal{A}$ is not allowed to send a function f such that $f(x_0^*) \neq f(x_1^*)$ to $O_{\mathcal{KG}}$.
3. $\mathcal{A}$ outputs a guess coin' for coin. The challenger outputs coin' as the final output of the experiment.

Hyb_1**:** This is the same as Hyb_0 except that for all $i \in [q]$, we generate $\mathsf{ske.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{skfe.sk}_i)$, where $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T[f_i, \mathsf{skl.ek}_i])$ and $(\mathsf{skl.ek}_i, \mathsf{skl}.s\mathcal{K}_i, \mathsf{skl.vk}_i) \leftarrow \mathsf{SKL.KeyGen}(1^\lambda)$. Note that the SKE secret key $\mathsf{ske.sk}$ never appears in the view of $\mathcal{A}$. Hence, we obtain $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| = \mathsf{negl}(\lambda)$ by the security of SKE.

Hyb_2**:** This is the same as Hyb_1 except that we generate $\mathsf{fe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{fe.pk}, (\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot))$, where $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, \bot, \mathsf{K}, 0, \bot))$. By the definition of W described in Figure 1, if we decrypt $\mathsf{fe.ct}^*$ by $\mathsf{fe.sk}_i$, we obtain

- $\mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f(x_0^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i))$ in Hyb_1 since the plaintext in $\mathsf{fe.ct}^*$ is $(x_0^* \| 0^{\ell_{\mathsf{pad}}}, \bot, \mathsf{K})$,
- $z_i = \mathsf{SKFE.Dec}(\mathsf{skfe.sk}_i, \mathsf{skfe.ct}^*)$ in Hyb_2 since $\mathsf{ske.ct}$ is a ciphertext of $\mathsf{skfe.sk}_i$ and the plaintext in $\mathsf{fe.ct}^*$ is $(\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot)$, where $\mathsf{skfe.ct}^* = \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, \bot, \mathsf{K}, 0, \bot))$. By the correctness of SKFE, $z_i = \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f(x_0^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i))$.

That is, for all $i \in [q]$, it holds that $W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i](x_0^* \| 0^{\ell_{\mathsf{pad}}}, \bot, \mathsf{K}) = W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i](\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot)$. Hence, we can use the security of PKFE and obtain $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| = \mathsf{negl}(\lambda)$. See Lemma C.1 for the detail.

After this game, we can focus on SKFE.

Hyb_3**:** This is the same as Hyb_2 except that we generate $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, x_1^*, \mathsf{K}, 0, \bot))$ and $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i])$, where $T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i]$ is described in Figure 3. Since $i \in [q]$, it holds that $T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, 0, \bot) = T[f_i, \mathsf{skl.ek}_i](x_0^*, \bot, \mathsf{K}, 0, \bot)$ for all $i \in [q]$. Hence, by the adaptively single-ciphertext function privacy of SKFE, we obtain $|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| = \mathsf{negl}(\lambda)$. See Lemma C.2 for the detail.

Hyb_3^j**:** This is the same as Hyb_3 except that we generate $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, x_1^*, \mathsf{K}, j, \bot))$. Apparently, Hyb_3^0 is the same as Hyb_3. We show it holds that $\left|\Pr\left[\mathsf{Hyb}_3^{j-1}=1\right]-\Pr\left[\mathsf{Hyb}_3^{j}=1\right]\right| = \mathsf{negl}(\lambda)$ for $j \in [q]$ in Lemma 7.7.

Hyb_4**:** This is the same as Hyb_3^q except that we generate $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T[f_i, \mathsf{skl.ek}_i])$ and $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_1^*, \bot, \mathsf{K}, 0, \bot))$. Recall that in Hyb_3^q, we use $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i])$ and $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, x_1^*, \mathsf{K}, q, \bot))$. By the definition of T_{hyb} and T, it holds that for all $i \in [q]$,

$$\begin{aligned} T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, q, \bot) &= \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_1^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i)) \\ &= T[f_i, \mathsf{skl.ek}_i](x_1^*, \bot, \mathsf{K}, 0, \bot). \end{aligned}$$

Hence, we can use the adaptively single-ciphertext function privacy of SKFE and obtain $\left|\Pr\left[\mathsf{Hyb}_3^q=1\right]-\Pr[\mathsf{Hyb}_4=1]\right| = \mathsf{negl}(\lambda)$. See Lemma C.3 for the detail.

Now, we use x_1^* instead of x_0^* and erased x_0^* in the challenge ciphertext. Hence, we focus on PKFE again and undo the changes from Hyb_1 to Hyb_2 and from Hyb_0 to Hyb_1.

Hyb_5**:** This is the same as Hyb_4 except that we generate $\mathsf{fe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{fe.pk}, (x_1^* \| 0^{\ell_{\mathsf{pad}}}, \bot, \mathsf{K}))$. This is the reverse transition from Hyb_1 to Hyb_2, so we obtain $|\Pr[\mathsf{Hyb}_4=1]-\Pr[\mathsf{Hyb}_5=1]| = \mathsf{negl}(\lambda)$ by the security of PKFE as the proof of Lemma C.1.

Hyb_6**:** This is the same as Hyb_5 except that we generate $\mathsf{ske.ct}_i \leftarrow \{0,1\}^{\ell}$. As the transition from Hyb_0 to Hyb_1, we obtain $|\Pr[\mathsf{Hyb}_5=1]-\Pr[\mathsf{Hyb}_6=1]| = \mathsf{negl}(\lambda)$ by the ciphertext pseudorandomness of SKE. It is easy to see that Hyb_6 is the same as $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{PKFE\text{-}SKL},\mathcal{A}}(1^\lambda, 1)$.

Function $T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0, x_1, \mathsf{K}, j, \mathsf{skl.ct}^*)$

Constants: Function f_i, PKE-SKL encryption key $\mathsf{skl.ek}_i$, index i.

Input: Plaintext x_0 and x_1, PRF key K, index j, a PKE-SKL ciphertext $\mathsf{skl.ct}^*$.

1. If $i \le j$, compute and output $\mathsf{skl.ct}_i := \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_1); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i))$.
2. If $i > j$, compute and output $\mathsf{skl.ct}_i := \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_0); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i))$.

Figure 3: The description of $T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i]$

If we prove Lemma 7.7, we complete the proof of Theorem 7.5. □

Lemma 7.7. *For all $j \in [q]$, it holds that $\left|\Pr\left[\mathsf{Hyb}_3^{j-1}=1\right]-\Pr\left[\mathsf{Hyb}_3^{j}=1\right]\right| = \mathsf{negl}(\lambda)$ if SKFE is fully function private, SKL is IND-KLA, and PRF is a puncturable PRF.*

Proof. We define a sequence of hybrid games.

G_0**:** This is the same as Hyb_3^{j-1}. That is, $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, x_1^*, \mathsf{K}, j-1, \bot))$ and $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i])$.

G_1**:** This is the same as G_0 except that we generate $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, x_1^*, \mathsf{K}, j, \mathsf{skl.ct}^*))$, where $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_0^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_j))$ and $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i])$, where $T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i]$ is described in Figure 4. By the definitions of T_{hyb} and T_{emb}, it holds that

$$T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, j-1, \bot) = T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, j, \mathsf{skl.ct}^*)$$

for all $i \in [q]$ since $\mathsf{skl.ct}^*$ is an encryption of $f_j(x_0^*)$. Hence, by the adaptively single-ciphertext function privacy of SKFE, we obtain $|\Pr[\mathsf{G}_0=1]-\Pr[\mathsf{G}_1=1]| = \mathsf{negl}(\lambda)$. See Lemma C.4 for the detail.

G_2: This is the same as G_1 except that we use a punctured PRF key $\mathsf{K}_{\neq \mathsf{skl.ek}_j} = \mathsf{Puncture}(\mathsf{K}, \mathsf{skl.ek}_j)$. By the functionality of punctured PRF keys, it holds that

$$T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, j, \mathsf{skl.ct}^*) = T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}_{\neq \mathsf{skl.ek}_j}, j, \mathsf{skl.ct}^*)$$

for all $i \in [q]$. Note that T_{emb} directly uses $\mathsf{skl.ct}^*$ instead of computing $\mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_0^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_j))$, so $\mathsf{K}_{\neq \mathsf{skl.ek}_j}$ is sufficient for the functional equivalence. The only difference between the two games is whether the PRF key is K or $\mathsf{K}_{\neq \mathsf{skl.ek}_j}$. Hence, we can use the adaptively single-ciphertext function privacy of SKFE and obtain $|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| = \mathsf{negl}(\lambda)$. We omit the proof since it is easy.

G_3: This is the same as G_2 except that we generate $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_0^*))$. That is, we use uniform randomness for generating $\mathsf{skl.ct}^*$. By the punctured pseudorandomness of PRF, we obtain $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| = \mathsf{negl}(\lambda)$. We omit the proof since it is easy.

G_4: This is the same as G_3 except that we generate $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_1^*))$. We consider two cases.

- If $(f_j, \mathsf{vk}_j, \bot)$ is recorded in $L_{\mathcal{KG}}$, that is, valid fsk_j is not returned, it must hold that $f_j(x_0^*) = f_j(x_1^*)$ by the requirement of Ada-IND-KLA security. In this case, the distribution of $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_0^*))$ is trivially the same as that of $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_1^*))$. Hence, we obtain $\Pr[\mathsf{G}_3 = 1] = \Pr[\mathsf{G}_4 = 1]$.
- If $(f_j, \mathsf{vk}_j, \top)$ is recorded in $L_{\mathcal{KG}}$, that is, it is certified that the adversary returned valid fsk_j, it could hold that $f_j(x_0^*) \neq f_j(x_1^*)$ by the requirement of Ada-IND-KLA security. In this case, we use IND-KLA security of SKL since $\mathsf{skl}.\mathit{sk}_j$ was returned. We have that $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_0^*))$ is computationally indistinguishable from $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_1^*))$. Hence, we obtain $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| = \mathsf{negl}(\lambda)$ in this case. See Lemma C.5 for the detail.

Hence, we obtain $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| = \mathsf{negl}(\lambda)$ in either cases.

G_5: This is the same as G_4 except that we undo the change in G_3. That is, we use $\mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_j)$ for the randomness of $\mathsf{skl.ct}^*$. We obtain $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| = \mathsf{negl}(\lambda)$ by the punctured pseudorandomness of PRF. We omit the proof since it is easy.

G_6: This is the same as G_5 except that we undo the change in G_2. That is, we use a unpunctured PRF key K. We obtain $|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| = \mathsf{negl}(\lambda)$ by the adaptively single-ciphertext function privacy as the transition from G_1 to G_2. So, we omit the proof.

G_7: This is the same as G_6 except that we undo the change in G_1, but the index is still j. That is, we use $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, x_1^*, \mathsf{K}, j, \bot))$ and $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i])$. We obtain $|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_7 = 1]| = \mathsf{negl}(\lambda)$ by the adaptively single-ciphertext function privacy of SKFE. The proof is similar to that of Lemma C.4. So, we omit the proof.

Function $T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i](x_0, x_1, \mathsf{K}, j, \mathsf{skl.ct}^*)$

Constants: Function f_i, encryption key of PKE-SKL $\mathsf{skl.ek}_i$, index i.

Input: Plaintext x_0, x_1, PRF key K, index j, an SKL ciphertext $\mathsf{skl.ct}^*$.

1. If $i = j$, output $\mathsf{skl.ct}^*$.
2. If $i < j$, compute and output $\mathsf{skl.ct}_i := \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_1); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i))$.
3. If $i > j$, compute and output $\mathsf{skl.ct}_i := \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_0); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i))$.

Figure 4: The description of $T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i]$

It is easy to see that G_7 is the same as Hyb_3^j. Therefore, we complete the proof. □

By Theorems 2.3, 2.8, 2.16, 2.19, 5.1 and 7.6, we obtain the following corollary.

Corollary 7.8. *If there exists IND-CPA secure PKE, there exists q-bounded Ada-IND-KLA PKFE-SKL for* P / poly.

By Theorems 2.3, 2.8, 2.16, 5.1 and 7.5 and known theorems about PKFE [GS16, LM16, KNTY19], we obtain the following corollary.

Corollary 7.9. *If there exists single-key selective-message-function secure*[23] *and weakly compact PKFE for* P / poly*, there exists Ada-IND-KLA PKFE for* P / poly.

Acknowledgement

We thank Jiayu Zhang for pointing out a technical similarity to [Zha21, Zha22], Prabhanjan Ananth for discussions on the relationship between our work and their concurrent work [APV23], and anonymous reviewers of QIP 2023 and Eurocrypt 2023 for their valuable comments. This work was supported in part by the DST "Swarnajayanti" fellowship, Cybersecurity Center of Excellence, IIT Madras, National Blockchain Project and the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of sponsors. The fourth author was partially supported by JST AIP Acceleration Research JPMJCR22U5 and JSPS KAKENHI Grant Number 19H01109, Japan.

References

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242. IEEE, 2009. (Cited on page 3, 11.)

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010. (Cited on page 53.)

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015. (Cited on page 11, 17.)

[AC02] Mark Adcock and Richard Cleve. A quantum goldreich-levin theorem with cryptographic applications. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002. (Cited on page 5, 18.)

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 41–60. ACM Press, May 2012. (Cited on page 3.)

[AGKZ20] Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 255–268. ACM Press, June 2020. (Cited on page 3.)

[AK21] Prabhanjan Ananth and Fatih Kaleoglu. Unclonable encryption, revisited. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 299–329. Springer, Heidelberg, November 2021. (Cited on page 11.)

[23] The adversary must select the target plaintext pair and function at the beginning of the game. This is the same as weakly selective security by Garg and Srinivasan [GS16].

[AKL+22] Prabhanjan Ananth, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry. On the feasibility of unclonable encryption, and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 212–241. Springer, Heidelberg, August 2022. (Cited on page 11.)

[AL21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 501–530. Springer, Heidelberg, October 2021. (Cited on page 3, 11.)

[ALL+21] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 526–555, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 3, 4, 11, 27.)

[APV23] Prabhanjan Ananth, Alexander Poremba, and Vinod Vaikuntanathan. Revocable cryptography from learning with errors. Cryptology ePrint Archive, Paper 2023/325, 2023. https://eprint.iacr.org/2023/325. (Cited on page 12, 59.)

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198. Springer, Heidelberg, December 2019. (Cited on page 17, 18.)

[BB20] Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *arXiv preprint arXiv:2003.06557*, 2020. (Cited on page 3.)

[BDTW01] Dan Boneh, Xuhua Ding, Gene Tsudik, and Chi-Ming Wong. A method for fast revocation of public key certificates and security capabilities. In Dan S. Wallach, editor, *USENIX Security 2001*. USENIX Association, August 2001. (Cited on page 3.)

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014. (Cited on page 52.)

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 14.)

[BI20] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 92–122. Springer, Heidelberg, November 2020. (Cited on page 11.)

[BJL+21] Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 90–120. Springer, Heidelberg, November 2021. (Cited on page 11.)

[BK22] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. Cryptology ePrint Archive, Report 2022/1178, 2022. https://eprint.iacr.org/2022/1178. (Cited on page 11.)

[BS18] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. *Journal of Cryptology*, 31(1):202–225, January 2018. (Cited on page 11, 17.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 14.)

[BZ13] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 361–379. Springer, Heidelberg, August 2013. (Cited on page 7, 8, 18.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010. (Cited on page 53.)

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 556–584, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 3, 5, 11, 18, 63, 64.)

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. *arXiv (CoRR)*, abs/2009.13865, 2020. (Cited on page 11.)

[CV22] Eric Culf and Thomas Vidick. A monogamy-of-entanglement game for subspace coset states. *Quantum*, 6:791, sep 2022. (Cited on page 3, 11.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. (Cited on page 14.)

[GKM+19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. Watermarking public-key cryptographic primitives. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 367–398. Springer, Heidelberg, August 2019. (Cited on page 4.)

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 361–388. Springer, Heidelberg, October / November 2016. (Cited on page 9.)

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 419–442. Springer, Heidelberg, October / November 2016. (Cited on page 59.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. (Cited on page 10, 17, 18, 28.)

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013. (Cited on page 52.)

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877, 2020. https://eprint.iacr.org/2020/877. (Cited on page 3, 63.)

[HMNY21] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 606–636. Springer, Heidelberg, December 2021. (Cited on page 11.)

[HMNY22] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting functional encryption. Cryptology ePrint Archive, Report 2022/969, 2022. https://eprint.iacr.org/2022/969. (Cited on page 11.)

[ISV+17] Gene Itkis, Emily Shen, Mayank Varia, David Wilson, and Arkady Yerukhimovich. Bounded-collusion attribute-based encryption from minimal assumptions. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 67–87. Springer, Heidelberg, March 2017. (Cited on page 10, 48, 50.)

[KN22a] Fuyuki Kitagawa and Ryo Nishimaki. Functional encryption with secure key leasing. *Asiacrypt 2022 (to appear)*, 2022. (Cited on page 3, 4, 11, 19, 53, 55.)

[KN22b] Fuyuki Kitagawa and Ryo Nishimaki. Watermarking PRFs against quantum adversaries. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 488–518. Springer, Heidelberg, May / June 2022. (Cited on page 8, 27, 28.)

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019. (Cited on page 59.)

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 31–61. Springer, Heidelberg, November 2021. (Cited on page 11.)

[KNYY20] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure inner product encryption from LWE. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 375–404. Springer, Heidelberg, December 2020. (Cited on page 52, 53.)

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013. (Cited on page 14.)

[LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 443–468. Springer, Heidelberg, October / November 2016. (Cited on page 59.)

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. (Cited on page 15.)

[MW05] Chris Marriott and John Watrous. Quantum arthur-merlin games. *Comput. Complex.*, 14(2):122–152, 2005. (Cited on page 8.)

[Por23] Alexander Poremba. Quantum proofs of deletion for learning with errors. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 90:1–90:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. (Cited on page 11.)

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010. (Cited on page 9.)

[SW22] Or Sattath and Shai Wyborski. Uncloneable decryptors from quantum copy-protection. *arXiv (CoRR)*, abs/2203.05866, 2022. (Cited on page 11.)

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Heidelberg, August 2019. (Cited on page 53.)

[Unr15] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015. (Cited on page 11.)

[Wie83] Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983. (Cited on page 3.)

[Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999. (Cited on page 19.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 15.)

[Zha20] Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 61–91. Springer, Heidelberg, November 2020. (Cited on page 8, 26, 27.)

[Zha21] Jiayu Zhang. Succinct blind quantum computation using a random oracle. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 1370–1383. ACM Press, June 2021. (Cited on page 12, 59.)

[Zha22] Jiayu Zhang. Classical verification of quantum computations in linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 46–57. IEEE, 2022. (Cited on page 12, 59.)

A SDE Implies PKE-SKL

In this section, we discuss the relationship between SDE and PKE-SKL. There are many incomparable security definitions for SDE in the literature. Coladangelo et al. [CLLZ21] defined two incomparable security definitions called *CPA-style anti-piracy* and *random challenge anti-piracy*.[24] All constructions of SDE in [CLLZ21] are shown to satisfy both CPA-style anti-piracy and random challenge anti-piracy. Georgiou and Zhandry [GZ20] defined yet another security definition, which is similar to but slightly different from CPA-style anti-piracy of [CLLZ21].[25] Though we do not see any relationships between security notions in [GZ20] and [CLLZ21], it seems possible to prove that the construction given in [GZ20] satisfies both CPA-style anti-piracy and random challenge anti-piracy of [CLLZ21] because it is very similar to one of the schemes given in [CLLZ21].[26] In the following, we show that SDE with random challenge anti-piracy implies IND-KLA secure PKE-SKL. This means that all known constructions of SDE can be used to construct PKE-SKL.

The definitions of SDE and its random challenge anti-piracy are given below. The syntax of SDE is identical to that of PKE except that the key generation and decryption algorithms are quantum and the decryption key is quantum.

Definition A.1 (Single-Decryptor Encryption). *A single-decryptor encryption (SDE) scheme* SDE *is a tuple of three algorithms* $(\mathcal{KG}, \mathsf{Enc}, \mathcal{Dec})$*. Below, let* $\mathcal{X}$ *be the message space of* SDE*.*

$\mathcal{KG}(1^\lambda) \to (\mathsf{ek}, d\mathcal{k})$**:** *The key generation algorithm takes a security parameter* 1^λ*, and outputs an encryption key* ek *and a decryption key* $d\mathcal{k}$*.*

$\mathsf{Enc}(\mathsf{ek}, \mathsf{m}) \to \mathsf{ct}$**:** *The encryption algorithm takes an encryption key* ek *and a message* $\mathsf{m} \in \mathcal{X}$*, and outputs a ciphertext* ct*.*

$\mathcal{Dec}(d\mathcal{k}, \mathsf{ct}) \to \tilde{\mathsf{m}}$**:** *The decryption algorithm takes a decryption key* $d\mathcal{k}$ *and a ciphertext* ct*, and outputs a value* $\tilde{\mathsf{m}}$*.*

Correctness: *For every* $\mathsf{m} \in \mathcal{X}$*, we have*

$$\Pr\left[\mathcal{Dec}(d\mathcal{k}, \mathsf{ct}) = \mathsf{m} \;\middle|\; \begin{array}{l}(\mathsf{ek}, d\mathcal{k}) \leftarrow \mathcal{KG}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m})\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

In the following definition of random challenge anti-piracy, we use the notion of quantum programs with classical inputs and outputs as defined in Definition 4.7.

Definition A.2 (Random Challenge Anti-Piracy). *We say that an SDE scheme* SDE *with the message space* $\mathcal{X}$ *satisfies random-challenge anti-piracy, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{rand\text{-}chal}}_{\mathsf{SDE},\mathcal{A}}(1^\lambda)$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

[24] They actually also defined stronger variants of them called strong anti-piracy security and strong anti-piracy against random plaintexts. See [CLLZ21, Definition 6.11 and D.4 in the full version] for the detail.

[25] We note that [GZ20] appeared before [CLLZ21].

[26] Here, we are referring to the construction of SDE based on one-shot signatures and extractable witness encryption in [GZ20, Section 5]. For proving that the scheme satisfies the security notions of [CLLZ21], we will need to go trough the "strong" variants of them similarly to [CLLZ21].

1. *$\mathcal{C}$ runs $(\mathsf{ek}, d\mathcal{k}) \leftarrow \mathcal{KG}(1^\lambda)$ and sends ek and $d\mathcal{k}$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends two (possibly entangled) quantum programs $(\mathcal{D}_0, \mathcal{D}_1)$ with classical inputs and outputs to $\mathcal{C}$.*
3. *For $b \in \{0,1\}$, $\mathcal{C}$ chooses $\mathsf{m}_b^* \leftarrow \mathcal{X}$, generates $\mathsf{ct}_b^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}_b^*)$, and runs $\mathcal{D}_b$ on input ct_b^* to obtain an output m_b. $\mathcal{C}$ outputs 1 if $\mathsf{m}_b = \mathsf{m}_b^*$ for $b \in \{0,1\}$ and otherwise outputs 0 as the final output of the experiment.*

For any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{rand\text{-}chal}}_{\mathsf{SDE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{rand\text{-}chal}}_{\mathsf{SDE},\mathcal{A}}(1^\lambda) \to 1\right] \leq \mathsf{negl}(\lambda).$$

We prove the following theorem.

Theorem A.3. *If there exists an SDE scheme that satisfies random challenge anti-piracy, there exists an IND-KLA secure PKE-SKL scheme.*

Proof. Let $\mathsf{SDE} = (\mathsf{SDE}.\mathcal{KG}, \mathsf{SDE.Enc}, \mathsf{SDE}.\mathcal{D}ec)$ be an SDE scheme that satisfies random challenge anti-piracy. By Theorem 3.9, it suffices to construct a one-query OW-KLA secure PKE-SKL scheme. We construct a one-query OW-KLA secure PKE-SKL scheme $\mathsf{SKL} = (\mathsf{SKL}.\mathcal{KG}, \mathsf{SKL.Enc}, \mathsf{SKL}.\mathcal{D}ec, \mathsf{SKL}.\mathcal{V}rfy)$ as follows.

$\mathsf{SKL}.\mathcal{KG}(1^\lambda)$**:** Run $(\mathsf{sde.ek}, \mathsf{sde}.d\mathcal{k}) \leftarrow \mathsf{SDE}.\mathcal{KG}(1^\lambda)$ and output $\mathsf{skl.ek} := \mathsf{sde.ek}$, $\mathsf{skl}.d\mathcal{k} := \mathsf{sde}.d\mathcal{k}$, and $\mathsf{skl.vk} := \mathsf{sde.ek}$.

$\mathsf{SKL.Enc}(\mathsf{skl.ek}, \mathsf{m})$**:** This is identical to $\mathsf{SDE.Enc}$.

$\mathsf{SKL}.\mathcal{D}ec(d\mathcal{k}, \mathsf{ct})$**:** This is identical to $\mathsf{SDE}.\mathcal{D}ec$.

$\mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}, \mathsf{skl}.\widetilde{d\mathcal{k}})$**:** Parse $\mathsf{skl.vk} = \mathsf{sde.ek}$, choose $\mathsf{m}^* \leftarrow \mathcal{X}$, run $\mathsf{ct}^* \leftarrow \mathsf{SDE.Enc}(\mathsf{sde.ek}, \mathsf{m}^*)$ and $\mathsf{m} \leftarrow \mathsf{SDE}.\mathcal{D}ec(\mathsf{skl}.\widetilde{d\mathcal{k}}, \mathsf{ct}^*)$, and output $\top$ if and only if $\mathsf{m} = \mathsf{m}^*$.

Suppose that SKL is not one-query OW-KLA secure. Let $\mathcal{A}$ be a QPT adversary that breaks the one-query OW-KLA security of SKL. We construct a QPT adversary $\mathcal{B}$ that breaks the random challenge anti-piracy of SDE as follows.

$\mathcal{B}(\mathsf{sde.ek}, \mathsf{sde}.d\mathcal{k})$**:** Set $\mathsf{skl.ek} := \mathsf{sde.ek}$, $\mathsf{skl}.d\mathcal{k} := \mathsf{sde}.d\mathcal{k}$, and $\mathsf{skl.vk} := \mathsf{sde.ek}$ and sends $(\mathsf{skl.ek}, \mathsf{skl}.d\mathcal{k}, \mathsf{skl.vk})$ to $\mathcal{A}$. When $\mathcal{A}$ makes a verification query $\mathsf{skl}.\widetilde{d\mathcal{k}}$, $\mathcal{B}$ returns 1 to $\mathcal{A}$ as the response from the oracle. Let $\mathcal{D}_0$ be the quantum program with classical inputs and outputs that takes ct as input and outputs $\mathsf{m} \leftarrow \mathsf{SDE}.\mathcal{D}ec(\mathsf{skl}.\widetilde{d\mathcal{k}}, \mathsf{ct})$. When $\mathcal{A}$ sends $\mathsf{RequestChallenge}$, let $\mathcal{D}_1$ be the quantum program with classical inputs and outputs, in which $\mathcal{A}$'s internal state is hardwired, that takes ct as input, runs the rest of $\mathcal{A}$ on the challenge ciphertext ct, and outputs $\mathcal{A}$'s output m. Output $(\mathcal{D}_0, \mathcal{D}_1)$.

By the construction of $\mathcal{B}$ and the deferred measurement principle, it is immediate to see that $\mathsf{Adv}^{\mathsf{rand\text{-}chal}}_{\mathsf{SDE},\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathsf{ow\text{-}kla}}_{\mathsf{SKL},\mathcal{A}}(\lambda)$. Thus, $\mathcal{B}$ breaks the random challenge anti-piracy of SDE, which is contradiction. Therefore, SKL is one-query OW-KLA secure. □

Remark A.4 (On CPA-Style Anti-Piracy). We do not know if SDE with CPA-style anti-piracy implies PKE-SKL. On the other hand, it seems possible to show that SDE with the "strong" variant of CPA-style anti-piracy (called strong anti piracy [CLLZ21, Definition 6.11 in the full version]) implies PKE-SKL. In the single-bit encryption setting, the security roughly means that the adversary given one decryption key cannot generate two "good" distinguishers that distinguish encryptions of 0 and 1. Then our idea is to construct a PKE-SKL scheme whose verification algorithm accepts if a returned decryption key gives a "good" distinguisher. Then the strong anti piracy ensures that if the adversary passes the verification, then it cannot keep a "good" distinguisher, which in particular means that it cannot distinguish encryptions of 0 and 1. Thus, the PKE-SKL scheme is one-query IND-KLA secure.

B OW-CPA from CoIC-KLA

We show the following lemma.

Lemma B.1. *If a PKE scheme with a super-polynomial-size message space is CoIC-KLA secure, then it is OW-CPA secure.*

Proof. Let $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a CoIC-KLA secure PKE scheme with the message space $\mathcal{X}$ such that $|\mathcal{X}|$ is super-polynomial in λ. Toward contradiction, suppose that it is not OW-CPA secure. Let $\mathcal{A}$ be an adversary that breaks OW-CPA security of PKE. Then we construct $\mathcal{B}$ that breaks CoIC-KLA security of PKE as follows.

$\mathcal{B}(\mathsf{ek}_0, \mathsf{ek}_1, \mathit{dk})$**:** Measure dk to get $(\beta, \mathsf{dk}_\beta)$ for $\beta \in \{0,1\}$. Choose $(\mathsf{m}_0^*, \mathsf{m}_1^*) \leftarrow \mathcal{X}^2$ and send $(\mathsf{m}_0^*, \mathsf{m}_1^*)$ to the challenger (without making any oracle query). Upon receiving $(\mathsf{ct}_0^*, \mathsf{ct}_1^*)$ from the challenger, run $\mathsf{m}'_\beta \leftarrow \mathsf{Dec}(\mathsf{dk}_\beta, \mathsf{ct}_\beta^*)$ and $\mathsf{m}'_{\beta\oplus 1} \leftarrow \mathcal{A}(\mathsf{ek}_{\beta\oplus 1}, \mathsf{ct}_{\beta\oplus 1}^*)$ and output 0 if $\mathsf{m}'_0 = \mathsf{m}'_1$ and 1 otherwise.

Note that the challenger implicitly chooses $a, b \leftarrow \{0,1\}$ and generates $\mathsf{ct}_0^* \leftarrow \mathsf{Enc}(\mathsf{ek}_0, \mathsf{m}_a^*)$ and $\mathsf{ct}_1^* \leftarrow \mathsf{Enc}(\mathsf{ek}_1, \mathsf{m}_{a\oplus b}^*)$. $\mathcal{B}$'s goal is to guess b.

If $b = 0$, by the correctness of PKE, we have $\Pr\left[\mathsf{m}'_\beta = \mathsf{m}_a^*\right] = 1 - \mathsf{negl}(\lambda)$. By the assumption that $\mathcal{A}$ breaks OW-CPA security, $\Pr\left[\mathsf{m}'_{\beta\oplus 1} = \mathsf{m}_a^*\right]$ is non-negligible. In particular, $\Pr[\mathcal{B}(\mathsf{ek}_0, \mathsf{ek}_1, \mathit{dk}) \to 0 | b = 0]$ is non-negligible. If $b = 1$, by the correctness of PKE, we have $\Pr\left[\mathsf{m}'_\beta = \mathsf{m}_{a+\beta}^*\right] = 1 - \mathsf{negl}(\lambda)$. On the other hand, $\mathsf{ct}_{\beta\oplus 1}^*$ contains no information of $\mathsf{m}_{a\oplus\beta}^*$. Therefore, $\Pr\left[\mathsf{m}'_{\beta\oplus 1} = \mathsf{m}_{a\oplus\beta}^*\right] \le 1/|\mathcal{X}| = \mathsf{negl}(\lambda)$. Thus, $\Pr[\mathcal{B}(\mathsf{ek}_0, \mathsf{ek}_1, \mathit{dk}) \to 0 | b = 1] = \mathsf{negl}(\lambda)$. Thus, $|2\Pr[\mathcal{B}(\mathsf{ek}_0, \mathsf{ek}_1, \mathit{dk}) \to b] - 1| = |\Pr[\mathcal{B}(\mathsf{ek}_0, \mathsf{ek}_1, \mathit{dk}) \to 0 | b = 0] - \Pr[\mathcal{B}(\mathsf{ek}_0, \mathsf{ek}_1, \mathit{dk}) \to 0 | b = 1]|$ is non-negligible. This contradicts the assumed CoIC-KLA security. Thus, PKE is OW-CPA secure. □

C Deferred Proofs for PKFE-SKL

In this section, we present the deferred proofs in Section 7.

Lemma C.1. *If* PKFE *is adaptively secure, it holds that* $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| = \mathsf{negl}(\lambda)$.

Proof. We construct an adversary $\mathcal{B}$ for PKFE by using the distinguisher $\mathcal{D}$ for these two games.

1. $\mathcal{B}$ is given $\mathsf{fe.pk}$ and sends $\mathsf{pk} := \mathsf{fe.pk}$ to $\mathcal{D}$. $\mathcal{B}$ also generates $\mathsf{ske.sk} \leftarrow \{0,1\}^\lambda$ and $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$.
2. When $\mathcal{D}$ sends f_i to $O_{\mathcal{KG}}$, $\mathcal{B}$ generates $(\mathsf{skl.ek}_i, \mathsf{skl}.\mathit{sk}_i, \mathsf{skl.vk}_i) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$, $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, V[f_i, \mathsf{skl.ek}_i])$, and $\mathsf{ske.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{skfe.sk}_i)$. Then, $\mathcal{B}$ sends $W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i]$ to its challenger and receives $\mathsf{fe.sk}_{W,i} \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i])$. $\mathcal{B}$ returns $\mathit{fsk}_i := (\mathsf{fe.sk}_{W,i}, \mathsf{skl}.\mathit{sk}_i)$ to $\mathcal{D}$ and adds $(f_i, \mathsf{skl.vk}_i, \bot)$ to $L_{\mathcal{KG}}$.
3. When $\mathcal{D}$ sends (f_i, fsk'_i) to $O_{\mathcal{V}\mathit{rfy}}$, $\mathcal{B}$ finds an entry $(f_i, \mathsf{skl.vk}_i, V_i)$ from $L_{\mathcal{KG}}$ and parses $\mathit{fsk}'_i = (\mathsf{fe.sk}'_i, \mathsf{skl}.\mathit{sk}'_i)$. $\mathcal{B}$ returns $d := \mathsf{SKL}.\mathcal{V}\mathit{rfy}(\mathsf{skl.vk}_i, \mathsf{skl}.\mathit{sk}'_i)$. If $V_i = \top$, $\mathcal{B}$ does not update the entry. Otherwise, $\mathcal{B}$ updates the entry by setting $V_i := d$.
4. When $\mathcal{D}$ sends (x_0^*, x_1^*), $\mathcal{B}$ generates $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, \bot, \mathsf{K}, 0, \bot))$. $\mathcal{B}$ sets $X_0^* := (x_0^* \| 0^{\ell_{\mathsf{pad}}}, \bot, \mathsf{K})$ and $X_1^* := (\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot)$, sends (X_0^*, X_1^*) to its challenger, and receives $\mathsf{fe.ct}^*$. $\mathcal{B}$ passes $\mathsf{ct}^* := \mathsf{fe.ct}^*$ to $\mathcal{D}$.
5. $\mathcal{B}$ outputs what $\mathcal{D}$ outputs.

By the definition of W described in Figure 1, if we decrypt $\mathsf{fe.ct}^*$ by $\mathsf{fe.sk}_{W,i}$, we obtain

- $\mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f(x_0); \mathsf{F}_\mathsf{K}(\mathsf{skl.ek}_i))$ if $\mathsf{fe.ct}^*$ is generated from X_0^*,

- $z_i = \mathsf{SKFE.Dec}(\mathsf{skfe.sk}_i, \mathsf{skfe.ct}^*)$ if $\mathsf{fe.ct}^*$ is generated from X_1^* since $\mathsf{ske.ct}_i$ is a ciphertext of $\mathsf{skfe.sk}_i$, where $\mathsf{skfe.ct}^* = \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, \bot, \mathsf{K}, 0, \bot))$. By the correctness of SKFE and the definition of $T[f_i, \mathsf{skl.ek}_i]$, it holds that $z_i = \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f(x_0^*); \mathsf{F}_\mathsf{K}(\mathsf{skl.ek}_i))$.

That is, for all $i \in [q]$, it holds that $W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i](X_0^*) = W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i](X_1^*)$, and $\mathcal{B}$ is a valid adversary of PKFE.

It is easy to see that if $\mathsf{fe.ct}^*$ is an encryption of X_0^* and X_1^*, $\mathcal{B}$ perfectly simulates Hyb_1 and Hyb_2, respectively. This completes the proof. □

Lemma C.2. *If* SKFE *is adaptively single-ciphertext function private, it holds that* $|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| = \mathsf{negl}(\lambda)$.

Proof. We construct an adversary $\mathcal{B}$ for SKFE by using the distinguisher $\mathcal{D}$ for these two games.

1. $\mathcal{B}$ generates $(\mathsf{fe.pk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and $\mathsf{ske.sk} \leftarrow \{0,1\}^\lambda$, and sends $\mathsf{pk} := \mathsf{fe.pk}$ to $\mathcal{D}$.
2. When $\mathcal{D}$ sends f_i to $O_{\mathcal{KG}}$, $\mathcal{B}$ generates $(\mathsf{skl.ek}_i, \mathsf{skl}.s\mathcal{K}_i, \mathsf{skl.vk}_i) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$, sends a key query $(F_{0,i}, F_{1,i}) := (T[f_i, \mathsf{skl.ek}_i], T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i])$ to its challenger, and receives $\mathsf{skfe.sk}_i$. $\mathcal{B}$ also generates $\mathsf{ske.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{skfe.sk}_i)$ and $\mathsf{fe.sk}_{W,i} \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i])$. $\mathcal{B}$ returns $fs\mathcal{K}_i := (\mathsf{fe.sk}_{W,i}, \mathsf{skl}.s\mathcal{K}_i)$ to $\mathcal{D}$ and adds $(f_i, \mathsf{skl.vk}_i, \bot)$ to $L_{\mathcal{KG}}$.
3. When $\mathcal{D}$ sends $(f_i, fs\mathcal{K}_i')$ to $O_{\mathcal{V}rfy}$, $\mathcal{B}$ finds an entry $(f_i, \mathsf{skl.vk}_i, V_i)$ from $L_{\mathcal{KG}}$ and parses $fs\mathcal{K}_i' = (\mathsf{fe.sk}_i', \mathsf{skl}.s\mathcal{K}_i')$. $\mathcal{B}$ returns $d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}_i, \mathsf{skl}.s\mathcal{K}_i')$. If $V_i = \top$, $\mathcal{B}$ does not update the entry. Otherwise, $\mathcal{B}$ updates the entry by setting $V_i := d$.
4. When $\mathcal{D}$ sends (x_0^*, x_1^*), $\mathcal{B}$ generates $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, sets $X_0^* := (x_0^*, \bot, \mathsf{K}, 0, \bot)$ and $X_1^* := (x_0^*, x_1^*, \mathsf{K}, 0, \bot)$, sends an encryption query (X_0^*, X_1^*) to its challenger, and receives $\mathsf{skfe.ct}^*$. $\mathcal{B}$ also generates $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.pk}, (\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot))$ and passes $\mathsf{ct}^* := \mathsf{fe.ct}^*$ to $\mathcal{D}$.
5. $\mathcal{B}$ outputs what $\mathcal{D}$ outputs.

Since $i \in [q]$, it holds that $T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, 0, \bot) = T[f_i, \mathsf{skl.ek}_i](x_0^*, \bot, \mathsf{K}, 0, \bot)$ for all $i \in [q]$. That is, $F_{0,i}(X_0^*) = F_{1,i}(X_1^*)$ for all $i \in [q]$ and $\mathcal{B}$ is an valid adversary for SKFE.

If $\mathsf{skfe.ct}^*$ is an encryption of X_0^* and $\mathsf{skfe.sk}_i$ is a functional decryption key for $F_{0,i}$, $\mathcal{B}$ perfectly simulate Hyb_2. If $\mathsf{skfe.ct}^*$ is an encryption of X_1^* and $\mathsf{skfe.sk}_i$ is a functional decryption key for $F_{1,i}$, $\mathcal{B}$ perfectly simulate Hyb_3. This completes the proof. □

Lemma C.3. *If* SKFE *is adaptively single-ciphertext function private, it holds that* $\left|\Pr\left[\mathsf{Hyb}_3^q = 1\right] - \Pr[\mathsf{Hyb}_4 = 1]\right| = \mathsf{negl}(\lambda)$.

Proof. We construct an adversary $\mathcal{B}$ for SKFE by using the distinguisher $\mathcal{D}$ for these two games.

1. $\mathcal{B}$ generates $(\mathsf{fe.pk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and $\mathsf{ske.sk} \leftarrow \{0,1\}^\lambda$, and sends $\mathsf{pk} := \mathsf{fe.pk}$ to $\mathcal{D}$.
2. When $\mathcal{D}$ sends f_i to $O_{\mathcal{KG}}$, $\mathcal{B}$ generates $(\mathsf{skl.ek}_i, \mathsf{skl}.s\mathcal{K}_i, \mathsf{skl.vk}_i) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$, sends a key query $(F_{0,i}, F_{1,i}) := (T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i], T[f_i, \mathsf{skl.ek}_i])$ to its challenger, and receives $\mathsf{skfe.sk}_i$. $\mathcal{B}$ also generates $\mathsf{ske.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{skfe.sk}_i)$ and $\mathsf{fe.sk}_{W,i} \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i])$. $\mathcal{B}$ returns $fs\mathcal{K}_i := (\mathsf{fe.sk}_{W,i}, \mathsf{skl}.s\mathcal{K}_i)$ to $\mathcal{D}$ and adds $(f_i, \mathsf{skl.vk}_i, \bot)$ to $L_{\mathcal{KG}}$.
3. When $\mathcal{D}$ sends $(f_i, fs\mathcal{K}_i')$ to $O_{\mathcal{V}rfy}$, $\mathcal{B}$ finds an entry $(f_i, \mathsf{skl.vk}_i, V_i)$ from $L_{\mathcal{KG}}$ and parses $fs\mathcal{K}_i' = (\mathsf{fe.sk}_i', \mathsf{skl}.s\mathcal{K}_i')$. $\mathcal{B}$ returns $d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}_i, \mathsf{skl}.s\mathcal{K}_i')$. If $V_i = \top$, $\mathcal{B}$ does not update the entry. Otherwise, $\mathcal{B}$ updates the entry by setting $V_i := d$.
4. When $\mathcal{D}$ sends (x_0^*, x_1^*), $\mathcal{B}$ generates $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, sets $X_0^* := (x_0^*, x_1^*, \mathsf{K}, q, \bot)$ and $X_1^* := (x_1^*, \bot, \mathsf{K}, 0, \bot)$, sends an encryption query (X_0^*, X_1^*) to its challenger, and receives $\mathsf{skfe.ct}^*$. $\mathcal{B}$ also generates $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.pk}, (\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot))$ and passes $\mathsf{ct}^* := \mathsf{fe.ct}^*$ to $\mathcal{D}$.

5. $\mathcal{B}$ outputs what $\mathcal{D}$ outputs.

By the definition of T_{hyb} and T, it holds that for all $i \in [q]$,

$$\begin{aligned} T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, q, \bot) &= \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_1^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i)) \\ &= T[f_i, \mathsf{skl.ek}_i](x_1^*, \bot, \mathsf{K}, 0, \bot). \end{aligned}$$

That is, $F_{0,i}(X_0^*) = F_{1,i}(X_1^*)$ for all $i \in [q]$ and $\mathcal{B}$ is an valid adversary for SKFE.

If $\mathsf{skfe.ct}^*$ is an encryption of X_0^* and $\mathsf{skfe.sk}_i$ is a functional decryption key for $F_{0,i}$, $\mathcal{B}$ perfectly simulate Hyb_3^q. If $\mathsf{skfe.ct}^*$ is an encryption of X_1^* and $\mathsf{skfe.sk}_i$ is a functional decryption key for $F_{1,i}$, $\mathcal{B}$ perfectly simulate Hyb_4. This completes the proof. □

Lemma C.4. *If* SKFE *is adaptively single-ciphertext function private, it holds that* $|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| = \mathsf{negl}(\lambda)$.

Proof. We construct an adversary $\mathcal{B}$ for SKFE by using the distinguisher $\mathcal{D}$ for these two games.

1. $\mathcal{B}$ generates $(\mathsf{fe.pk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and $\mathsf{ske.sk} \leftarrow \{0,1\}^\lambda$, and sends $\mathsf{pk} := \mathsf{fe.pk}$ to $\mathcal{D}$.
2. When $\mathcal{D}$ sends f_i to $O_{\mathcal{KG}}$, $\mathcal{B}$ generates $(\mathsf{skl.ek}_i, \mathsf{skl.}s\mathcal{K}_i, \mathsf{skl.vk}_i) \leftarrow \mathsf{SKL.}\mathcal{KG}(1^\lambda)$, sends a key query $(F_{0,i}, F_{1,i}) := (T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i], T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i])$ to its challenger, and receives $\mathsf{skfe.sk}_i$. $\mathcal{B}$ also generates $\mathsf{ske.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{skfe.sk}_i)$ and $\mathsf{fe.sk}_{W,i} \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i])$. $\mathcal{B}$ returns $\mathit{fs}\mathcal{K}_i := (\mathsf{fe.sk}_{W,i}, \mathsf{skl.}s\mathcal{K}_i)$ to $\mathcal{D}$ and adds $(f_i, \mathsf{skl.vk}_i, \bot)$ to $L_{\mathcal{KG}}$.
3. When $\mathcal{D}$ sends $(f_i, \mathit{fs}\mathcal{K}_i')$ to $O_{\mathcal{V}\mathit{rfy}}$, $\mathcal{B}$ finds an entry $(f_i, \mathsf{skl.vk}_i, V_i)$ from $L_{\mathcal{KG}}$ and parses $\mathit{fs}\mathcal{K}_i' = (\mathsf{fe.sk}_i', \mathsf{skl.}s\mathcal{K}_i')$. $\mathcal{B}$ returns $d := \mathsf{SKL.}\mathcal{V}\mathit{rfy}(\mathsf{skl.vk}_i, \mathsf{skl.}s\mathcal{K}_i')$. If $V_i = \top$, $\mathcal{B}$ does not update the entry. Otherwise, $\mathcal{B}$ updates the entry by setting $V_i := d$.
4. When $\mathcal{D}$ sends (x_0^*, x_1^*), $\mathcal{B}$ generates $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x_0^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_j))$, sets $X_0^* := (x_0^*, x_1^*, \mathsf{K}, j-1, \bot)$ and $X_1^* := (x_0^*, x_1^*, \mathsf{K}, j, \mathsf{skl.ct}^*)$, sends an encryption query (X_0^*, X_1^*) to its challenger, and receives $\mathsf{skfe.ct}^*$. $\mathcal{B}$ generates $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.pk}, (\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot))$ and passes $\mathsf{ct}^* := \mathsf{fe.ct}^*$ to $\mathcal{D}$.
5. $\mathcal{B}$ outputs what $\mathcal{D}$ outputs.

By the definitions of T_{hyb} and T_{emb}, it holds that

$$\begin{aligned} T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, j-1, \bot) &= \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_0^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i)) \\ &= T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, j, \mathsf{skl.ct}^*) \end{aligned}$$

for all $i \in [j, q]$ since $\mathsf{skl.ct}^*$ is an encryption of $f_j(x_0)$. it also holds that

$$\begin{aligned} T_{\mathsf{hyb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, j-1, \bot) &= \mathsf{SKL.Enc}(\mathsf{skl.ek}_i, f_i(x_1^*); \mathsf{F}_{\mathsf{K}}(\mathsf{skl.ek}_i)) \\ &= T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i](x_0^*, x_1^*, \mathsf{K}, j, \mathsf{skl.ct}^*) \end{aligned}$$

for all $i \in [1, j-1]$. Hence, for all $i \in [q]$, it holds that $F_{0,i}(X_0^*) = F_{1,i}(X_1*)$ and $\mathcal{B}$ is an valid adversary for SKFE. If $\mathsf{skfe.ct}^*$ is an encryption of X_0^* and $\mathsf{skfe.sk}_i$ is a functional decryption key for $F_{0,i}$, $\mathcal{B}$ perfectly simulate G_0. If $\mathsf{skfe.ct}^*$ is an encryption of X_1^* and $\mathsf{skfe.sk}_i$ is a functional decryption key for $F_{1,i}$, $\mathcal{B}$ perfectly simulate G_1. This completes the proof. □

Lemma C.5. *If* SKL *IND-KLA, it holds that* $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| = \mathsf{negl}(\lambda)$.

Proof. We focus on the case where the adversary returns a valid $\mathit{fs}\mathcal{K}_j = (\mathsf{fe.sk}_{W,j}, \mathsf{skl.}s\mathcal{K}_j)$, which is the answer to the j-th key query, since $f_j(x_0^*) = f_j(x_1^*)$ must hold if $\mathit{fs}\mathcal{K}_j$ is not returned. Hence $f_j(x_0^*) \neq f_j(x_1^*)$ is allowed in this case.

We construct an adversary $\mathcal{B}$ for SKL by using the distinguisher $\mathcal{D}$ for these two games.

1. $\mathcal{B}$ is given $(\mathsf{skl.ek}^*, \mathsf{skl}.s\mathcal{K}^*)$ and sets $(\mathsf{skl.ek}_j, \mathsf{skl.ek}_j) := (\mathsf{skl.ek}^*, \mathsf{skl}.s\mathcal{K}^*)$.

2. $\mathcal{B}$ generates $(\mathsf{fe.pk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$, and $\mathsf{ske.sk} \leftarrow \{0,1\}^\lambda$, and sends $\mathsf{pk} := \mathsf{fe.pk}$ to $\mathcal{D}$.

3. When $\mathcal{D}$ sends the i-th query f_i to $O_{\mathcal{KG}}$, if $i \neq j$, $\mathcal{B}$ generates $(\mathsf{skl.ek}_i, \mathsf{skl}.s\mathcal{K}_i, \mathsf{skl.vk}_i) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$. For all $i \in [q]$, $\mathcal{B}$ generates $\mathsf{skfe.sk}_i \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, T_{\mathsf{emb}}[f_i, \mathsf{skl.ek}_i, i])$, $\mathsf{ske.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathsf{skfe.sk}_i)$, and $\mathsf{fe.sk}_{W,i} \leftarrow \mathsf{FE.KG}(\mathsf{fe.msk}, W[f_i, \mathsf{skl.ek}_i, \mathsf{ske.ct}_i])$, and returns $fs\mathcal{K}_i := (\mathsf{fe.sk}_{W,i}, \mathsf{skl}.s\mathcal{K}_i)$ to $\mathcal{D}$. Note that $\mathsf{skl}.s\mathcal{K}_j = \mathsf{skl}.s\mathcal{K}$ is given from the challenger. If $i \neq j$, $\mathcal{B}$ adds $(f_i, \mathsf{skl.vk}_i, \bot)$ to $L_{\mathcal{KG}}$. If $i = j$, $\mathcal{B}$ adds $(f_j, \bot, \bot)$ to $L_{\mathcal{KG}}$.

4. When $\mathcal{D}$ sends $(f_i, fs\mathcal{K}'_i)$ to $O_{\mathcal{V}rfy}$, $\mathcal{B}$ finds an entry $(f_i, \mathsf{skl.vk}_i, V_i)$ from $L_{\mathcal{KG}}$ and parses $fs\mathcal{K}'_i = (\mathsf{fe.sk}'_i, \mathsf{skl}.s\mathcal{K}'_i)$.

   - If $f_i \neq f_j$, $\mathcal{B}$ returns $d := \mathsf{SKL}.\mathcal{V}rfy(\mathsf{skl.vk}_i, \mathsf{skl}.s\mathcal{K}'_i)$ since $\mathsf{skl.vk}_i \neq \bot$. If $V_i = \top$ $\mathcal{B}$ does not update the entry. Otherwise, $\mathcal{B}$ updates the entry by setting $V_i := d$.
   - Else if $f_i = f_j$, $\mathcal{B}$ sends $\mathsf{skl}.s\mathcal{K}'_j$ to its challenger ($O_{\mathsf{SKL}.\mathcal{V}rfy}$ of IND-KLA), receives the result d_j, and passes d_j to $\mathcal{D}$. If $V_j = \top$, $\mathcal{B}$ does not update the entry. Otherwise, $\mathcal{B}$ updates the entry by setting $V_j := d_j$.

5. When $\mathcal{D}$ sends (x_0^*, x_1^*), $\mathcal{B}$ generates $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and $\mathsf{K}_{\neq \mathsf{skl.ek}_j} = \mathsf{Puncture}(\mathsf{K}, \mathsf{skl.ek}_j)$, sends (x_0^*, x_1^*) to its challenger, and receives $\mathsf{skl.ct}^* \leftarrow \mathsf{SKL.Enc}(\mathsf{skl.ek}_j, f_j(x^*_{\mathsf{coin}}))$. $\mathcal{B}$ generates $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, (x_0^*, x_1^*, \mathsf{K}_{\neq \mathsf{skl.ek}_j}, j, \mathsf{skl.ct}^*))$ and $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.pk}, (\mathsf{skfe.ct}^*, \mathsf{ske.sk}, \bot))$ and passes $\mathsf{ct}^* := \mathsf{fe.ct}^*$ to $\mathcal{D}$.

6. $\mathcal{B}$ outputs what $\mathcal{D}$ outputs.

It is easy to see that $\mathcal{B}$ perfectly simulates G_3 and G_4 if $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$, respectively. This completes the proof. □