# Public Key Encryption with Secure Key Leasing

Shweta Agrawal[1], Fuyuki Kitagawa[2], Ryo Nishimaki[2], Shota Yamada[3], and Takashi Yamakawa[2]

[1] IIT Madras, Chennai, India
[2] NTT Social Informatics Laboratories, Tokyo, Japan
[3] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

**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[4]. 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

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

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 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 [11], quantum money [35], quantum copy protection [1], one shot signatures [5] 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 [1], 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 [2]. 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 [9], 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 [12] 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 [20] 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 [17] 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 [19]). Very recently, Kitagawa and Nishimaki [27] 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 [3,9] that copy protection is a stronger notion than SSL[5] – 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 [20,17,27] imply our notion of PKE with secure key leasing from their respective assumptions, which all include at least the assumption of iO (see Appendix A of the full version for the detail). Additionally, in the context of public key FE, the only prior work by Kitagawa and Nishimaki [27] 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 [27]. 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. [3] studied

[5] 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.

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 [22], 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 $\mathsf{SKL}$ 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 $\mathcal{KG}$.[6]

– $\mathcal{KG}$ outputs a quantum decryption key $\mathcal{dk}$ instead of a classical decryption key.
– $\mathcal{KG}$ outputs a (secret) verification key $\mathsf{vk}$, 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, $\mathsf{SKL}$ should satisfy verification correctness that states that $\mathcal{Vrfy}(\mathsf{vk}, \mathcal{dk}) = \top$ holds, where $(\mathsf{ek}, \mathcal{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$.

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 $(\mathsf{ek}, \mathcal{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$ and sends $\mathsf{ek}$ and $\mathcal{dk}$ to an adversary $\mathcal{A}$.
2. $\mathcal{A}$ sends two challenge plaintexts $(\mathsf{m}_0^*, \mathsf{m}_1^*)$ and a quantum state $\widetilde{\mathcal{dk}}$ that is supposed to be a correct decryption key. The challenger checks if $\mathcal{Vrfy}(\mathsf{vk}, \widetilde{\mathcal{dk}}) = \top$ holds. If not, $\mathcal{A}$ is regarded as invalid and the game ends here. Otherwise, the game goes to the next step.[7]

[6] 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.

[7] 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 $\widetilde{\mathcal{dk}}$ is $\top$. In this overview, we focus on the "1-query" security for simplicity.

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 $\mathsf{coin}'$.

IND-KLA security guarantees that any QPT $\mathcal{A}$ cannot guess $\mathsf{coin}$ 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 $\left|\Pr\left[\mathsf{coin}' = \mathsf{coin} \mid \mathcal{Vrfy}(\mathsf{vk}, \widetilde{dk}) = \top\right] - 1/2\right| = \mathsf{negl}(\lambda)$.

*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 [4,17]. 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 $\mathsf{Basic} = (\mathsf{Basic}.\mathcal{KG}, \mathsf{Basic}.\mathsf{Enc}, \mathsf{Basic}.\mathcal{Dec}, \mathsf{Basic}.\mathcal{Vrfy})$ 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 $\mathsf{ek} := (\mathsf{ek}_0, \mathsf{ek}_1)$, $dk := 1/\sqrt{2}(|0\rangle|\mathsf{dk}_0\rangle + |1\rangle|\mathsf{dk}_1\rangle)$, and $\mathsf{vk} := (\mathsf{dk}_0, \mathsf{dk}_1)$. Given $\mathsf{m}$ and $\mathsf{ek}$, $\mathsf{Basic.Enc}$ 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 $\mathsf{dk}_0$ and $\mathsf{dk}_1$, respectively, in superposition. Since both decryptions result in the same message $\mathsf{m}$, we can decrypt ciphertexts without collapsing $dk$. Finally, $\mathsf{Basic}.\mathcal{Vrfy}$ checks if the input decryption key is an equal-weight superposition of $\mathsf{dk}_0$ and $\mathsf{dk}_1$. 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 $\Pi_{\mathrm{vrfy}}$.

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 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 $\mathsf{Basic}$ 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 $\mathsf{Basic}$ using CoIC-KLA security. Then, we explain how to achieve a full OW-KLA secure scheme by applying parallel amplification to $\mathsf{Basic}$.

*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 $\mathsf{ek}_0$, $\mathsf{ek}_1$, 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 $(\varPi_{\mathrm{vrfy}}, I - \varPi_{\mathrm{vrfy}})$.
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 $\mathsf{ct}_0$ and $\mathsf{ct}_1$ 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 $b = 0$, $\mathsf{ct}_0$ and $\mathsf{ct}_1$ are ciphertexts of the same plaintext $\mathsf{m}_a^*$. On the other hand, if $b = 1$, $\mathsf{ct}_0$ and $\mathsf{ct}_1$ are ciphertexts of the different plaintexts $\mathsf{m}_a^*$ and $\mathsf{m}_{1 \oplus a}^*$. Thus, we call this security notion consistent or inconsistent security.

*1/2-OW-KLA security of* $\mathsf{Basic}$. We explain how to prove 1/2-OW-KLA security of $\mathsf{Basic}$ based on CoIC-KLA security of $\mathsf{PKE}$. The OW-KLA security game for $\mathsf{Basic}$ 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 $\mathsf{ek}$ 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 $\varPi_{\mathrm{vrfy}}$ 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 $\mathsf{m}^*$ 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 $\varPi_{\mathrm{vrfy}}$ 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 $\widetilde{dk}$ after $\mathcal{A}$ outputs $\mathsf{m}'$.

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 $\mathsf{ct}_1$ 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 $\mathsf{ct}_0$ and $\mathsf{ct}_1$ 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 $\mathsf{PKE}$.

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

**Fact 1:** Assume $\mathsf{PKE}$ 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 $\epsilon$ is the success probability of the original experiment [13, Lemma 2.1].

Suppose $\Pr[S_3] = 1/2 + 1/\mathrm{poly}(\lambda)$ for some polynomial poly. This means that conditioned that $\mathsf{m}' \in \{m^*, \tilde{m}\}$, $\widetilde{dk}$ returned by $\mathcal{A}$ passes the verification with probability significantly greater than 1/2. Thus, if we measure $\widetilde{dk}$ in the computational basis, we obtain $\mathsf{dk}_0$ with some inverse polynomial probability and also $\mathsf{dk}_1$ with some inverse polynomial probability. (If either one is obtained with overwhelming probability, $\widetilde{dk}$ 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 $\mathsf{PKE}$ satisfies CoIC-KLA security, $\mathsf{Basic}$ 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 $\mathsf{Basic}$ in the following way. When generating a key tuple, we generate $\lambda$ key tuples $(\mathsf{ek}_i, \widetilde{dk}_i, \mathsf{vk}_i)$ of $\mathsf{Basic}$ and set $\mathsf{ek}' := (\mathsf{ek}_i)_{i\in[\lambda}$, $\widetilde{dk}' := (\widetilde{dk}_i)_{i\in[\lambda]}$, and $\mathsf{vk}' := (\mathsf{vk}_i)_{i\in[\lambda]}$. When encrypting a plaintext $\mathsf{m}$, we divide it into $\lambda$ pieces $\mathsf{m}_1, \cdots, \mathsf{m}_\lambda$, and encrypt each $\mathsf{m}_i$ using

$\mathsf{ek}_i$. Then decryption and verification are performed naturally by running the underlying procedures in $\mathsf{Basic}$ 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 $\mathsf{Basic}$. 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 $\mathsf{CPFE}$ 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 $\mathsf{mpk}$ and a master secret key $\mathsf{msk}$. $\mathsf{FE.KG}$ takes as input $\mathsf{msk}$ and a string $x$ and outputs a decryption key $\mathsf{sk}_x$ tied to the string $x$. $\mathsf{FE.Enc}$ takes as input $\mathsf{mpk}$ and a description of a circuit $C$ and outputs a ciphertext $\mathsf{ct}$. If we decrypt this ciphertext $\mathsf{ct}$ with $\mathsf{sk}_x$ 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 $\mathsf{sk}_x$ 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 $\mathsf{m}$, $\mathsf{PKE.Enc}$ outputs $\mathsf{FE.Enc}(\mathsf{mpk}, C[\mathsf{m}])$, where $C[\mathsf{m}]$ is the constant circuit that outputs $\mathsf{m}$ on any input. Given $\mathsf{dk} = \mathsf{sk}_x$ and $\mathsf{ct}$, $\mathsf{PKE.Dec}$ simply outputs $\mathsf{CPFE.Dec}(\mathsf{sk}_x, \mathsf{ct})$. We see that $\mathsf{PKE}$ satisfies decryption correctness from that of $\mathsf{CPFE}$.

Before proving CoIC-KLA security of $\mathsf{PKE}$, we explain a nice tracing property of $\mathsf{PKE}$ 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 $\mathsf{PKE.KG}$ 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 $\mathsf{CPFE}$'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 $\mathsf{sk}_x$, 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 [28] 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 [37] that is based on the technique by Marriott and Watrous [30]. 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 $\mathsf{PKE.KG}$, and sends $\mathsf{ek}_0$, $\mathsf{ek}_1$, 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 $\mathsf{PKE}$ using Fact 3. To this end, we introduce one more fact.

**Fact 4:** Given $\mathsf{mpk}_0$, $\mathsf{mpk}_1$, 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 $\mathsf{PKE.KG}$, 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 $\epsilon$ is the success probability of the original experiment [13, Lemma 2.1].

Suppose there exists a QPT adversary $\mathcal{A}$ that breaks CoIC-KLA security of $\mathsf{PKE}$. We consider the following adversary $\mathcal{B}$ using $\mathcal{A}$. Given $\mathsf{mpk}_0$, $\mathsf{mpk}_1$, 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 $\mathit{dk} := 1/\sqrt{2}(|0\rangle|\mathsf{sk}_{x_0}\rangle + |1\rangle|\mathsf{sk}_{x_1}\rangle)$ 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 $\boldsymbol{U}$ 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 $\mathsf{PKE}$ 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.[8] 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 $\mathsf{ABE}$ 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 $\mathsf{mpk}$ and a master secret key $\mathsf{msk}$. $\mathsf{ABE.KG}$ takes as input $\mathsf{msk}$ and a key attribute $y$ and outputs a user secret key $\mathsf{sk}_y$ tied to the attribute $y$. $\mathsf{ABE.Enc}$ takes as input $\mathsf{mpk}$, a ciphertext attribute $x$, and a message $\mathsf{m}$ and outputs a ciphertext $\mathsf{ct}$. 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

[8] 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.

generation algorithm is now quantum and it outputs user secret key $\mathit{usk}_y$ along with verification key $\mathsf{vk}$. We also additionally introduce a verification algorithm that takes $\mathsf{vk}$ and a quantum state $\mathit{usk}'$ and outputs $\top$ if it judges that the user secret key corresponding to $\mathsf{vk}$ 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 $\mathsf{m}$ 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 $\mathsf{SKL}$.

The generation of user individual $\mathsf{SKL}$ ciphertext is somewhat non-trivial since $\mathsf{ABE}$ can only encrypt a single message. In order to achieve this, we use an idea similar to [32,23] that combines encryption with the garbled circuits. In particular, we garble the encryption circuit of $\mathsf{SKL}$ 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 inputs are revealed. Therefore, we are only able to prove 1-bounded distinguishing key security,[9] 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 [21,26] 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

[9] 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.

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 [26], 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 [26], 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.

*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 $\mathsf{FE}$ 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 $\mathsf{pk}$ and a master secret key $\mathsf{msk}$. $\mathsf{FE.KG}$ takes as input $\mathsf{msk}$ and a function $f \in \mathcal{F}$ and outputs a functional decryption key $\mathsf{sk}_f$ tied to the function $f$. $\mathsf{FE.Enc}$ takes as input $\mathsf{pk}$ and a plaintext $x$ and outputs a ciphertext $\mathsf{ct}$. 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 $\mathcal{sk}_f$ along with verification key $\mathsf{vk}$. We also introduce a verification algorithm that takes $\mathsf{vk}$ and a quantum state $\mathcal{sk}'$ and outputs $\top$ if it judges that the functional decryption key corresponding to $\mathsf{vk}$ 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 $(\mathsf{skl.ek}, \mathsf{skl}.\mathpzc{sk}, \mathsf{skl.vk})$ 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 $\mathsf{SKL.Enc}(\mathsf{skl.ek}, f(x))$.[10] A functional decryption key for $f$ consists of $(\mathsf{fe.sk}_W, \mathsf{skl}.\mathpzc{sk})$. A ciphertext of $x$ is a (classical) PKFE ciphertext $\mathsf{FE.Enc}(\mathsf{fe.pk}, x)$. If we return $\mathsf{skl}.\mathpzc{sk}$ 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.

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) [6,14] 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 $\mathpzc{sk}_f = (\mathsf{fe.sk}_W, \mathsf{skl}.\mathpzc{sk})$ for $f$ is returned. This is because $\mathsf{skl}.\mathpzc{sk}$ is returned and we can use IND-KLA security under $\mathsf{skl.ek}$.

## 1.4 Other Related Work

*Quantum Copy Protection.* Aaronson [1] 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. [18] provided a quantum copy-protection scheme for a class of evasive functions in the QROM. Subsequently, Aaronson et al. [3] 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. [17] 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 [19]). Ananth et al. [7,8]

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

also constructed copy protection for point functions, which in turn can be transformed into copy protection for compute-and-compare programs. Sattath and Wyborski [33] 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 [9], 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. [18] improved the security notion achieved by [9] by relying on the QROM, for the same class of evasive functions. Additionally, Kitagawa, Nishimaki and Yamakawa [29] 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. [16] 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 [27], 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 [15] 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 [24,31,10,25]. The root of quantum encryption with certified deletion is revocable quantum time-released encryption by Unruh [34]. 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.

## 2 Preliminaries

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

Let $[\ell]$ denote the set of integers $\{1, \cdots, \ell\}$, $\lambda$ 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 $y \leftarrow \mathscr{A}(\chi)$ denote assigning to $y$ the output of a probabilistic or deterministic algorithm $\mathsf{A}$ and a quantum algorithm $\mathscr{A}$ on an input $x$ and $\chi$, respectively. When we explicitly show that $\mathsf{A}$ 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 $\mathsf{negl}$ 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 $i$th bit of $x$ and $y$, respectively.

*Standard cryptographic tools.* We omit the definitions of standard cryptographic tools including SKE, PKE, ABE, FE, puncturable PRFs, and garbling schemes. See Section 2.1 of the full version for their definitions.

## 3 Public Key Encryption with Secure Key Leasing

We define PKE-SKL and its security notions and show a relationship between them.

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

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

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

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

$\mathscr{Vrfy}(\mathsf{vk}, \widetilde{\mathscr{dk}}) \rightarrow \top/\bot$**:** *The verification algorithm takes a verification key* $\mathsf{vk}$ *and a (possibly malformed) decryption key* $\widetilde{\mathscr{dk}}$*, and outputs* $\top$ *or* $\bot$.

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

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

**Verification correctness:** *We have*

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

*Remark 3.1.* 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 $\mathsf{ct}$ and $\mathscr{dk}$, if we decrypt $\mathsf{ct}$ by using $\mathscr{dk}$, 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 [36]. A similar remark applies to all variants of PKE-SKL (IBE, ABE, and FE with SKL) defined in this paper.

*Remark 3.2.* Though we are the first to define PKE with secure key leasing, SKFE with secure key leasing was already defined by Kitagawa and Nishimaki [27]. 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 two security definitions for PKE-SKL, IND-KLA and OW-KLA security.

**Definition 3.2 (IND-KLA Security).** *We say that a PKE-SKL scheme* $\mathsf{SKL}$ *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* $(\mathsf{ek}, \mathcal{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$ *and sends* $\mathsf{ek}$ *and* $\mathcal{dk}$ *to* $\mathcal{A}$*.*
2. *Throughout the experiment,* $\mathcal{A}$ *can access the following (stateful) verification oracle* $O_{\mathcal{Vrfy}}$ *where* $V$ *is initialized to be* $\bot$*:*

   $O_{\mathcal{Vrfy}}(\widetilde{\mathcal{dk}})$**:** *It runs* $d \leftarrow \mathsf{Vrfy}(\mathsf{vk}, \widetilde{\mathcal{dk}})$ *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* $\mathsf{ct}^*$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* $\mathsf{coin}'$ *for* $\mathsf{coin}$*.* $\mathcal{C}$ *outputs* $\mathsf{coin}'$ *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* $\mathsf{SKL}$ *is 1-query IND-KLA secure if the above holds for any QPT* $\mathcal{A}$ *that makes at most one query to* $O_{\mathcal{Vrfy}}$*.*

*Remark 3.3.* When we consider a 1-query adversary, we can assume that its query is made before receiving the challenge ciphertext $\mathsf{ct}^*$ without loss of generality. This is because otherwise the experiment always outputs 0.

*Remark 3.4.* 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.

**Definition 3.3 (OW-KLA Security).** *We say that a PKE-SKL scheme* $\mathsf{SKL}$ *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* $(\mathsf{ek}, \mathcal{dk}, \mathsf{vk}) \leftarrow \mathcal{KG}(1^\lambda)$ *and sends* $\mathsf{ek}$ *and* $\mathcal{dk}$ *to* $\mathcal{A}$*.*
2. *Throughout the experiment,* $\mathcal{A}$ *can access the following (stateful) verification oracle* $O_{\mathcal{Vrfy}}$ *where* $V$ *is initialized to be* $\perp$*:*

   $O_{\mathcal{Vrfy}}(\widetilde{\mathcal{dk}})$**:** *It runs* $d \leftarrow \mathsf{Vrfy}(\mathsf{vk}, \widetilde{\mathcal{dk}})$ *and returns* $d$*. If* $V = \perp$ *and* $d = \top$*, it updates* $V := \top$*.*
3. $\mathcal{A}$ *sends* $\mathsf{RequestChallenge}$ *to* $\mathcal{C}$*. If* $V = \perp$*,* $\mathcal{C}$ *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* $\mathsf{ct}^*$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* $\mathsf{m}$*.* $\mathcal{C}$ *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* $\mathsf{SKL}$ *is 1-query OW-KLA secure if the above holds for any QPT* $\mathcal{A}$ *that makes at most one query to* $O_{\mathcal{Vrfy}}$*.*

We show the following theorem.

**Theorem 3.1.** *If there exists a 1-query OW-KLA secure PKE-SKL scheme, there exists an IND-KLA secure PKE-SKL scheme.*

Thus, it suffices to construct 1-query OW-KLA secure scheme for constructing IND-KLA secure scheme. The proof is based on quantum Goldreich-Levin lemma with quantum auxiliar inputs [4,17] and goes through an additional security notion called one-more unreturnability (OMUR). See Section 3.2 of the full version for the proof.

## 4 Public Key Encryption with CoIC-KLA Security

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 Sec. 5.

### 4.1 Definition

**Definition 4.1 (CoIC-KLA Security).** *We say that a PKE scheme* $\mathsf{PKE}$ *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 $\widetilde{dk} := \frac{1}{\sqrt{2}}(|0\rangle |\mathsf{dk}_0\rangle + |1\rangle |\mathsf{dk}_1\rangle)$. $\mathcal{C}$ sends $\mathsf{ek}_0$, $\mathsf{ek}_1$, and $\widetilde{dk}$ to $\mathcal{A}$. $\mathcal{A}$ can get access to the following oracle only once.*

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

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

   *It returns the measurement outcome (indicating whether the state was projected onto $\Pi_{\mathrm{vrfy}}$ 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 $\mathsf{ct}_0^*$ and $\mathsf{ct}_1^*$ 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*

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

## 4.2 Construction

We construct a CoIC-KLA secure PKE $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ using a 1-key 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 $\mathsf{m}$ 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 $\mathsf{PKE}$ follows from that of $\mathsf{CPFE}$. We show the following theorem.

**Theorem 4.1.** *If $\mathsf{CPFE}$ is 1-key secure, then $\mathsf{PKE}$ is CoIC-KLA secure.*

In the full version, we actually prove that $\mathsf{PKE}$ satisfies a security notion called strong CoIC-KLA security, which implies CoIC-KLA security. See Section 4.3 of the full version for the proof.

Since 1-key CPFE exists if IND-CPA secure PKE exists, the above theorem implies the following theorem.

**Theorem 4.2.** *If there is an IND-CPA secure PKE scheme, then there is a CoIC-KLA secure PKE scheme.*

## 5 Construction of PKE-SKL

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 $\ell = \omega(\log \lambda)$. Then, we construct a PKE-SKL scheme $(\mathsf{SKL}.\mathcal{KG}, \mathsf{SKL.Enc}, \mathsf{SKL}.\mathcal{Dec}, \mathsf{SKL}.\mathcal{Vrfy})$ 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

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

and a verification key

$$\mathsf{vk} := \{\mathsf{cPKE.dk}_{i,b}\}_{i\in[\lambda], b\in\{0,1\}}.$$

$\mathsf{SKL.Enc}(\mathsf{ek}, \mathsf{m})$**:**
- Parse $\mathsf{ek} = \{\mathsf{cPKE.ek}_{i,b}\}_{i\in[\lambda], b\in\{0,1\}}$ and $\mathsf{m} = \mathsf{m}_1 \| \ldots \| \mathsf{m}_\lambda$ 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 $\mathsf{ct} := \{\mathsf{cPKE.ct}_{i,b}\}_{i\in[\lambda], b\in\{0,1\}}$.

$\mathsf{SKL}.\mathcal{Dec}(\mathcal{dk}, \mathsf{ct})$**:**
- Parse $\mathcal{dk} = \bigotimes_{i\in[\lambda]} \mathcal{dk}_i$ and $\mathsf{ct} = \{\mathsf{cPKE.ct}_{i,b}\}_{i\in[\lambda], b\in\{0,1\}}$.
- Let $U_{\mathrm{dec}}$ be a unitary such that for all $\mathsf{cPKE.dk}'$, $\mathsf{cPKE.ct}'_0$, and $\mathsf{cPKE.ct}'_1$:

$$|b\rangle \left|\mathsf{cPKE.dk}'\right\rangle \left|\mathsf{cPKE.ct}'_0, \mathsf{cPKE.ct}'_1\right\rangle |0\rangle$$
$$\xrightarrow{U_{\mathrm{dec}}} |b\rangle \left|\mathsf{cPKE.dk}'\right\rangle \left|\mathsf{cPKE.ct}'_0, \mathsf{cPKE.ct}'_1\right\rangle \left|\mathsf{cPKE.Dec}(\mathsf{cPKE.dk}', \mathsf{cPKE.ct}'_b)\right\rangle$$

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

$$U_{\mathrm{dec}}\left(\widetilde{dk}_i \otimes |\mathsf{cPKE.ct}_{i,0}, \mathsf{cPKE.ct}_{i,1}\rangle\langle \mathsf{cPKE.ct}_{i,0}, \mathsf{cPKE.ct}_{i,1}| \otimes |0\rangle\langle 0|\right) U_{\mathrm{dec}}^{\dagger},$$

measure the rightmost register, and let $\mathsf{m}'_i$ be the measurement outcome.

– Output $\mathsf{m}' \coloneqq \mathsf{m}'_1 \| \ldots \| \mathsf{m}'_\lambda$.

$\mathsf{SKL}.\mathcal{Vrfy}(\mathsf{vk}, \widetilde{dk})$:

– Parse $\mathsf{vk} = \{\mathsf{cPKE.dk}_{i,b}\}_{i \in [\lambda], b \in \{0,1\}}$.

– Apply a binary-outcome measurement $(\boldsymbol{I} - \Pi_{\mathrm{vrfy}}^{\mathsf{vk}}, \Pi_{\mathrm{vrfy}}^{\mathsf{vk}})$ on $\widetilde{dk}$ where $\Pi_{\mathrm{vrfy}}^{\mathsf{vk}}$ is the projection onto the right decryption key, i.e.,

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

If the measurement outcome is 1 (indicating that the state was projected onto $\Pi_{\mathrm{vrfy}}^{\mathsf{vk}}$), 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 5.1.** *If* cPKE *is CoIC-KLA secure, then* SKL *is 1-query OW-KLA secure.*

See Section 5 of the full version for the proof.

By combining Theorems 3.1, 4.2 and 5.1, we obtain the following theorem.

**Theorem 5.2.** *If there is an IND-CPA secure PKE scheme, then there is an IND-KLA secure PKE-SKL scheme.*

## 6 Attribute-based Encryption with Secure Key Leasing

### 6.1 Definitions

The syntax of ABE-SKL and its security are defined as follows.

**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* $1^\lambda$, *and outputs a public key* pk *and master secret key* msk.

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

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

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

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

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

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

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

**Definition 6.2 (Adaptive/Selective Indistinguishability against Key Leasing Attacks).** *We say that an ABE-SKL scheme* $\mathsf{ABE\text{-}SKL}$ *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* $(y, \mathsf{vk}, V)$ *from* $L_{\mathcal{KG}}$*. If there is such an entry, it returns* $\bot$*. Otherwise, it generates* $(\mathcal{usk}, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y)$*, sends* $\mathcal{usk}$ *to* $\mathcal{A}$*, and adds* $(y, \mathsf{vk}, \bot)$ *to* $L_{\mathcal{KG}}$*.*

   $O_{\mathcal{Vrfy}}(y, \mathcal{usk}')$**:** *Given* $(y, \mathcal{usk}')$*, it finds an entry* $(y, \mathsf{vk}, V)$ *from* $L_{\mathcal{KG}}$*. (If there is no such entry, it returns* $\bot$*.) It then runs* $d := \mathcal{Vrfy}(\mathsf{vk}, \mathcal{usk}')$ *and returns* $d$ *to* $\mathcal{A}$*. If* $V = \bot$*, it updates the entry into* $(y, \mathsf{vk}, d)$*.*
2. *When* $\mathcal{A}$ *sends* $(x^*, m_0, m_1)$ *to the challenger, the challenger checks if for any entry* $(y, \mathsf{vk}, V)$ *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* $\mathsf{ct}^*$ *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* $\mathsf{coin}'$ *for* $\mathsf{coin}$*. The challenger outputs* $\mathsf{coin}'$ *as the final output of the experiment.*

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

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ada\text{-}lessor}}_{\mathsf{PKFE\text{-}SKL},\mathcal{A}}(\lambda) :=& \left|\Pr\Big[\mathsf{Exp}^{\mathsf{ada\text{-}lessor}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(1^\lambda, 0) \to 1\Big] - \Pr\Big[\mathsf{Exp}^{\mathsf{ada\text{-}lessor}}_{\mathsf{ABE\text{-}SKL},\mathcal{A}}(1^\lambda, 1) \to 1\Big]\right|\\ &\le \mathsf{negl}(\lambda).\end{aligned}$$

*We say that* $\mathsf{ABE\text{-}SKL}$ *is secure against selective indistinguishability against key leasing attacks (Sel-IND-KLA) if the above holds for all QPT adversaries that declare* $x^*$ *at the beginning of the experiment.*

*Remark 6.1.* In Definition 6.2, the key generation oracle returns $\perp$ 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 the following security notion where we introduce additional restriction that the number of distinguishing keys that are issued (and eventually returned) before $\mathsf{ct}^*$ 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 6.3 (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 $(y, \mathsf{vk}, V)$ 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 $\mathsf{ct}^*$ 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 $\mathsf{SKL}$ 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 $\mathsf{GC}$ 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 $(\mathsf{skl.ek}, \mathsf{skl}.\mathcal{dk}, \mathsf{skl.vk}) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$.
- 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 $\mathsf{skl.ek}$.
- Output $\mathcal{usk} := (\{\mathsf{abe.sk}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \mathsf{skl.ek}, \mathsf{skl}.\mathcal{dk})$ 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 $\mathsf{skl.ek}$ of $\mathsf{SKL}$ 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 $\mathsf{ct} := (\{\mathsf{abe.ct}_{i,b}\}_{i\in[\ell_{\mathsf{ek}}],b\in\{0,1\}}, \widetilde{E})$.

$\mathcal{D}ec(\mathcal{usk}, x, \mathsf{ct})$**:**
- Parse $\mathcal{usk} = (\{\mathsf{abe.sk}_i\}_{i\in[\ell_{\mathsf{ek}}]}, \mathsf{skl.ek}, \mathsf{skl}.\mathcal{dk})$ 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 $m' \leftarrow \mathsf{SKL}.\mathcal{D}ec(\mathsf{skl}.\mathcal{dk}, \mathsf{skl.ct})$.

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

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 $\mathsf{skl.ek}$ by the correctness of $\mathsf{ABE}$. Therefore, $\mathsf{skl.ct}$ recovered by the garbled circuit evaluation equals to $\mathsf{SKL.Enc}(\mathsf{skl.ek}, m; \mathsf{R})$ by the correctness of $\mathsf{GC}$. Then, the message $m$ is recovered in the last step by the correctness of $\mathsf{SKL}$. We can also see that the verification correctness follows from that of $\mathsf{SKL}$.

**Theorem 6.1.** *If* $\mathsf{ABE}$ *is adaptively (resp., selectively) secure,* $\mathsf{GC}$ *is secure, and* $\mathsf{SKL}$ *is IND-KLA secure, then* $\mathsf{1ABE}$ *above is* 1*-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) secure.*

See Section 6.2 of the full version for the proof.

## 6.3 $Q$-Bounded Distinguishing Key Construction

By using the technique of [26], we can upgrade 1-bounded distinguishing key security to $Q$-bounded distinguishing key security for any polynomial $Q = Q(\lambda)$. Then we obtain the following theorem.

**Theorem 6.2.** *If there exists an adaptively (resp., selectively) secure an ABE scheme for relation $R$, then for any polynomial $Q = Q(\lambda)$, there is a $Q$-bounded distinguishing key Ada-IND-KLA (resp., Sel-IND-KLA) secure ABE scheme for relation $R$.*

See Section 6.3 of the full version for the proof. We remark that Theorem 6.2 preserves the relation $R$. Thus, this in particular gives a compiler that upgrades (selectively or adaptively secure) normal IBE into IBE-SKL.

# 7 Functional Encryption with Secure Key Leasing

## 7.1 Definitions

The syntax of FE-SKL is defined as follows.

**Definition 7.1 (PKFE with Secure Key Leasing).** *A PKFE-SKL scheme* $\mathsf{PKFE\text{-}SKL}$ *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* $\mathsf{PKFE\text{-}SKL}$, *respectively.*

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

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

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

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

$\mathcal{Vrfy}(\mathsf{vk}, \mathit{fsk}') \to \top/\bot$**:** *The verification algorithm takes a verification key* $\mathsf{vk}$ *and a quantum state* $\mathit{fsk}'$, *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 7.1.* Although Kitagawa and Nishimaki [27] 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 also has a classical certificate generation algorithm.

**Definition 7.2 (Adaptive Indistinguishability against Key Leasing Attacks).** *We say that a PKFE-SKL scheme* $\mathsf{PKFE\text{-}SKL}$ *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* $(f, \mathsf{vk}, V)$ *from* $L_{\mathcal{KG}}$. *If there is such an entry, it returns* $\bot$. *Otherwise, it generates* $(\mathit{fsk}, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, f)$, *sends* $\mathit{fsk}$ *to* $\mathcal{A}$, *and adds* $(f, \mathsf{vk}, \bot)$ *to* $L_{\mathcal{KG}}$.

$O_{\mathcal{Vrfy}}(f, \mathit{fsk}')$**:** *Given $(f, \mathit{fsk}')$, it finds an entry $(f, \mathsf{vk}, V)$ from $L_{\mathcal{KG}}$. (If there is no such entry, it returns $\bot$.) It computes $d \leftarrow \mathcal{Vrfy}(\mathsf{vk}, \mathit{fsk}')$ and sends $d$ to $\mathcal{A}$. If $V = \top$, it does not update $L_{\mathcal{KG}}$. Else if $V = \bot$, 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 $(f, \mathsf{vk}, V)$ 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 $\mathsf{ct}^*$ 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 $\mathsf{coin}'$ for $\mathsf{coin}$. The challenger outputs $\mathsf{coin}'$ as the final output of the experiment.*

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

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

*Remark 7.2.* Definition 7.2 assumes that the adversary does not get more than one decryption key for the same $f$ for simplification as Remark 6.1.

## 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 $\mathsf{PKFE\text{-}SKL}$, $|\mathsf{skfe.ct}|$ is the ciphertext length of $\mathsf{SKFE}$, and $|\mathsf{ske.ct}|$ is the ciphertext length of $\mathsf{SKE}$.

$\mathsf{Setup}(1^\lambda)$**:**
- 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 $(\mathsf{skl.ek}, \mathsf{skl.}\mathit{sk}, \mathsf{skl.vk}) \leftarrow \mathsf{SKL}.\mathcal{KG}(1^\lambda)$.
- 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 $\mathit{fsk} := (\mathsf{fe.sk}_W, \mathsf{skl.}\mathit{sk})$ and $\mathsf{vk} := \mathsf{skl.vk}$.

$\mathsf{Enc}(\mathsf{pk}, x)$**:**
- Choose $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.

```
    – Compute $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.pk}, (x \| 0^{\ell_{\mathsf{pad}}}, \bot, \mathsf{K}))$.
    – Output $\mathsf{ct} := \mathsf{fe.ct}$.
$\mathcal{Dec}(\mathit{fsk}, \mathsf{ct})$:
    – 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 $y \leftarrow \mathsf{SKL}.\mathcal{Dec}(\mathsf{skl}.\mathit{sk}, \mathsf{skl.ct})$.
$\mathcal{Vrfy}(\mathsf{vk}, \mathit{fsk}')$:
    – Parse $\mathsf{vk} = \mathsf{skl.vk}$ and $\mathit{fsk}' = (\mathsf{fe.sk}', \mathsf{skl}.\mathit{sk}')$.
    – Compute and output $\mathsf{SKL}.\mathcal{Vrfy}(\mathsf{skl.vk}, \mathsf{skl}.\mathit{sk}')$.
```

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

Constants: Function $f$, PKE-SKL encryption key $\mathsf{skl.ek}$, SKE ciphertext $\mathsf{ske.ct}$.
Input: Plaintext $x'$, SKE key $\mathsf{ske.sk}$, PRF key $\mathsf{K}$.

 1. If $\mathsf{ske.sk} = \bot$, do the following:
      – Parse $x' = x \| \overline{x}$ such that $|\overline{x}| = \ell_{\mathsf{pad}}$.
      – Compute and output $\mathsf{skl.ct} := \mathsf{SKL.Enc}(\mathsf{skl.ek}, f(x); \mathsf{F_K}(\mathsf{skl.ek}))$.
 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 $z := \mathsf{SKFE.Dec}(\mathsf{skfe.sk}, x')$.
```

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

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. We prove the security of PKFE-SKL.

**Theorem 7.1.** *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 7.2.** *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 poofs are given in Section 7.3 of the full version.

## Acknowledgement

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

1. Aaronson, S.: Quantum copy-protection and quantum money. In: 2009 24th Annual IEEE Conference on Computational Complexity. pp. 229–242. IEEE (2009). https://doi.org/10.1109/ccc.2009.42
2. Aaronson, S., Christiano, P.: Quantum money from hidden subspaces. In: Karloff, H.J., Pitassi, T. (eds.) 44th ACM STOC. pp. 41–60. ACM Press (May 2012). https://doi.org/10.1145/2213977.2213983
3. Aaronson, S., Liu, J., Liu, Q., Zhandry, M., Zhang, R.: New approaches for quantum copy-protection. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 526–555. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_19
4. Adcock, M., Cleve, R.: A quantum goldreich-levin theorem with cryptographic applications. In: Alt, H., Ferreira, A. (eds.) STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2285, pp. 323–334. Springer (2002). https://doi.org/10.1007/3-540-45841-7_26
5. Amos, R., Georgiou, M., Kiayias, A., Zhandry, M.: One-shot signatures and applications to hybrid quantum/classical authentication. In: Makarychev, K., Makarychev, Y., Tulsiani, M., Kamath, G., Chuzhoy, J. (eds.) 52nd ACM STOC. pp. 255–268. ACM Press (Jun 2020). https://doi.org/10.1145/3357713.3384304
6. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_32
7. Ananth, P., Kaleoglu, F.: Unclonable encryption, revisited. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part I. LNCS, vol. 13042, pp. 299–329. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90459-3_11
8. Ananth, P., Kaleoglu, F., Li, X., Liu, Q., Zhandry, M.: On the feasibility of unclonable encryption, and more. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 212–241. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_8
9. Ananth, P., La Placa, R.L.: Secure software leasing. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 501–530. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77886-6_17
10. Bartusek, J., Khurana, D.: Cryptography with certified deletion. Cryptology ePrint Archive, Report 2022/1178 (2022), https://eprint.iacr.org/2022/1178
11. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. arXiv preprint arXiv:2003.06557 (2020). https://doi.org/10.1016/j.tcs.2014.05.025
12. Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certificates and security capabilities. In: Wallach, D.S. (ed.) USENIX Security 2001. USENIX Association (Aug 2001)

13. Boneh, D., Zhandry, M.: Secure signatures and chosen ciphertext security in a quantum computing world. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 361–379. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_21
14. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. Journal of Cryptology **31**(1), 202–225 (Jan 2018). https://doi.org/10.1007/s00145-017-9255-y
15. Broadbent, A., Islam, R.: Quantum encryption with certified deletion. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 92–122. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64381-2_4
16. Broadbent, A., Jeffery, S., Lord, S., Podder, S., Sundaram, A.: Secure software leasing without assumptions. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part I. LNCS, vol. 13042, pp. 90–120. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90459-3_4
17. Coladangelo, A., Liu, J., Liu, Q., Zhandry, M.: Hidden cosets and applications to unclonable cryptography. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 556–584. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_20
18. Coladangelo, A., Majenz, C., Poremba, A.: Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. arXiv (CoRR) **abs/2009.13865** (2020), https://arxiv.org/abs/2009.13865
19. Culf, E., Vidick, T.: A monogamy-of-entanglement game for subspace coset states. Quantum **6**, 791 (sep 2022). https://doi.org/10.22331/q-2022-09-01-791
20. Georgiou, M., Zhandry, M.: Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877 (2020), https://eprint.iacr.org/2020/877
21. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_11
22. Goyal, R., Kim, S., Manohar, N., Waters, B., Wu, D.J.: Watermarking public-key cryptographic primitives. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 367–398. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_12
23. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_14
24. Hiroka, T., Morimae, T., Nishimaki, R., Yamakawa, T.: Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 606–636. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_21
25. Hiroka, T., Morimae, T., Nishimaki, R., Yamakawa, T.: Certified everlasting functional encryption. Cryptology ePrint Archive, Report 2022/969 (2022), https://eprint.iacr.org/2022/969
26. Itkis, G., Shen, E., Varia, M., Wilson, D., Yerukhimovich, A.: Bounded-collusion attribute-based encryption from minimal assumptions. In: Fehr, S. (ed.) PKC 2017, Part II. LNCS, vol. 10175, pp. 67–87. Springer, Heidelberg (Mar 2017). https://doi.org/10.1007/978-3-662-54388-7_3
27. Kitagawa, F., Nishimaki, R.: Functional encryption with secure key leasing. Asiacrypt 2022 (to appear) (2022)

28. Kitagawa, F., Nishimaki, R.: Watermarking PRFs against quantum adversaries. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 488–518. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_18
29. Kitagawa, F., Nishimaki, R., Yamakawa, T.: Secure software leasing from standard assumptions. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part I. LNCS, vol. 13042, pp. 31–61. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90459-3_2
30. Marriott, C., Watrous, J.: Quantum arthur-merlin games. Comput. Complex. **14**(2), 122–152 (2005). https://doi.org/10.1007/s00037-005-0194-x
31. Poremba, A.: Quantum proofs of deletion for learning with errors. In: Kalai, Y.T. (ed.) 14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA. LIPIcs, vol. 251, pp. 90:1–90:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.ITCS.2023.90, https://doi.org/10.4230/LIPIcs.ITCS.2023.90
32. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010. pp. 463–472. ACM Press (Oct 2010). https://doi.org/10.1145/1866307.1866359
33. Sattath, O., Wyborski, S.: Uncloneable decryptors from quantum copy-protection. arXiv (CoRR) **abs/2203.05866** (2022). https://doi.org/10.48550/arXiv.2203.05866, https://doi.org/10.48550/arXiv.2203.05866
34. Unruh, D.: Revocable quantum timed-release encryption. J. ACM **62**(6), 49:1–49:76 (2015)
35. Wiesner, S.: Conjugate coding. ACM Sigact News **15**(1), 78–88 (1983). https://doi.org/10.1145/1008908.1008920
36. Winter, A.J.: Coding theorem and strong converse for quantum channels. IEEE Trans. Inf. Theory **45**(7), 2481–2485 (1999). https://doi.org/10.1109/18.796385
37. Zhandry, M.: Schrödinger's pirate: How to trace a quantum decoder. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 61–91. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64381-2_3