PKE and ABE with Collusion-Resistant Secure Key Leasing

Fuyuki Kitagawa[†][◇], Ryo Nishimaki[†][◇], Nikhil Pappu[*][†][⋆]

[†]NTT Social Informatics Laboratories, Tokyo, Japan

{fuyuki.kitagawa,ryo.nishimaki}@ntt.com

[◇]NTT Research Center for Theoretical Quantum Information, Atsugi, Japan

[⋆]Portland State University, USA

nikpappu@pdx.edu

February 23, 2025

Abstract

Secure key leasing (SKL) is an advanced encryption functionality that allows a secret key holder to generate a quantum decryption key and securely lease it to a user. Once the user returns the quantum decryption key (or provides a classical certificate confirming its deletion), they lose their decryption capability. Previous works on public key encryption with SKL (PKE-SKL) have only considered the single-key security model, where the adversary receives at most one quantum decryption key. However, this model does not accurately reflect real-world applications of PKE-SKL. To address this limitation, we introduce *collusion-resistant security* for PKE-SKL (denoted as PKE-CR-SKL). In this model, the adversary can adaptively obtain multiple quantum decryption keys and access a verification oracle which validates the correctness of queried quantum decryption keys. Importantly, the size of the public key and ciphertexts must remain independent of the total number of generated quantum decryption keys. We present the following constructions:

- A PKE-CR-SKL scheme based on the learning with errors (LWE) assumption.
- An attribute-based encryption scheme with collusion-resistant SKL (ABE-CR-SKL), also based on the LWE assumption.
- An ABE-CR-SKL scheme with classical certificates, relying on multi-input ABE with polynomial arity.

[*]Part of this work was done while visiting NTT Social Informatics Laboratories as an internship.

[†]Supported by the US National Science Foundation (NSF) via Fang Song's Career Award (CCF-2054758).

Contents

1 Introduction

Secure key leasing. Encryption with secure key leasing (SKL) enables a secret key holder to generate a quantum decryption key and lease it securely to another party. Once the lessee returns the quantum decryption key, they lose their ability to decrypt ciphertexts. Since its introduction by Kitagawa and Nishimaki for secret key functional encryption (SKFE) [KN22], SKL has been extensively studied [AKN$^+$23, APV23, CGJL23, MPY23, AHH24, BGK$^+$24, KMY24] due to its strong security guarantee and practical applications.

Collusion-resistant SKL. Most prior works [AKN$^+$23, APV23, CGJL23, MPY23, AHH24] study how to achieve public-key encryption (PKE) with SKL schemes from standard cryptographic assumptions. All prior works on PKE-SKL have explored the setting where an adversary can obtain *only one* quantum decryption key. However, this single-key security model does not accurately reflect real-world scenarios. In practice, once a lessee returns their decryption key and it is verified (i.e., after revocation), the lessor may lease another decryption key, even to the same lessee. Moreover, in realistic settings, a single secret key holder may need to generate and lease *multiple* quantum decryption keys to various entities. To accurately capture this setting, we consider adversaries capable of obtaining *an unbounded number of quantum decryption keys*, even in the standard PKE setting. We define this model as *collusion-resistant* SKL.

Previous works [APV23, AHH24] presented delegation tasks as a compelling application like the following: Consider a scenario where a system administrator unexpectedly needs to take leave and must temporarily assign their responsibilities—including access to encrypted data—to a colleague by providing decryption keys. In such cases, the single-key security model is inadequate, as the administrator may need to take leave multiple times or assign their responsibilities to different colleagues. Another potential application of collusion-resistant PKE-SKL is in streaming services. Encrypted videos are made accessible to subscribers through quantum decryption keys. When a user unsubscribes, they return their leased keys, losing access to the content, and their subscription fees are canceled. By utilizing an attribute-based encryption [SW05] variant of collusion-resistant PKE-SKL, it becomes possible to precisely control video access based on user attributes, such as location-restrictions or premium and basic subscription plans.

Our goal: collusion-resistant SKL from weaker assumptions. The notion of collusion-resistant SKL is both natural and compelling. However, it has yet to be achieved from well-established assumptions. All known PKE with SKL schemes based on standard assumptions [AKN$^+$23, APV23, CGJL23, AHH24, KMY24] do not remain secure in the collusion-resistant setting. While some existing constructions seem to imply collusion-resistant SKL, such as public-key functional encryption (PKFE) schemes with SKL [AKN$^+$23, BGK$^+$24] and collusion-resistant single decryptor encryption (SDE)[1] [ÇG24], they rely on strong assumptions. These include post-quantum secure indistinguishability obfuscation (IO) or collusion-resistant PKFE, which in turn implies IO, albeit with a sub-exponential security loss [BV18, AJ15, AJS15]. Achieving them from well-established assumptions still remains elusive. In this work, we aim to construct the first collusion-resistant SKL schemes based on weaker assumptions.

1.1 Our Results

Our main contributions are summarized as follows:

1. *Definition of collusion-resistant PKE-SKL (PKE-CR-SKL):* We formally define PKE-CR-SKL, ensuring that if all quantum decryption keys are returned and successfully verified, users lose decryption capabilities. We extend the indistinguishability against key leasing attacks (IND-KLA) security definition [AKN$^+$23] to the collusion-resistant setting. One notable feature of this definition is that the adversary can send multiple queries to the verification oracle, which confirms the validity of returned decryption keys. Another important feature is that the public key and ciphertext size are independent of the number of leased decryption keys (up to logarithmic factors).

2. *Construction of IND-KLA secure PKE with collusion-resistant SKL:* We propose an IND-KLA secure PKE-CR-SKL scheme based on the learning with errors (LWE) assumption.

[1] In short, SDE is PKE where the decryption keys are copy-protected (i.e., unclonable). In general, SDE implies encryption with SKL (see the discussion by Agrawal et al. [AKN$^+$23] for details).

Table 1: Comparison of SKL. VO means security in the presence of the verification oracle for certificates.

	Primitive	Collusion-resistant SKL	VO	Certificate	Assumption
[AKN+23]	PKE	–	✓	quantum	PKE
[AKN+23]	ABE	bounded	✓	quantum	ABE
[AKN+23]	bCR PKFE[a]	bounded	✓	quantum	PKE
[AKN+23]	PKFE	✓	✓	quantum	PKFE[b]
[BGK+24]	PKFE[c]	✓	✓[d]	classical	IO
[APV23, AHH24]	PKE (FHE)	–	–	classical	LWE
[CGJL23]	PKE (FHE)	–	–	classical	LWE
[KMY24]	PKE	–	✓[e]	classical	PKE
Ours1	PKE	✓	✓	quantum	LWE
Ours2	ABE[c]	✓	✓	quantum	LWE
Ours3	PKE	✓	✓	classical	MI-ABE
Ours4	ABE[c]	✓	✓	classical	MI-ABE

a bCR denotes bounded collusion-resistant. This scheme only satisfies the bounded collusion resistance of standard PKFE.
b Collusion-resistant PKFE implies IO up to sub-exponential security loss.
c These schemes are selectively secure, where adversaries must declare the target plaintexts (PKFE case) or attributes (ABE case) at the beginning.
d This scheme has public verifiability (the verification key for certificates is public).
e This scheme is secure even if the verification key is revealed to the adversary *after* the adversary outputs a valid certificate.

3. *Attribute-based encryption with collusion-resistant SKL:* We construct an ABE-CR-SKL scheme, also based on the LWE assumption. Since PKE is a special case of ABE, ABE-CR-SKL trivially implies PKE-CR-SKL. We first present the PKE-CR-SKL scheme separately to provide a clearer foundation for understanding the ABE-CR-SKL construction.

4. *PKE and ABE with collusion-resistant SKL and classical certificates:* We also propose an IND-KLA secure ABE-CR-SKL scheme that utilizes *classical* certificates, whereas the constructions above rely on quantum certificates.[2] In this model, a classical certificate can be derived from a leased quantum decryption key, and successful verification guarantees security. Our scheme is based on multi-input ABE (MI-ABE), which is a potentially weaker assumption than collusion-resistant PKFE. We specify required properties for MI-ABE and discuss the relationship between MI-ABE and other primitives in Section 8.1. ABE-CR-SKL with classical certificates trivially implies PKE-CR-SKL with classical certificates.

We introduce fascinating techniques to achieve our results. These include the classical decryption property and the notion of key-testability for secret key encryption with collusion-resistant SKL (SKE-CR-SKL) and secret key functional encryption with collusion-resistant SKL (SKFE-CR-SKL). Then, we transform SKE-CR-SKL (resp. SKFE-CR-SKL) into PKE-CR-SKL (resp. ABE-CR-SKL) by using classical ABE and compute-and-compare obfuscation [WZ17, GKW17]. In these transformations, we use decryption keys of SKE-CR-SKL (resp. SKFE-CR-SKL) as key attributes of ABE in a superposition way. See Section 2 for the details.

1.2 Related Work

Encryption with SKL. Agrawal et al. [AKN+23] presented PKE-SKL, ABE-SKL, and PKFE-SKL schemes. Their PKE-SKL scheme is based on standard PKE, and its certificate is quantum. This scheme is not collusion-resistant. Their ABE-SKL scheme is based on PKE-SKL and standard (collusion-resistant) ABE, and its certificates are ones of the underlying PKE-SKL. This scheme is bounded collusion-resistant, that is, the adversary can obtain an a-priori bounded number of quantum decryption keys that can decrypt target ciphertexts. Their PKFE-SKL scheme is based on PKE-SKL and standard (collusion-resistant) PKFE, and its certificates are ones of the underlying PKE-SKL. This scheme is collusion-resistant. (If we instantiate the PKFE-SKL scheme with bounded collusion-resistant PKFE, the

[2]Quantum decryption keys function as quantum certificates.

scheme is bounded collusion-resistant with respect to both standard PKFE and SKL.) All schemes mentioned above are secure in the presence of the verification oracle for certificates.

Bartusek et al. [BGK$^+$24] also presented a PKFE-SKL scheme. Their scheme is collusion-resistant in the selective model, where the target plaintext is declared at the beginning of the game. In addition, their certificates are classical and publicly verifiable. However, their scheme relies on IO.

Ananth, Poremba, and Vaikuntanathan [APV23] presented a PKE-SKL scheme with classical certificates based on the LWE assumption and an unproven conjecture. Since the encryption algorithm of this scheme is essentially the same as the Dual-Regev PKE scheme [GPV08], their scheme achieves fully homomorphic encryption (FHE) with SKL. Later, Ananth, Hu, and Huang [AHH24] present a new security analysis for Ananth et al.'s scheme and removed the conjecture. Chardouvelis, Goyal, Jain, and Liu [CGJL23] presented a PKE-SKL scheme with classical certificates based on the LWE assumption. Their scheme is based on the Regev PKE scheme [Reg09], so it also achieves FHE with SKL. In addition, their scheme also achieves classical communication, where all messages between senders and receivers are classical. Kitagawa, Morimae, and Yamakawa [KMY24] presented a PKE-SKL scheme with classical certificates based on PKE. These three works do not achieve collusion resistance. Ananth et al. [APV23, AHH24] and Chardouvelis et al. [CGJL23] do not consider the verification oracle in their security definitions, while Kitagawa et al. [KMY24] does.[3] We summarize the works on encryption with SKL in Table 1.

Secure software leasing. Ananth and La Placa [AL21] introduced secure software leasing, which encodes classical programs into quantum programs that we can securely lease. We can view SKL as secure software leasing for decryption functions. However, previous works on secure software leasing consider a sub-class of evasive functions [AL21, CMP20, KNY21, BJL$^+$21] or PRFs [KNY21], which do not support decryption functions. Moreover, they consider a weak security model in which pirated programs use *honest* evaluation algorithms [AL21, KN23, BJL$^+$21] or rely on the quantum random oracle model [CMP20]. Bartusek et al. [BGK$^+$24] achieve secure software leasing supporting all differing inputs circuits[4] in a strong security model where pirated programs can use arbitrary evaluation algorithms. However, their scheme relies on IO.

Multi-input ABE. Roughly speaking, MI-ABE is ABE that can support multiple ciphertext-attributes (or multiple key-attributes). To achieve our classical certificates scheme, we need MI-ABE for polynomial-size circuits where the number of slots is polynomial, and we can generate ciphertexts for one slot using a public key. See Definition 8.1 for the definition. However, as we review previous works on MI-ABE for general circuits[5] below, none of them achieves what we need without using IO.[6]

Agrawal, Yadav, and Yamada [AYY22] proposed *two-input* ABE for polynomial-size circuits based on lattices. However, the scheme is heuristic (no reduction-based security proof) and needs a master *secret key to generate ciphertexts for all slots*. Agrawal, Rossi, Yadav, and Yamada [ARYY23] proposed MI-ABE for NC^1 from the evasive LWE assumption and MI-ABE for polynomial-size circuits from the evasive LWE and tensor LWE assumptions. Their scheme allows to generate ciphertexts for one slot using a public key. However, the number of slots is *constant*. Agrawal, Kumari, and Yamada [AKY24] proposed MI-ABE[7] for polynomial-size circuits based on the evasive LWE assumption. In their scheme, the number of slots is polynomial. However, we need a master *secret key to generate ciphertexts for all input-attributes*.

Broadcast encryption. Broadcast encryption [FN94] enables a sender to generate ciphertexts intended for a specific subset of users. Only the designated users can decrypt the ciphertexts, while even if all other users collude, they cannot recover the message. A key performance metric for broadcast encryption is the size of the public key and ciphertexts. Several works have proposed optimal broadcast encryption schemes, where these sizes are $\mathrm{poly}(\log N)$ and N is the total number of users. The constructions by Agrawal and Yamada [AY20] and Agrawal, Wichs, and Yamada [AWY20]

[3]More precisely, the work considers adversaries that receive a verification key after they output a valid certificate. Kitagawa et al. show that their scheme can be converted to satisfy IND-KLA by Agrawal et al. [AKN$^+$23].

[4]Roughly speaking, a pair of circuits (C_0, C_1) is differing input if it is hard to find an input y such that $C_0(y) \neq C_1(y)$.

[5]Francati, Fior, Malavolta, and Venturi [FFMV23] and Agrawal, Tomida, and Yadav [ATY23] proposed MI-ABE for restricted functionalities.

[6]IO implies multi-input functional encryption [GGG$^+$14], which implies MI-ABE.

[7]Precisely speaking, their scheme is predicate encryption, which satisfies privacy for attributes.

rely on the LWE assumption and *pairings*, which are not post-quantum secure. The construction by Wee [Wee22] relies on the *evasive LWE assumption*, which is a non-falsifiable assumption and has known counterexamples [BÜW24]. The construction by Brakerski and Vaikuntanathan [BV22] relies on *heuristics* and lacks a reduction-based proof.

While broadcast encryption is particularly well-suited for streaming services, PKE-CR-SKL can also be applied in this domain. Each approach has its own advantages and limitations. One advantage of broadcast encryption is that it allows the sender to specify recipients at the encryption phase, whereas PKE-CR-SKL does not. However, PKE-CR-SKL offers three notable advantages over optimal broadcast encryption.

1. **Efficient revocation:** Since decryption keys in broadcast encryption are classical, the system requires maintaining a revocation list, which senders use to revoke users, whereas PKE-CR-SKL eliminates the need for such lists.

2. **Seamless user expansion:** In broadcast encryption, the user set is fixed during the setup phase, meaning that adding new users requires updating the encryption key. In contrast, PKE-CR-SKL allows new users to be added without requiring any key updates.

3. **Weaker cryptographic assumptions:** Our PKE-CR-SKL scheme is based on the standard LWE assumption, whereas post-quantum secure optimal broadcast encryption relies on the evasive LWE assumption, which has counterexamples [BÜW24].

In terms of asymptotic efficiency, both optimal broadcast encryption and PKE-CR-SKL achieve the same public key and ciphertext sizes.

Certified deletion. Broadbent and Islam [BI20] introduced encryption with certified deletion, where we can generate classical certificates to guarantee that *ciphertexts* were deleted. Subsequent works improved Broadbent and Islam's work and achieved advanced encryption with certified deletion [HMNY21, Por23, BK23, HKM+24, BGK+24] and publicly verifiable deletion [HMNY21, BGK+24, KNY23, BKM+23]. Compute-and-compare obfuscation with certified deletion introduced by Hiroka et al. [HKM+24] is essentially the same as secure software leasing in the strong security model.

Single decryptor encryption. Georgiou and Zhandry [GZ20] introduced the notion of SDE. They constructed a public-key SDE scheme from one-shot signatures [AGKZ20] and extractable witness encryption with quantum auxiliary information [GGSW13, GKP+13]. Coladangelo, Liu, Liu, and Zhandry [CLLZ21] constructed a public-key SDE scheme from IO [BGI+12] and extractable witness encryption or from subexponentially secure IO, subexponentially secure OWF, and LWE by combining the results by Culf and Vidick [CV22]. Kitagawa and Nishimaki [KN22] introduced the notion of single-decryptor functional encryption (SDFE), where each functional decryption key is copy-protected and constructed single decryptor PKFE for P/poly from the subexponential hardness of IO and LWE. These works consider the setting where the adversary receives only one copy-protected decryption key. Liu, Liu, Qian, and Zhandry [LLQZ22] study SDE in the collusion-resistant setting, where the adversary receives multiple copy-protected decryption keys. They constructed a public-key SDE scheme with bounded collusion-resistant copy-protected keys from subexponentially secure IO and subexponentially secure LWE.

Multi-copy revocable encryption. Ananth, Mutreja, and Poremba [AMP24] introduced multi-copy revocable encryption. This notion considers the setting where we can revoke *ciphertexts* (not decryption keys), and the adversary receives multiple copies of the target *ciphertext* (they are pure states). Hence, this notion is different from secure key leasing (or key-revocable cryptography).

2 Technical Overview

We now provide a technical overview of this work. Our primary focus here is on constructing PKE-CR-SKL and ABE-CR-SKL based on the LWE assumption. For an overview of our variants with classical certificates based on MI-ABE, please refer to Section 8.3.

2.1 Defining PKE with Collusion-Resistant SKL

We will begin by describing the definition of PKE with Secure Key-Leasing (PKE-SKL) [AKN+23, APV23], and then get into our collusion-resistant generalization of it. PKE-SKL is a cryptographic primitive consisting of five algorithms: $\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}$ and $\mathcal{Vrfy}$. The algorithm Setup samples a public encryption-key ek and a "master" secret-key msk. The encryption algorithm Enc takes a message m and ek as inputs and produces a corresponding ciphertext ct. Both these algorithms are classical and similar to their counterparts in standard PKE. On the other hand, the key-generation algorithm $\mathcal{KG}$ is quantum, and produces a pair of keys $(\mathcal{dk}, \mathsf{vk})$ given msk as input. Here, $\mathcal{dk}$ is a quantum decryption-key using which $\mathcal{Dec}$ can decrypt arbitrarily many ciphertexts encrypted under ek. The other key vk is a classical verification-key, the purpose of which will be clear in a moment.

The setting to consider is one where an adversary is given a decryption-key $\mathcal{dk}$, and is later asked to return it back. Intuitively, we wish to guarantee that if $\mathcal{dk}$ is returned, the adversary can no longer decrypt ciphertexts encrypted under ek. Since a malicious adversary may even send a malformed key $\widetilde{\mathcal{dk}}$, we need a way to tell whether the decryption-key has been correctly returned. This is the purpose of the algorithm $\mathcal{Vrfy}$, which performs such a check using the corresponding (private) verification-key vk. It is required that if the state $\mathcal{dk}$ is sent back undisturbed, then $\mathcal{Vrfy}$ must accept. On the other hand, if $\mathcal{Vrfy}$ accepts (even for a possibly malformed state), then the adversary must lose the ability to decrypt. This loss in the ability to decrypt is captured formally by a cryptographic game, where an adversary is asked to distinguish between ciphertexts of different messages, after passing the verification check.

So far, we have described the notion of PKE-SKL. In this work, we study the notion of PKE with Collusion-Resistant SKL (PKE-CR-SKL). This primitive has the same syntax as PKE-SKL, but the aforementioned security requirement is now stronger. Specifically, an adversary that obtains polynomially-many decryption keys and (verifiably) returns them all, should no longer be able to decrypt. This is defined formally in our notion of IND-KLA (Indistinguishability under Key-Leasing Attacks) security (Definition 4.3). As standard in cryptography, it is characterized by a game between a challenger $\mathcal{Ch}$ and a QPT adversary $\mathcal{A}$. An informal description of the game follows:

IND-KLA Game in the Collusion-Resistant Setting

1. $\mathcal{Ch}$ samples $(\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends ek to $\mathcal{A}$.
2. Then, $\mathcal{A}$ requests q decryption keys corresponding to ek, where q is some arbitrary polynomial in λ.[8]
3. $\mathcal{Ch}$ generates $(\mathcal{dk}_i, \mathsf{vk}_i)_{i\in[q]}$ using q independent invocations of $\mathcal{KG}(\mathsf{msk})$. It sends $\{\mathcal{dk}_i\}_{i\in[q]}$ to $\mathcal{A}$.
4. Corresponding to each index $i \in [q]$, $\mathcal{A}$ is allowed oracle access to the algorithm $\mathcal{Vrfy}(\mathsf{vk}_i, \cdot)$ in the following sense: For a quantum state $\widetilde{\mathcal{dk}}$ queried by $\mathcal{A}$, the oracle evaluates $\mathcal{Vrfy}(\mathsf{vk}_i, \widetilde{\mathcal{dk}})$ and measures the verification result. It then returns the obtained classical outcome (which indicates accept/reject). We emphasize that $\mathcal{A}$ is allowed to interleave queries corresponding to indices $i \in [q]$, and can also make its queries adaptively.
5. If at-least one query of $\mathcal{A}$ to $\mathcal{Vrfy}(\mathsf{vk}_i, \cdot)$ produces an accept output for every $i \in [q]$, the game proceeds to the challenge phase. Otherwise, the game aborts.
6. In the challenge phase, $\mathcal{A}$ specifies a pair of messages $(\mathsf{m}_0, \mathsf{m}_1)$. $\mathcal{Ch}$ sends $\mathsf{m}_{\mathsf{coin}}$ to $\mathcal{A}$ for a random bit coin.
7. $\mathcal{A}$ outputs coin'.

$\mathcal{A}$ wins the game whenever $\mathsf{coin} = \mathsf{coin}'$. The security requirement is that for every QPT adversary, the winning probability conditioned on the game not aborting is negligibly close to $1/2$. Observe that the adversary is allowed to make polynomially-many attempts in order to verifiably return a decryption-key $\mathcal{dk}_i$, and unsuccessful attempts are not penalized. We also emphasize that in the definition, q is an unbounded polynomial, i.e., the construction is not allowed to depend on q in any way.

2.2 Insecurity of Direct Extensions of Prior Work

Next, we will provide some intuition regarding the insecurity of direct extensions of prior works to this stronger setting.

[8] Without loss of generality, we can assume that $\mathcal{A}$ asks for sufficiently many keys at the beginning itself. Hence, even adversaries that can request additional keys after accessing the verification oracle are covered by this definition.

Let us consider the PKE-SKL scheme due to Agrawal et al. [AKN$^+$23] for demonstration. The decryption-keys in their scheme are of the following form[9]:

$$\frac{1}{\sqrt{2}}\big(\,|0\rangle\,|\mathsf{sk}_0\rangle + |1\rangle\,|\mathsf{sk}_1\rangle\,\big)$$

Here, $\mathsf{sk}_0, \mathsf{sk}_1$ are secret-keys corresponding to public-keys $\mathsf{pk}_0, \mathsf{pk}_1$ respectively of a standard PKE scheme. Specifically, the pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ are generated using independent invocations of the setup algorithm of PKE. The PKE-SKL public-key consists of the pair $(\mathsf{pk}_0, \mathsf{pk}_1)$. The encryption algorithm outputs ciphertexts of the form $(\mathsf{ct}_0, \mathsf{ct}_1)$, where for each $i \in \{0,1\}$, ct_i encrypts the plaintext under pk_i. The verification procedure requires the adversary to return the decryption-key, and checks if it is the same as the above state. The intuition is that if the adversary retains the ability to decrypt, then it cannot pass the verification check with probability close to 1. For instance, measuring the state provides the adversary with sk_0 or sk_1 which is sufficient for decryption, but it clearly destroys the above quantum state.

Consider now the scenario where the public-key $(\mathsf{pk}_0, \mathsf{pk}_1)$ is fixed, and $n = \mathrm{poly}(\lambda)$ copies of the above decryption-key are given to an adversary. In this case, it is easy to see that the adversary can simply measure the states to obtain both sk_0 and sk_1. Even though this process is destructive, since sk_0 and sk_1 completely describe the state, the adversary can just recreate many copies of it and pass the deletion checks. Consequently, one might be tempted to encode other secret information in the Hadamard basis by introducing random phases. However, we observe that such approaches also fail due to simple gentle-measurement attacks. For example, consider the combined state of the n decryption-keys in such a case:

$$\frac{1}{2^{n/2}}\Big(\,|0\ldots0\rangle\,|\mathsf{sk}_0\ldots\mathsf{sk}_0\rangle - |00\ldots1\rangle\,|\mathsf{sk}_0\mathsf{sk}_0\ldots\mathsf{sk}_1\rangle + \cdots - |1\ldots1\rangle\,|\mathsf{sk}_1\ldots\mathsf{sk}_1\rangle\,\Big)$$

Clearly, only the last term of the superposition does not contain sk_0, and the term only has negligible amplitude. Hence, one can compute sk_0 on another register and measure it, without disturbing the state more than a negligible amount. As a result, the verification checks can be passed while retaining the ability to decrypt. We note that all existing constructions of encryption with SKL can be broken with similar collusion attacks.

The reason our scheme does not run into such an attack is because we rely on the notion of Attribute-Based Encryption (ABE), which enables exponentially-many secret-keys for every public-key. Consequently, different decryption-keys can be generated as superpositions of different ABE secret-keys. However, it is not clear how this intuition alone can be used to establish security in a provable manner, and we require additional ideas. We now describe these ideas at a high-level.

2.3 Idea Behind the PKE-CR-SKL Scheme

In order to explain the basic idea behind our PKE-CR-SKL scheme, we will first introduce a key-building block, a primitive called SKE-CR-SKL. This is basically a secret-key variant of PKE-CR-SKL, i.e., the setup algorithm only samples a master secret-key $\mathsf{ske.msk}$, and the encryption algorithm encrypts plaintexts under $\mathsf{ske.msk}$. The security requirement is similar. An adversary that receives polynomially many decryption-keys and returns them all, should not be able to distinguish between ciphertexts of different messages. We refer to this security notion as one-time IND-KLA (OT-IND-KLA) security (Definition 4.7). The "one-time" prefix refers to the fact that unlike in PKE-CR-SKL, the adversary does not have the ability to perform chosen plaintext attacks. In other words, the adversary does not see any ciphertexts before it is required to return its decryption-keys. Although this is a weak security guarantee, it suffices for our PKE-CR-SKL scheme. The description of the PKE-CR-SKL scheme now follows.

The key-generation procedure involves first generating an SKE-CR-SKL decryption-key, which is represented in the computational basis as $\mathsf{ske}.\mathcal{dk} = \sum_u \alpha_u\,|u\rangle$. Actually, our SKE-CR-SKL scheme needs to satisfy another crucial property, which we call the classical decryption property. This property requires the existence of a classical deterministic algorithm CDec with the following guarantee. For any SKE-CR-SKL ciphertext $\mathsf{ske.ct}$, $\mathsf{CDec}(u, \mathsf{ske.ct})$ correctly decrypts $\mathsf{ske.ct}$ for every string u in the superposition of $\mathsf{ske}.\mathcal{dk}$. Our construction exploits this fact with the help of an ABE scheme as follows.

[9] In actuality, they use a parallel repetition to amplify security.

The actual decryption key is generated as $dk := \sum_u \alpha_u |u\rangle \otimes |\mathsf{abe.sk}_u\rangle$ where $\mathsf{abe.sk}_u$ is an ABE secret-key corresponding to the key-attribute u. The idea is to now have the encryption algorithm encrypt the plaintext using the ABE scheme, under a carefully chosen ciphertext-policy. Specifically, we wish to embed an SKE-CR-SKL ciphertext $\mathsf{ske.ct}^*$ as part of the policy-circuit, such that the outcome of $\mathsf{CDec}(u, \mathsf{ske.ct}^*)$ determines whether the key $\mathsf{abe.sk}_u$ can decrypt the ABE ciphertext or not. This allows us to consider two ciphertexts $\mathsf{ske.ct}^*_0, \mathsf{ske.ct}^*_1$ of different plaintexts, such that when $\mathsf{ske.ct}^*_0$ is embedded in the policy, then every ABE key $\mathsf{abe.sk}_u$ satisfies the ABE relation. On the other hand, no ABE key satisfies the relation when $\mathsf{ske.ct}^*_1$ is embedded. Observe that in the former case, dk allows for decryption without disturbing the state (by gentle measurement) while in the latter case, the security of ABE ensures that no adversary can distinguish ciphertexts of different messages[10]. Hence, if we were to undetectably switch the policy-circuit from one corresponding to $\mathsf{ske.ct}^*_0$ to one with $\mathsf{ske.ct}^*_1$, we are done.

While we cannot argue this directly, our main idea is that this switch is undetectable using OT-IND-KLA security, given that all the SKE-CR-SKL keys $\mathsf{ske.}dk$ are returned. This can be enforced because all the leased decryption keys dk are required to be returned. Consequently, the verification procedure uncomputes the ABE secret-key register of the returned keys, followed by verifying all the obtained SKE-CR-SKL keys $\mathsf{ske.}dk'$. However, there is one problem with the template as we have described it so far. This is that both $\mathsf{ske.ct}^*_0, \mathsf{ske.ct}^*_1$ are SKE-CR-SKL ciphertexts, which inherently depend on the master secret-key. In order to remove this dependence and achieve public-key encryption, the actual encryption algorithm uses a dummy ciphertext-policy $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda)$ where $\mathsf{CC.Sim}$ is the simulator of a compute-and-compare obfuscation scheme, a notion we will explain shortly. We will then rely on the security of this obfuscation scheme to switch $\widetilde{C}$ in the security proof, to an appropriate obfuscated circuit with $\mathsf{ske.ct}^*_0$ embedded in it.

In more detail, the ABE scheme allows a key with attribute u to decrypt if and only if the ciphertext policy-circuit $\widetilde{C}$ satisfies $\widetilde{C}(u) = \perp$. Observe that in the construction, $\widetilde{C}$ is generated as $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda)$, which outputs $\perp$ on every input u. Consider now a circuit $\widetilde{C}^*$ that is an obfuscation of the circuit $\mathbf{CC}[D[\mathsf{ske.ct}^*_0], \mathsf{lock}, 0]$ which is described as follows:

Description of $\mathbf{CC}[D[\mathsf{ske.ct}^*_0], \mathsf{lock}, 0]$:

- $\mathsf{ske.ct}^*_0$ is an SKE-CR-SKL encryption of the (dummy) message 0^λ.
- $D[\mathsf{ske.ct}^*_0]$ is a circuit with $\mathsf{ske.ct}^*_0$ hardwired. It is defined as $D[\mathsf{ske.ct}^*_0](x) = \mathsf{CDec}(x, \mathsf{ske.ct}^*_0)$.
- lock is a value chosen uniformly at random, independently of all other values.
- On input x, the circuit outputs 0 if $D[\mathsf{ske.ct}^*_0](x) = \mathsf{lock}$. Otherwise, it outputs $\perp$.

The above circuit belongs to a sub-class of circuits known as compute-and-compare circuits. Recall that our goal was to avoid the use of IO. We can get away with using IO for this sub-class of circuits, as these so-called compute-and-compare obfuscation schemes are known from LWE [GKW17, WZ17].

The security of the obfuscation can now be used to argue that the switch from $\widetilde{C}$ to $\widetilde{C}^*$ is indistinguishable. Note that to invoke this security guarantee, lock must be a uniform value that is independent of all other values. Next, we can rely on the OT-IND-KLA security of SKE-CR-SKL to switch the ciphertext $\mathsf{ske.ct}^*_0$ embedded in the circuit D to some other ciphertext $\mathsf{ske.ct}^*_1$. The switch would be indistinguishable given that the SKE-CR-SKL decryption-keys are revoked, which we can enforce as mentioned previously. Crucially, we will generate $\mathsf{ske.ct}^*_1$ as an encryption of the value lock corresponding to the above compute-and-compare circuit.[11] This ensures that for every attribute u in the superposition of an SKE-CR-SKL decryption-key $\mathsf{ske.}dk = \sum_u \alpha_u |u\rangle$, the algorithm $\mathsf{CDec}(u, \mathsf{ske.ct}^*_1)$ outputs the value lock. As a consequence, the circuit $\widetilde{C}^*$ will output 0 instead of $\perp$, meaning the key $\mathsf{abe.sk}_u$ does not satisfy the relation in this hybrid, as desired.

It will be clear in the next subsection that our SKE-CR-SKL scheme is implied by OWFs. Since compute-and-compare obfuscation is known from LWE [GKW17, WZ17], and so is Attribute-Based Encryption for polynomial-size circuits [BGG$^+$14][12], we have the following theorem:

Theorem 2.1. *There exists a PKE-CR-SKL scheme satisfying IND-KLA security, assuming the polynomial hardness of the LWE assumption.*

[10]This requires that the ABE scheme is secure even given superposition access to the key-generation oracle.

[11]Although the compute-and-compare circuit depends on lock in this hybrid, the switch is still justified by OT-IND-KLA security.

[12]We show that their ABE scheme is secure even with superposition access to the key-generation oracle.

We also observe that using similar ideas as in PKE-CR-SKL, one can obtain an analogous notion of selectively-secure ABE with collusion-resistant secure-key leasing (ABE-CR-SKL) for arbitrary polynomial-time computable circuits. Intuitively, this primitive allows the adversary to declare a target ciphertext-attribute and then make arbitrarily many key-queries adaptively, even ones which satisfy the ABE relation. Then, as long as the adversary verifiably returns all the keys that satisfy the relation, it must lose the ability to decrypt. This is captured formally in the notion of selective IND-KLA security (Definition 7.12). To realize this primitive, we introduce a notion called secret-key functional-encryption with collusion-resistant secure key-leasing (SKFE-CR-SKL), which is a functional encryption analogue of SKE-CR-SKL. From this primitive, we require a notion called selective single-ciphertext security (Definition 7.2), that is similar to the OT-IND-KLA security of SKE-CR-SKL. Like SKE-CR-SKL, we observe that SKFE-CR-SKL is also implied by OWFs. The other elements of the ABE-CR-SKL construction are the same as in PKE-CR-SKL, namely compute-and-compare obfuscation and an ABE scheme. Consequently, we have the following theorem:

Theorem 2.2. *There exists an ABE-CR-SKL scheme satisfying selective IND-KLA security, assuming the polynomial hardness of the LWE assumption.*

2.4 Constructing SKE-CR-SKL

Next, we will describe how we realize the aforementioned building-block of SKE-CR-SKL satisfying the classical decryption property (Definition 4.5). Our construction will make use of a BB84-based secret-key encryption with certified-deletion (SKE-CD) scheme, a brief description of which is as follows. This is an encryption scheme where the ciphertexts are quantum BB84 states [Wie83, BB20]. Given such a ciphertext, an adversary is later asked to provide a certificate of deletion. If a certificate is provided and verified to be correct, it is guaranteed that the adversary learns nothing about the plaintext even if the secret-key is later revealed. Crucially, we require that the SKE-CD scheme also satisfies a classical decryption property (Definition 3.12). Intuitively, this requires that if the ciphertext is of the form $\mathsf{skecd}.\mathpzc{ct} = \sum_u \alpha_u |u\rangle$, then every string u in the superposition can be used to decrypt correctly. Specifically, there exists a classical deterministic algorithm $\mathsf{SKECD.CDec}$ such that $\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u)$ correctly decrypts $\mathsf{skecd}.\mathpzc{ct}$, where $\mathsf{skecd.sk}$ is the secret-key.

Let us now recall the functionality offered by SKE-CR-SKL. The encryption algorithm Enc takes as input a master secret-key $\mathsf{ske.msk}$ and a plaintext m and outputs a classical ciphertext $\mathsf{ske.ct}$. The key-generation algorithm $\mathcal{KG}$ takes as input $\mathsf{ske.msk}$ and produces a quantum decryption key $\mathsf{ske}.\mathpzc{dk}$ along with a corresponding verification-key $\mathsf{ske.vk}$. Decryption of $\mathsf{ske.ct}$ can be performed by $\mathcal{D}ec$ using $\mathsf{ske}.\mathpzc{dk}$ without disturbing the state by more than a negligible amount. Furthermore, an adversary that receives q (unbounded polynomially many) decryption-keys can be asked to return all of them, before which it does not get to see any ciphertext encrypted under $\mathsf{ske.msk}$. Each returned key can be verified using the verification algorithm and the corresponding verification key. If all q keys are verifiably returned, then it is guaranteed that the adversary cannot distinguish a pair of ciphertexts (of different messages) encrypted under $\mathsf{ske.msk}$. This requirement, termed as OT-IND-KLA security, is captured formally in Definition 4.7. The intuition behind the construction is now described as follows:

Let us begin with the simple encryption algorithm. $\mathsf{Enc}(\mathsf{ske.msk}, \mathsf{m})$ produces a classical output $\mathsf{ske.ct} = (\mathsf{skecd.sk}, z := \mathsf{m} \oplus r)$, where $\mathsf{skecd.sk}$ and r are values specified by $\mathsf{ske.msk}$. The value $\mathsf{skecd.sk}$ is a secret key of an SKE-CD scheme SKECD, while r is a string chosen uniformly at random. Clearly, one can retrieve m from $\mathsf{ske.ct}$ given r. Consequently, each decryption key $\mathsf{ske}.\mathpzc{dk}$ is essentially an SKECD encryption of r. The idea is that the secret-key $\mathsf{skecd.sk}$ can be obtained from $\mathsf{ske.ct}$, which can then be used to retrieve r from $\mathsf{ske}.\mathpzc{dk}$. As a consequence, collusion-resistance (OT-IND-KLA security) can be argued based on the security of SKECD by utilizing the following observations:

- Each decryption-key contains an SKECD encryption of r using independent randomness.
- The adversary must return all the decryption keys (containing SKECD ciphertexts) before it receives the challenge ciphertext (containing the SKECD secret-key).

Furthermore, since $\mathsf{ske}.\mathpzc{dk}$ is essentially an SKECD ciphertext, the classical decryption property of SKE-CR-SKL follows easily from the analogous classical decryption property of BB84-based SKE-CD (Definition 3.12).

2.5 Handling Verification Queries

In our previous discussion about the idea behind the PKE-CR-SKL scheme, we left out some details regarding the following. We did not discuss how the key-generation oracle of the ABE scheme can be used to simulate the adversary's view, in the hybrid where $\mathsf{ske.ct}_1^*$ is embedded in the circuit $\widetilde{C}$. Firstly, we note that the ABE scheme can handle superposition key-queries, which we establish by a straightforward argument about the LWE-based ABE scheme of Boneh et al. [BGG+14]. Recall now that in this hybrid, for every leased decryption-key $dk = \sum_u \alpha_u |u\rangle \otimes |\mathsf{abe.sk}_u\rangle$, each $\mathsf{abe.sk}_u$ can be obtained (in superposition) by querying $\mathsf{ske}.dk = \sum_u \alpha_u |u\rangle$ to the oracle of ABE. However, we observe that responses to verification queries made by the adversary are not so straightforward to simulate. This is because for each verification query, the ABE secret-key register needs to be uncomputed, for which we will again rely on the ABE key-generation oracle. Specifically, the problem is that for a malformed key $\widetilde{dk}$, there may exist some $\widetilde{u}$ in the superposition which actually satisfies the ABE relation. Recall that by definition of this relation, it follows that $\mathsf{CDec}(\widetilde{u}, \mathsf{ske.ct}_1^*)$ incorrectly decrypts the ciphertext $\mathsf{ske.ct}_1^*$. To fix this issue, we upgrade our SKE-CR-SKL scheme to satisfy another property called key-testability. This involves the existence of a classical algorithm $\mathsf{KeyTest}$ which accepts or rejects. The property requires that when $\mathsf{KeyTest}$ is applied in superposition followed by post-selecting on it accepting, every $\widetilde{u}$ in the superposition decrypts correctly. As a result, we are able to apply this key-testing procedure to the register of the SKE-CR-SKL key, followed by simulating the adversary's view using the ABE oracle. Note that we will now consider $\mathcal{KG}(\mathsf{ske.msk})$ to output an additional testing-key $\mathsf{ske.tk}$ along with $\mathsf{ske}.dk$ and $\mathsf{ske.vk}$. The key-testability requirements are specified in more detail as follows:

- **Security:** There exists an algorithm $\mathsf{KeyTest}$ such that no QPT adversary can produce a classical value dk and a message m such that:
  - $\mathsf{CDec}(\mathsf{dk}, \mathsf{ske.ct}) \neq \mathsf{m}$, where $\mathsf{ske.ct} \leftarrow \mathsf{Enc}(\mathsf{ske.msk}, \mathsf{m})$.
  - $\mathsf{KeyTest}(\mathsf{ske.tk}, \mathsf{dk}) = 1$.
- **Correctness:** If $\mathsf{KeyTest}$ is applied in superposition to $\mathsf{ske}.dk$ and the output is measured to obtain outcome 1, $\mathsf{ske}.dk$ should be almost undisturbed.

2.6 Upgrading SKE-CR-SKL with Key-Testability

We will now explain how the SKE-CR-SKL construction is modified to satisfy the aforementioned key-testability property. First, we mention a classical decryption property of SKECD (Definition 3.12) that we crucially rely on. A ciphertext $\mathsf{skecd}.ct$ (corresponding to some message m) of SKECD is essentially a BB84 state $|x\rangle_\theta$. The property guarantees the existence of an algorithm $\mathsf{SKECD.CDec}$ such that for any string u that matches x at all computational basis positions specified by θ, $\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u) = \mathsf{m}$ with overwhelming probability. Recall now that in our SKE-CR-SKL construction, ciphertexts are of the form $\mathsf{ske.ct} = (\mathsf{skecd.sk}, z = r \oplus \mathsf{m})$ and decryption-keys are essentially SKECD encryptions of the value r. Consequently, the algorithm CDec of SKE-CR-SKL works as follows:

$\underline{\mathsf{CDec}(u, \mathsf{ske.ct})}$:

- Parse $\mathsf{ske.ct} = (\mathsf{skecd.sk}, z)$.
- Output $z \oplus \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u)$.

As a result, it is sufficient for us to ensure that no QPT adversary can output a value dk such that $\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, \mathsf{dk})$ produces a value different from r. By the aforementioned classical-decryption property of SKECD, it suffices to bind the adversary to the computational basis values of a ciphertext $\mathsf{skecd}.ct = |x\rangle_\theta$. For this, we employ a technique reminiscent of the Lamport-signature scheme [Lam79]. Thereby, an additional "signature" register that is entangled with the SKECD ciphertext $\mathsf{skecd}.ct = |x\rangle_\theta$ is utilized. We note that similar techniques for signing BB84 states were employed in previous works on certified deletion [HKM+24, KNY23]. Specifically, let $\mathsf{SKECD.CT_i}$ denote the register holding the i-th qubit of $\mathsf{skecd}.ct$ and $s_{i,0}, s_{i,1}$ be randomly chosen pre-images from the domain of an OWF f. Then, the following map is performed on registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ where the latter is initialized to $|0\ldots0\rangle$:

$$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \to |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_i}}$$

Let ρ_i be the resulting state. The SKE-CR-SKL decryption-key $\mathsf{ske}.\mathcal{dk}$ is set to be the state $\rho_1 \otimes \ldots \otimes \rho_{\ell_{\mathsf{ct}}}$ where ℓ_{ct} is the length of $\mathsf{skecd}.ct$. Thereby, the testing-key $\mathsf{ske.tk}$ will consist of the values $f(s_{i,0}), f(s_{i,1})$ for each $i \in [\ell_{\mathsf{ct}}]$. Observe now that for a returned (possibly altered) decryption-key $\widetilde{\mathcal{dk}}$ (or $\mathsf{ske}.\widetilde{\mathcal{dk}}$), it is possible to check for each qubit whether the superposition term u_i is associated with the correct pre-image s_{i,u_i}. This can be done by forward evaluating the pre-image register and comparing it with the value $f(s_{i,u_i})$ that is specified in $\mathsf{ske.tk}$. This is essentially the $\mathsf{KeyTest}$ algorithm. It is easy to see that this procedure does not disturb the state when applied to the unaltered decryption key $\mathcal{dk}$ (or $\mathsf{ske}.\mathcal{dk}$). Moreover, observe that the adversary does not receive the pre-image $s_{i,1-x[i]}$ for any computational basis position i. Consequently, we show that the adversary cannot produce a value dk whose computational basis values are inconsistent with those of x, unless it can invert outputs of f. From the previous discussion, it follows that if the computational basis values of dk are consistent with x, then dk cannot result in the incorrect decryption of $\mathsf{ske.ct}$.

3 Preliminaries

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

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

3.1 One Way to Hiding Lemmas

Lemma 3.1 (O2H Lemma [AHU19]). *Let $G, H : X \to Y$ be any functions, z be a random value, and $S \subseteq X$ be a random set such that $G(x) = H(x)$ holds for every $x \notin S$. The tuple (G, H, S, z) may have arbitrary joint distribution. Furthermore, let $\mathcal{A}$ be a quantum oracle algorithm that makes at most q quantum queries. Let $\mathcal{B}$ be an algorithm such that $\mathcal{B}^H$ on input z chooses $i \leftarrow [q]$, runs $\mathcal{A}^H(z)$, measures $\mathcal{A}$'s i-th query, and outputs the measurement outcome. Then, we have:*

$$\left|\Pr\left[\mathcal{A}^H(z) = 1\right] - \Pr\left[\mathcal{A}^G(z) = 1\right]\right| \le 2q \cdot \sqrt{\Pr[\mathcal{B}^H(z) \in S]} \ .$$

We require a generalization of this lemma, where $\mathcal{A}$ receives an additional quantum oracle $\mathcal{Q}$ in both worlds. Consequently, we consider $\mathcal{B}$ to be given oracle access to $\mathcal{Q}$, which it uses to simulate the oracle calls of $\mathcal{A}$ to $\mathcal{Q}$. Notice that if the outputs of $\mathcal{Q}$ were classical, we could have simply defined augmented oracles G' (likewise H') based on G (likewise H) and $\mathcal{Q}$. However, the oracles $\mathcal{Q}$ we consider will have classical inputs and quantum state outputs. Consequently, the lemma we require is stated as follows:

Lemma 3.2 (O2H Lemma with Auxiliary Quantum Oracle). *Let $G, H : X \to Y$ be any functions, z be a random value, and $S \subseteq X$ be a random set such that $G(x) = H(x)$ holds for every $x \notin S$. The tuple (G, H, S, z) may have arbitrary joint distribution. Furthermore, let $\mathcal{Q}$ be a quantum oracle that is arbitrarily correlated with the tuple (G, H, S, z), takes classical input and produces a (possibly mixed) quantum state as output. Let $\mathcal{A}$ be a quantum oracle algorithm that makes at most q quantum queries to the oracles H or G, and arbitrarily many queries to the oracle $\mathcal{Q}$. Let $\mathcal{B}$ be an algorithm such that $\mathcal{B}^{\mathcal{Q},H}$ on input z, chooses $i \leftarrow [q]$, runs $\mathcal{A}^{\mathcal{Q},H}(z)$, measures $\mathcal{A}$'s i-th query to H, and outputs the measurement outcome. Then, we have:*

$$\left|\Pr\left[\mathcal{A}^{\mathcal{Q},H}(z)=1\right]-\Pr\left[\mathcal{A}^{\mathcal{Q},G}(z)=1\right]\right| \le 2q\cdot\sqrt{\Pr[\mathcal{B}^{\mathcal{Q},H}(z)\in S]}\ .$$

Proof of Lemma 3.2. Let us consider an adversary $\widetilde{\mathcal{A}}$ that receives as input the description $\langle\mathcal{Q}\rangle$ of $\mathcal{Q}$[13], along with the input z used by $\mathcal{A}$. Given oracle access to H (likewise G) and $(z,\langle\mathcal{Q}\rangle)$ as input, $\widetilde{\mathcal{A}}$ simply runs $\mathcal{A}^{H,\mathcal{Q}}(z)$ (likewise $\mathcal{A}^{G,\mathcal{Q}}(z)$) by simulating its queries to $\mathcal{Q}$ using the description $\langle\mathcal{Q}\rangle$. Then, the O2H Lemma (Lemma 3.1) implies the existence of an algorithm $\widetilde{\mathcal{B}}^{H}(z,\langle\mathcal{Q}\rangle)$ that chooses $i\leftarrow[q]$, runs $\widetilde{\mathcal{A}}^{H}(z,\langle\mathcal{Q}\rangle)$, measures its i-th query to H and outputs the measurement outcome. Observe that the algorithms $\widetilde{A}$ and $\widetilde{B}$ do not make use of the description $\langle\mathcal{Q}\rangle$ except for simulating the queries made by $\mathcal{A}$. Consequently, there exists an algorithm $\mathcal{B}^{\mathcal{Q},H}(z)$ equivalent to $\widetilde{\mathcal{B}}^{H}(z,\langle\mathcal{Q}\rangle)$ that directly runs $\mathcal{A}$ (instead of $\widetilde{\mathcal{A}}$) and simulates its oracle queries to $\mathcal{Q}$ using its own access to $\mathcal{Q}$. ☐

Remark 3.3. We assume that $\mathcal{B}$ also outputs the measured index i. However, this output is not taken into account for notation such as $\mathcal{B}^{\mathcal{Q},H}(z)\in S$ for the sake of simplicity.

3.2 Standard Cryptographic Tools

Attribute-Based Encryption.

Definition 3.4 (Attribute-Based Encryption). *An ABE scheme* ABE *is a tuple of four PPT algorithms* $(\mathsf{Setup},\mathsf{KG},\mathsf{Enc},\mathsf{Dec})$. *Below, let* $\mathcal{X}=\{\mathcal{X}_\lambda\}_\lambda$, $\mathcal{Y}=\{\mathcal{Y}_\lambda\}_\lambda$, *and* $R=\{R_\lambda:\mathcal{X}_\lambda\times\mathcal{Y}_\lambda\to\{0,1\}\}_\lambda$ *be the ciphertext attribute space, key attribute space, and the relation associated with* ABE, *respectively. We note that we will abuse the notation and occasionally drop the subscript for these spaces for notational simplicity. We also note that the message space is set to be* $\{0,1\}^\ell$ *below.*

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

$\mathsf{KG}(\mathsf{msk},y,r)\to\mathsf{sk}_y$**:** *The key generation algorithm* KG *takes a master secret key* msk, *a key attribute* $y\in\mathcal{Y}$, *and explicit randomness* r. *It outputs a decryption key* sk_y. *Note that* KG *is deterministic.*[14]

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

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

Correctness: *We require that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_y,\mathsf{ct})=\mathsf{m}\ :\ \begin{array}{l}(\mathsf{pk},\mathsf{msk})\leftarrow\mathsf{Setup}(1^\lambda),\\ r\leftarrow\{0,1\}^{\mathrm{poly}(\lambda)},\\ \mathsf{sk}_y\leftarrow\mathsf{KG}(\mathsf{msk},y,r),\\ \mathsf{ct}\leftarrow\mathsf{Enc}(\mathsf{pk},x,\mathsf{m})\end{array}\right]\ge 1-\mathsf{negl}(\lambda).$$

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

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

We introduce a new security notion for ABE that we call quantum selective-security for ABE where the adversary is allowed to get access to the key generation oracle in super-position.

[13]The O2H Lemma (Lemma 3.1) holds even if z is exponentially large, so the description of $\mathcal{Q}$ need not be concise.

[14]In the standard syntax, KG does not take explicit randomness, and is probabilistic. This change is just for notational convention in our schemes.

Definition 3.5 (Quantum Selective-Security for ABE). *We say that* ABE *is a* selective-secure *ABE scheme for relation* $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, *if it satisfies the following requirement, formalized by the experiment* $\mathsf{Exp}^{\mathsf{q\text{-}sel\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{Ch}$:

1. $\mathcal{A}$ *declares the challenge ciphertext attribute* x^*. $\mathcal{Ch}$ *runs* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *can get access to the following quantum key generation oracle.*

   $O_{\mathsf{qkg}}(\mathsf{Y}, \mathsf{Z})$: *Given two registers* Y *and* Z, *it first applies the map* $|y\rangle_{\mathsf{Y}} |b\rangle_{\mathsf{B}} \to |y\rangle_{\mathsf{Y}} |b \oplus R(x^*, y)\rangle_{\mathsf{B}}$ *and measures the register* B, *where* B *is initialized to* $|0\rangle_{\mathsf{B}}$. *If the result is* 0, *it returns* $\perp$. *Otherwise, it chooses* $r \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)}$, *applies the map* $|y\rangle_{\mathsf{Y}} |z\rangle_{\mathsf{Z}} \to |y\rangle_{\mathsf{Y}} |z \oplus \mathsf{KG}(\mathsf{msk}, y, r)\rangle_{\mathsf{Z}}$ *and returns the registers* Y *and* Z.
3. *At some point,* $\mathcal{A}$ *sends* $(\mathsf{m}_0, \mathsf{m}_1)$ *to* $\mathcal{Ch}$. *Then,* $\mathcal{Ch}$ *generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x^*, \mathsf{m}_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$.
4. *Again,* $\mathcal{A}$ *can get access to the oracle* O_{qkg}.
5. $\mathcal{A}$ *outputs a guess* coin' *for* coin *and the experiment outputs* coin'.

We say that ABE *satisfies quantum selective security if, for all QPT* $\mathcal{A}$, *it holds that*

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

Boneh and Zhandry [BZ13b] introduced a similar quantum security notion for IBE and argued that it is straightforward to prove the quantum security of the IBE scheme by [ABB10], by leveraging the lattice trapdoor based proof technique. It is easy to prove the quantum selective security of the ABE scheme for circuits by Boneh et al. [BGG+14], which relies on the lattice trapdoor based proof technique as well. Formally, we have the following theorem.

Theorem 3.6. *Assuming the hardness of the LWE problem, there exists a quantum selectively secure ABE scheme for all relations computable in polynomial time.*

We elaborate on this in Appendix A.

Compute-and-Compare Obfuscation. We define a class of circuits called compute-and-compare circuits as follows:

Definition 3.7 (Compute-and-Compare Circuits). *A compute-and-compare circuit* $\mathbf{CC}[P, \mathsf{lock}, \mathsf{m}]$ *is of the form*

$$\mathbf{CC}[P, \mathsf{lock}, \mathsf{m}](x) \begin{cases} \mathsf{m} & \textit{if } P(x) = \mathsf{lock} \\ \perp & \textit{otherwise} \end{cases}$$

where P *is a circuit,* lock *is a string called the lock value, and* m *is a message.*

We now introduce the definition of compute-and-compare obfuscation. We assume that a program P has an associated set of parameters pp_P (input size, output size, circuit size) which we do not need to hide.

Definition 3.8 (Compute-and-Compare Obfuscation). *A PPT algorithm* $\mathsf{CC.Obf}$ *is an obfuscator for the family of distributions* $D = \{D_\lambda\}$ *if the following holds:*

Functionality Preserving: *There exists a negligible function* negl *such that for all programs* P, *all lock values* lock, *and all messages* m, *it holds that*

$$\Pr\left[\forall x, \widetilde{P}(x) = \mathbf{CC}[P, \mathsf{lock}, \mathsf{m}](x) : \widetilde{P} \leftarrow \mathsf{CC.Obf}(1^\lambda, P, \mathsf{lock}, \mathsf{m})\right] \geq 1 - \mathsf{negl}(\lambda).$$

Distributional Indistinguishability: *There exists an efficient simulator* Sim *such that for all messages* m*, we have*

$$(\mathsf{CC.Obf}(1^\lambda, P, \mathsf{lock}, \mathsf{m}), \mathit{aux}_\chi) \approx (\mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_P, |\mathsf{m}|), \mathit{aux}_\chi),$$

where $(P, \mathsf{lock}, \mathit{aux}_\chi) \leftarrow D_\lambda$.

Theorem 3.9 ([GKW17, WZ17]). *If the LWE assumption holds, there exists compute-and-compare obfuscation for all families of distributions* $D = \{D_\lambda\}$*, where each* D_λ *outputs uniformly random lock value* lock *independent of* P *and* aux_χ.

We present the definitions for SKE with certified deletion.

Definition 3.10 (SKE-CD (Syntax)). *An SKE-CD scheme is a tuple of algorithms* $(\mathsf{KG}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *with plaintext space* $\mathcal{M}$ *and key space* $\mathcal{K}$.

$\mathsf{KG}(1^\lambda) \rightarrow \mathsf{sk}$**:** *The key generation algorithm takes as input the security parameter* 1^λ *and outputs a secret key* $\mathsf{sk} \in \mathcal{K}$.

$\mathcal{Enc}(\mathsf{sk}, \mathsf{m}) \rightarrow (\mathit{ct}, \mathsf{vk})$**:** *The encryption algorithm takes as input* sk *and a plaintext* $\mathsf{m} \in \mathcal{M}$ *and outputs a ciphertext* ct *and a verification key* vk.

$\mathcal{Dec}(\mathsf{sk}, \mathit{ct}) \rightarrow \mathsf{m}'$**:** *The decryption algorithm takes as input* sk *and* ct *and outputs a plaintext* $\mathsf{m}' \in \mathcal{M}$ *or* $\perp$.

$\mathcal{Del}(\mathit{ct}) \rightarrow \mathsf{cert}$**:** *The deletion algorithm takes as input* ct *and outputs a certificate* cert.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \rightarrow \top/\perp$**:** *The verification algorithm takes* vk *and* cert *as input and outputs* $\top$ *or* $\perp$.

Decryption correctness: *There exists a negligible function* negl *such that for any* $\mathsf{m} \in \mathcal{M}$*,*

$$\Pr\left[\mathcal{Dec}(\mathsf{sk}, \mathit{ct}) = \mathsf{m} \ :\ \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KG}(1^\lambda) \\ (\mathit{ct}, \mathsf{vk}) \leftarrow \mathcal{Enc}(\mathsf{sk}, \mathsf{m}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Verification correctness: *There exists a negligible function* negl *such that for any* $\mathsf{m} \in \mathcal{M}$*,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) = \top \ :\ \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KG}(1^\lambda) \\ (\mathit{ct}, \mathsf{vk}) \leftarrow \mathcal{Enc}(\mathsf{sk}, \mathsf{m}) \\ \mathsf{cert} \leftarrow \mathcal{Del}(\mathit{ct}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

We introduce indistinguishability against Chosen Verification Attacks (CVA).

Definition 3.11 (IND-CVA-CD Security). *We consider the following security experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, \mathsf{coin})$.

1. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{KG}(1^\lambda)$.
2. *Thoughout the experiment,* $\mathcal{A}$ *can get access to the following oracle.*

   $O_{\mathcal{Enc}}(\mathsf{m})$**:** *On input* m*, it generates* $(\mathit{ct}, \mathsf{vk}) \leftarrow \mathcal{Enc}(\mathsf{sk}, \mathsf{m})$ *and returns* $(\mathsf{vk}, \mathit{ct})$.
3. $\mathcal{A}$ *sends* $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M}^2$ *to the challenger.*
4. *The challenger computes* $(\mathit{ct}^*, \mathsf{vk}^*) \leftarrow \mathcal{Enc}(\mathsf{sk}, \mathsf{m}_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$.
5. *Hereafter,* $\mathcal{A}$ *can get access to the following oracle, where* V *is initialized to* $\perp$.

   $O_{\mathsf{Vrfy}}(\mathsf{cert})$**:** *On input* cert*, it returns* sk *and updates* V *to* $\top$ *if* $\mathsf{Vrfy}(\mathsf{vk}^*, \mathsf{cert}) = \top$*. Otherwise, it returns* $\perp$.
6. *When* $\mathcal{A}$ *outputs* $\mathsf{coin}' \in \{0, 1\}$*, the experiment outputs* coin' *if* $V = \top$ *and otherwise outputs* 0.

We say that SKECD *is IND-CVA-CD secure if for any QPT* $\mathcal{A}$*, it holds that*

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

Definition 3.12 (BB84-Based SKE-CD). *We say that an SKE-CD scheme* $\mathsf{SKECD} = (\mathsf{KG}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *is a BB84-based SKE-CD scheme if it satisfies the following conditions.*

- *Let* $(ct, \mathsf{vk}) \leftarrow \mathcal{Enc}(\mathsf{sk}, \mathsf{m})$*.* vk *is of the form* $(x,\theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$*, and* ct *is of the form* $|\psi_1\rangle \otimes \cdots \otimes |\psi_{\ell_{\mathsf{ct}}}\rangle$*, where*

$$|\psi_i\rangle = \begin{cases} |x[i]\rangle & if\ \ \theta[i]=0 \\ |0\rangle + (-1)^{x[i]}|1\rangle & if\ \ \theta[i]=1. \end{cases}$$

  Moreover, there exists $n < \ell_{\mathsf{ct}}$ *such that* $\theta[i]=0$ *for every* $i \in [n+1, \ell_{\mathsf{ct}}]$*. We call* $x[n+1]\|\cdots\|x[\ell_{\mathsf{ct}}]$ *a classical part of* ct*. The parameter* n *is specified by a construction. The classical part has information of* θ*, and we can compute* θ *from it and* sk*.*

- $\mathcal{Del}(ct)$ *measures each qubit of* ct *in the Hadamard basis and outputs the measurement result* $\mathsf{cert} \in \{0,1\}^{\ell_{\mathsf{ct}}}$*.*

- $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$ *outputs* $\top$ *if* $\mathsf{cert}[i] = x[i]$ *holds for every* $i \in [n]$ *such that* $\theta[i]=1$*, and* 0 *otherwise.*

- ***Classical Decryption Property:*** *There exists an additional deterministic polynomial time algorithm* CDec *with the following property. Let* $(ct, \mathsf{vk}) \leftarrow \mathcal{Enc}(\mathsf{sk}, \mathsf{m})$*, where* $\mathsf{vk} = (x,\theta)$*. Let* $u \in \{0,1\}^{\ell_{\mathsf{ct}}}$ *be any string such that* $u[i] = x[i]$ *for all* $i : \theta[i]=0$*. Then, the following holds:*

$$\Pr\left[\mathsf{CDec}(\mathsf{sk}, u) = \mathsf{m}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Theorem 3.13. *There exists a BB84-based SKE-CD scheme satisfying IND-CVA-CD security, assuming the existence of a CPA-secure Secret-Key Encryption scheme.*

Kitagawa and Nishimaki [KN22] claimed the same statement as Theorem 3.13. However, their proof has a gap because known BB84-based SKE-CD schemes do not satisfy the unique certificate property, which they introduced. Hence, we prove Theorem 3.13 in Appendix B.

4 Encryption with Collusion-Resistant SKL

In this section, we define the notions of public-key and secret-key encryption with collusion-resistant secure key-leasing.

4.1 Definitions of PKE-CR-SKL

The syntax of PKE-CR-SKL is defined as follows.

Definition 4.1 (PKE-CR-SKL). *A PKE-CR-SKL scheme* $\mathsf{PKE\text{-}CR\text{-}SKL}$ *is a tuple of five algorithms* $(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$*. Below, let* $\mathcal{M}$ *be the message space of* $\mathsf{PKE\text{-}CR\text{-}SKL}$*.*

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

$\mathcal{KG}(\mathsf{msk}) \rightarrow (dk, \mathsf{vk})$**:** *The key generation algorithm takes the master secret-key* msk *as input, and outputs a decryption key* dk *and a verification key* vk*.*

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

$\mathcal{D}ec(dk, \mathsf{ct}) \to \widetilde{\mathsf{m}}/\bot$**:** *The decryption algorithm takes a decryption key* dk *and a ciphertext* ct*, and outputs a value* $\widetilde{\mathsf{m}}$ *or* $\bot$*.*

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

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

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

Verification correctness: *We have*

$$\Pr\left[\mathcal{V}rfy(\mathsf{vk}, dk) = \top \ : \ \begin{array}{l}(\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (dk, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Remark 4.2. We can assume without loss of generality that a decryption key of a PKE-CR-SKL scheme is reusable, i.e., it can be reused to decrypt (polynomially) many ciphertexts. In particular, we can assume that for honestly generated ct and dk, if we decrypt ct by using dk, the state of the decryption key after the decryption is negligibly close to that before the decryption in terms of trace distance. This is because the output of the decryption is almost deterministic by decryption correctness, and thus such an operation can be done without almost disturbing the input state by the gentle measurement lemma [Win99].

Definition 4.3 (IND-KLA Security). *We say that a PKE-CR-SKL scheme* $\mathsf{PKE\text{-}CR\text{-}SKL}$ *with the message space* $\mathcal{M}$ *is IND-KLA secure, if it satisfies the following requirement, formalized by the experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{PKE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}h$*:*

1. $\mathcal{C}h$ *runs* $(\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* ek *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *requests* q *decryption keys for some polynomial* q*.* $\mathcal{C}h$ *generates* $(dk_i, \mathsf{vk}_i) \leftarrow \mathcal{KG}(\mathsf{msk})$ *for every* $i \in [q]$ *and sends* $dk_1, \ldots, dk_q$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *can get access to the following (stateful) verification oracle* $O_{\mathcal{V}rfy}$ *where* V_i *is initialized to* $\bot$ *for all* $i \in [q]$*:*

   $O_{\mathcal{V}rfy}(i, \widetilde{dk})$**:** *It runs* $d \leftarrow \mathcal{V}rfy(\mathsf{vk}_i, \widetilde{dk})$ *and returns* d*.*
   If $V_i = \bot$ *and* $d = \top$*, it updates* $V_i := \top$*.*

4. $\mathcal{A}$ *sends* $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ *to the challenger. If* $V_i = \bot$ *for some* $i \in [q]$*, the challenger outputs* 0 *as the final output of this experiment. Otherwise, the challenger generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{m}^*_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *outputs a guess* coin' *for* coin*.* $\mathcal{C}h$ *outputs* coin' *as the final output of the experiment.*

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

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

4.2 Definitions of SKE-CR-SKL

The syntax of SKE-CR-SKL is defined as follows.

Definition 4.4 (SKE-CR-SKL). *An SKE-CR-SKL scheme* $\mathsf{SKE\text{-}CR\text{-}SKL}$ *is a tuple of five algorithms* $(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy)$*. Below, let* $\mathcal{M}$ *be the message space of* $\mathsf{SKE\text{-}CR\text{-}SKL}$*.*

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

$\mathcal{KG}(\mathsf{msk}) \to (\mathcal{dk}, \mathsf{vk}, \mathsf{tk})$**:** *The key generation algorithm takes the master secret-key* msk *as input. It outputs a decryption key* $\mathcal{dk}$*, a verification key* vk*, and a testing key* tk.

$\mathsf{Enc}(\mathsf{msk}, \mathsf{m}) \to \mathsf{ct}$**:** *The encryption algorithm takes the master secret-key* msk *and a message* $\mathsf{m} \in \mathcal{M}$*, and outputs a ciphertext* ct.

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

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

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

$$\Pr\left[\mathcal{Dec}(\mathcal{dk}, \mathsf{ct}) = \mathsf{m} \ :\ \begin{array}{l} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathcal{dk}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathsf{m}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Verification correctness: *We have*

$$\Pr\left[\mathcal{Vrfy}(\mathsf{vk}, \mathcal{dk}) = \top \ :\ \begin{array}{l} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathcal{dk}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 4.5 (Classical Decryption Property). *We say that* $\mathsf{SKE\text{-}CR\text{-}SKL} = (\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$ *has the classical decryption property if there exists a deterministic polynomial time algorithm* CDec *such that given* $\mathcal{dk}$ *in the register* DK *and ciphertext* ct*,* $\mathcal{Dec}$ *applies the map* $|u\rangle_{\mathsf{DK}} |v\rangle_{\mathsf{MSG}} \to |u\rangle_{\mathsf{DK}} |v \oplus \mathsf{CDec}(u, \mathsf{ct})\rangle_{\mathsf{MSG}}$ *and outputs the measurement result of the register* MSG *in the computational basis, where* MSG *is initialized to* $|0\cdots 0\rangle_{\mathsf{MSG}}$.

Definition 4.6 (Key Testability). *We say that an SKE-CR-SKL scheme* $\mathsf{SKE\text{-}CR\text{-}SKL}$ *with the classical decryption property satisfies key testability, if there exists a classical deterministic algorithm* $\mathsf{KeyTest}$ *that satisfies the following conditions:*

- ***Syntax:*** $\mathsf{KeyTest}$ *takes as input a testing key* tk *and a classical string* dk *as input. It outputs* 0 *or* 1.
- ***Correctness:*** *Let* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathcal{dk}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk})$*. We denote the register holding* $\mathcal{dk}$ *as* DK*. Let* KT *be a register that is initialized to* $|0\rangle_{\mathsf{KT}}$*. If we apply the map* $|u\rangle_{\mathsf{DK}} |\beta\rangle_{\mathsf{KT}} \to |u\rangle_{\mathsf{DK}} |\beta \oplus \mathsf{KeyTest}(\mathsf{tk}, u)\rangle_{\mathsf{KT}}$ *to the registers* DK *and* KT *and then measure* KT *in the computational basis, we obtain* 1 *with overwhelming probability.*
- ***Security:*** *Consider the following experiment* $\mathsf{Exp}^{\mathsf{key\text{-}test}}_{\mathsf{SKE\text{-}CR\text{-}SKL}, \mathcal{A}}(1^\lambda)$.

  1. *The challenger* $\mathcal{Ch}$ *runs* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *and initializes* $\mathcal{A}$ *with input* msk.
  2. $\mathcal{A}$ *requests* q *decryption keys for some polynomial* q*.* $\mathcal{Ch}$ *generates* $(\mathcal{dk}_i, \mathsf{vk}_i, \mathsf{tk}_i) \leftarrow \mathcal{KG}(\mathsf{msk})$ *for every* $i \in [q]$ *and sends* $(\mathcal{dk}_i, \mathsf{vk}_i, \mathsf{tk}_i)_{i \in [q]}$ *to* $\mathcal{A}$.
  3. $\mathcal{A}$ *sends* $(k, \mathsf{dk}, \mathsf{m})$ *to* $\mathcal{Ch}$*, where* k *is an index,* dk *is a classical string and* m *is a message.* $\mathcal{Ch}$ *generates* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathsf{m})$*.* $\mathcal{Ch}$ *outputs* $\top$ *if* $\mathsf{KeyTest}(\mathsf{tk}_k, \mathsf{dk}) = 1$ *and* $\mathsf{CDec}(\mathsf{dk}, \mathsf{ct}) \neq \mathsf{m}$*. Otherwise,* $\mathcal{Ch}$ *outputs* $\bot$.

  For all QPT $\mathcal{A}$*, the following must hold:*

$$\mathsf{Adv}^{\mathsf{key\text{-}test}}_{\mathsf{SKE\text{-}CR\text{-}SKL}, \mathcal{A}}(1^\lambda) := \Pr\Big[\mathsf{Exp}^{\mathsf{key\text{-}test}}_{\mathsf{SKE\text{-}CR\text{-}SKL}, \mathcal{A}}(1^\lambda) \to \top\Big] \leq \mathsf{negl}(\lambda).$$

Definition 4.7 (OT-IND-KLA Security). *We say that an SKE-CR-SKL scheme with key testability* $\mathsf{SKE\text{-}CR\text{-}SKL}$ *is OT-IND-KLA secure, if it satisfies the following requirement, formalized by the experiment* $\mathsf{Exp}^{\mathsf{ot\text{-}ind\text{-}kla}}_{\mathsf{SKE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda,\mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{Ch}$*:*

1. $\mathcal{Ch}$ *runs* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *and initializes* $\mathcal{A}$ *with the security parameter* 1^λ.
2. $\mathcal{A}$ *requests* q *decryption keys for some polynomial* q*. The challenger generates* $(\mathcal{dk}_i,\mathsf{vk}_i,\mathsf{tk}_i) \leftarrow \mathcal{KG}(\mathsf{msk})$ *for every* $i \in [q]$ *and sends* $(\mathcal{dk}_i,\mathsf{tk}_i)_{i\in[q]}$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *can get access to the following (stateful) verification oracle* $O_{\mathcal{Vrfy}}$ *where* V_i *is initialized to be* $\perp$*:*

   $O_{\mathcal{Vrfy}}(i,\widetilde{\mathcal{dk}})$**:** *It runs* $d \leftarrow \mathcal{Vrfy}(\mathsf{vk}_i,\widetilde{\mathcal{dk}})$ *and returns* d.
   If $V_i = \perp$ *and* $d = \top$*, it updates* $V_i := \top$.

4. $\mathcal{A}$ *sends* $(\mathsf{m}_0^*,\mathsf{m}_1^*) \in \mathcal{M}^2$ *to the challenger. If* $V_i = \perp$ *for some* $i \in [q]$*,* $\mathcal{Ch}$ *outputs* 0 *as the final output of this experiment. Otherwise,* $\mathcal{Ch}$ *generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{msk},\mathsf{m}^*_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$.
5. $\mathcal{A}$ *outputs a guess* coin' *for* coin*.* $\mathcal{Ch}$ *outputs* coin' *as the final output of the experiment.*

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

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

5 SKE-CR-SKL with Key Testability

In this section, we show how to achieve SKE-CR-SKL introduced in Section 4.2.

5.1 Construction

We construct an SKE-CR-SKL scheme with key testability $\mathsf{SKE\text{-}CR\text{-}SKL} = \mathsf{SKE\text{-}CR\text{-}SKL}.(\mathsf{Setup},\mathcal{KG},\mathsf{Enc},\mathcal{Dec},\mathcal{Vrfy})$ having the additional algorithms CDec and $\mathsf{KeyTest}$, using the following building blocks.

- BB84-based SKE-CD scheme (Definition 3.12) $\mathsf{SKECD} = \mathsf{SKECD}.(\mathsf{KG},\mathcal{Enc},\mathcal{Dec},\mathcal{Del},\mathsf{Vrfy})$ having the classical decryption algorithm $\mathsf{SKECD.CDec}$.
- OWF $f : \{0,1\}^\lambda \to \{0,1\}^{p(\lambda)}$ for some polynomial p.

Let $\mathcal{M} := \{0,1\}^{\ell_{\mathsf{m}}}$ be the plaintext space. The construction is as follows:

$\mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$**:**

1. Generate $r \leftarrow \{0,1\}^{\ell_{\mathsf{m}}}$.
2. Generate $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$.
3. Output $\mathsf{msk} := (\mathsf{skecd.sk}, r)$.

$\mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk})$**:**

1. Parse $\mathsf{msk} = (\mathsf{skecd.sk}, r)$.
2. Generate $(\mathsf{skecd}.\mathcal{ct},\mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, r)$. $\mathsf{skecd.vk}$ is of the form $(x,\theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, and $\mathsf{skecd}.\mathcal{ct}$ is of the form $|\psi_1\rangle_{\mathsf{SKECD.CT}_1} \otimes \cdots \otimes |\psi_{\ell_{\mathsf{ct}}}\rangle_{\mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}}$.
3. Generate $s_{i,b} \leftarrow \{0,1\}^\lambda$ and compute $t_{i,b} \leftarrow f(s_{i,b})$ for every $i \in [\ell_{\mathsf{ct}}]$ and $b \in \{0,1\}$. Set $T := t_{1,0}\|t_{1,1}\|\cdots\|t_{\ell_{\mathsf{ct}},0}\|t_{\ell_{\mathsf{ct}},1}$ and $S = \{s_{i,0} \oplus s_{i,1}\}_{i\in[\ell_{\mathsf{ct}}]\,:\,\theta[i]=1}$.

4. Prepare a register $\mathsf{S_i}$ that is initialized to $|0^\lambda\rangle_{\mathsf{S_i}}$ for every $i \in [\ell_{\mathsf{ct}}]$.
5. For every $i \in [\ell_{\mathsf{ct}}]$, apply the map
$$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \to |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_i}}$$
to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ and obtain the resulting state ρ_i.
6. Output $\mathcal{dk} = (\rho_i)_{i\in[\ell_{\mathsf{ct}}]}$, $\mathsf{vk} = (x, \theta, S)$, and $\mathsf{tk} = T$.

$\mathsf{SKE\text{-}CR\text{-}SKL.Enc}(\mathsf{msk}, \mathsf{m})$:

1. Parse $\mathsf{msk} = (\mathsf{skecd.sk}, r)$.
2. Output $\mathsf{ct} := (\mathsf{skecd.sk}, r \oplus \mathsf{m})$.

$\mathsf{SKE\text{-}CR\text{-}SKL.CDec}(\mathsf{dk}, \mathsf{ct})$:

1. Parse $\mathsf{ct} = (\mathsf{skecd.sk}, z)$. Let $\widetilde{\mathsf{dk}}$ be the sub-string of dk on register $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$.
2. Output $z \oplus \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, \widetilde{\mathsf{dk}})$.

$\mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{Dec}(\mathcal{dk}, \mathsf{ct})$:

1. Parse $(\rho_i)_{i\in[\ell_{\mathsf{ct}}]}$. We denote the register holding ρ_i as $\mathsf{SKECD.CT_i} \otimes \mathsf{S_i}$ for every $i \in [\ell_{\mathsf{ct}}]$.
2. Prepare a register MSG of ℓ_{m} qubits that is initialized to $|0\cdots0\rangle_{\mathsf{MSG}}$.
3. Apply the map
$$\begin{aligned}&|u\rangle_{\bigotimes_{i\in[\ell_{\mathsf{ct}}]}\mathsf{SKECD.CT_i}} \otimes |w\rangle_{\mathsf{MSG}} \to\\ &|u\rangle_{\bigotimes_{i\in[\ell_{\mathsf{ct}}]}\mathsf{SKECD.CT_i}} \otimes |w \oplus \mathsf{SKE\text{-}CR\text{-}SKL.CDec}(u, \mathsf{ct})\rangle_{\mathsf{MSG}}\end{aligned}$$
to the registers $\bigotimes_{i\in[\ell_{\mathsf{ct}}]}\mathsf{SKECD.CT_i}$ and MSG.
4. Measure MSG in the computational basis and output the result m'.

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

1. Parse $\mathsf{vk} = (x, \theta, S = \{s_{i,0} \oplus s_{i,1}\}_{i\in[\ell_{\mathsf{ct}}]\,:\,\theta[i]=1})$ and $\mathcal{dk} = (\rho_i)_{i\in[\ell_{\mathsf{ct}}]}$ where ρ_i is a state on the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$.
2. For every $i \in [\ell_{\mathsf{ct}}]$, measure ρ_i in the Hadamard basis to get outcomes c_i, d_i corresponding to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ respectively.
3. Output $\top$ if $x[i] = c_i \oplus d_i \cdot (s_{i,0} \oplus s_{i,1})$ holds for every $i \in [\ell_{\mathsf{ct}}]$ such that $\theta[i] = 1$. Otherwise, output $\bot$.

$\mathsf{SKE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{tk}, \mathsf{dk})$:

1. Parse dk as a string over the registers $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ and $\mathsf{S} = \mathsf{S}_1 \otimes \cdots \otimes \mathsf{S}_{\ell_{\mathsf{ct}}}$. Let u_i denote the value on $\mathsf{SKECD.CT_i}$ and v_i the value on $\mathsf{S_i}$. Parse tk as $T = t_{1,0}\|t_{1,1}\|\cdots\|t_{\ell_{\mathsf{ct}},0}\|t_{\ell_{\mathsf{ct}},1}$.
2. Let $\mathsf{Check}[t_{i,0}, t_{i_1}](u_i, v_i)$ be the deterministic algorithm that outputs 1 if $f(v_i) = t_{i,u_i}$ holds and 0 otherwise.
3. Output $\mathsf{Check}[t_{1,0}, t_{1,1}](u_1, v_1) \wedge \cdots \wedge \mathsf{Check}[t_{\ell_{\mathsf{ct}},0}, t_{\ell_{\mathsf{ct}},1}](u_{\ell_{\mathsf{ct}}}, v_{\ell_{\mathsf{ct}}})$.

Decryption correctness: For a ciphertext $\mathsf{ct} = (\mathsf{skecd.sk}, r \oplus \mathsf{m})$, the decryption algorithm $\mathcal{Dec}$ performs the following computation:

$$
\begin{aligned}
&|u\rangle_{\bigotimes_{i\in[\ell_{\mathsf{ct}}]} \mathsf{SKECD.CT_i}} \otimes |w\rangle_{\mathsf{MSG}} \to \\
&|u\rangle_{\bigotimes_{i\in[\ell_{\mathsf{ct}}]} \mathsf{SKECD.CT_i}} \otimes |w \oplus r \oplus \mathsf{m} \oplus \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u)\rangle_{\mathsf{MSG}}
\end{aligned}
$$

Recall that $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ is a register corresponding to the ciphertext of a BB84-based SKE-CD scheme SKECD. Hence, from the Classical Decryption property of SKECD (Definition 3.12), it must be that $\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u) = r$ for every u in the superposition of $\mathsf{skecd}.ct$. Consequently, m is written onto the MSG register in each term of the superposition and decryption correctness follows.

Verification correctness: Observe that for the Hadamard basis positions ($i \in [\ell_{\mathsf{ct}}]$ such that $\theta[i] = 1$), ρ_i is of the form:

$$\rho[i] = |0\rangle_{\mathsf{SKECD.CT_i}} |s_{i,0}\rangle_{\mathsf{S_i}} + (-1)^{x[i]} |1\rangle_{\mathsf{SKECD.CT_i}} |s_{i,1}\rangle_{\mathsf{S_i}}$$

It is easy to see that measuring the state in the Hadamard basis gives outcomes c_i, d_i (on registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ respectively) satisfying $x[i] = c_i \oplus d_i \cdot (s_{i,0} \oplus s_{i,1})$. Hence, the verification correctness follows.

Theorem 5.1. *There exists an SKE-CR-SKL scheme satisfying OT-IND-KLA security and Key-Testability, assuming the existence of a BB84-based SKE-CD scheme and the existence of an OWF.*

We prove this theorem in the subsequent sections.

5.2 Proof of OT-IND-KLA Security

Let $\mathcal{A}$ be an adversary for the OT-IND-KLA security of the construction SKE-CR-SKL that makes use of a BB84-based SKE-CD scheme SKECD. Consider the hybrid $\mathsf{Hyb}_j^{\mathsf{coin}}$ defined as follows:

$\mathsf{Hyb}_j^{\mathsf{coin}}$**:**

1. The challenger $\mathcal{Ch}$ runs $\mathsf{msk} \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$ and initializes $\mathcal{A}$ with input 1^λ.
2. $\mathcal{A}$ requests q decryption keys for some polynomial q. For each $k \in [j]$, $\mathcal{Ch}$ generates $(\mathcal{dk}_i, \mathsf{vk}_i, \mathsf{tk}_i) \leftarrow \widetilde{\mathcal{KG}}(\mathsf{msk})$ where $\widetilde{\mathcal{KG}}$ is defined as follows (the difference from $\mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}$ is colored in red):

$\widetilde{\mathcal{KG}}(\mathsf{msk})$:

   (a) Parse $\mathsf{msk} = (\mathsf{skecd.sk}, r)$.
   (b) Sample $\widetilde{r} \leftarrow \{0,1\}^{\ell_{\mathsf{m}}}$.
   (c) Generate $(\mathsf{skecd}.ct, \mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, \widetilde{r})$. $\mathsf{skecd.vk}$ is of the form $(x, \theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, and $\mathsf{skecd}.ct$ can be described as $|\psi_1\rangle_{\mathsf{SKECD.CT_1}} \otimes \cdots \otimes |\psi_{\ell_{\mathsf{ct}}}\rangle_{\mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}}$.
   (d) Generate $s_{i,b} \leftarrow \{0,1\}^\lambda$ and compute $t_{i,b} \leftarrow f(s_{i,b})$ for every $i \in [\ell_{\mathsf{ct}}]$ and $b \in \{0,1\}$. Set $T := t_{1,0} \| t_{1,1} \| \cdots \| t_{\ell_{\mathsf{ct}},0} \| t_{\ell_{\mathsf{ct}},1}$ and $S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_{\mathsf{ct}}] : \theta[i]=1}$.
   (e) Prepare a register $\mathsf{S_i}$ that is initialized to $|0\cdots0\rangle_{\mathsf{S_i}}$ for every $i \in [\ell_{\mathsf{ct}}]$.
   (f) For every $i \in [\ell_{\mathsf{ct}}]$, apply the map

$$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \to |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_1}}$$

   to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ and obtain the resulting state ρ_i.
   (g) Output $\mathcal{dk} = (\rho_i)_{i\in[\ell_{\mathsf{ct}}]}$, $\mathsf{vk} = (x, \theta, S)$, and $\mathsf{tk} = T$.

3. On the other hand, for $k = j+1, \dots, q$, $\mathcal{Ch}$ generates $(\mathcal{dk}_k, \mathsf{vk}_k, \mathsf{tk}_k) \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk})$. Then, $\mathcal{Ch}$ sends $(\mathcal{dk}_k, \mathsf{tk}_k)_{k \in [q]}$ to $\mathcal{A}$.
4. $\mathcal{A}$ can get access to the following (stateful) verification oracle $O_{\mathcal{Vrfy}}$ where V_i is initialized to $\bot$:

   $O_{\mathcal{Vrfy}}(i, \widetilde{\mathcal{dk}})$**:** It runs $d \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{vk}_i, \widetilde{\mathcal{dk}})$ and returns d. If $V_i = \bot$ and $d = \top$, it updates $V_i := \top$.
5. $\mathcal{A}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \{0,1\}^{\ell_\mathsf{m}} \times \{0,1\}^{\ell_\mathsf{m}}$ to $\mathcal{Ch}$. If $V_i = \bot$ for some $i \in [q]$, $\mathcal{Ch}$ outputs 0 as the final output of this experiment. Otherwise, $\mathcal{Ch}$ generates $\mathsf{ct}^* \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Enc}(\mathsf{msk}, \mathsf{m}_{\mathsf{coin}}^*)$ and sends ct^* to $\mathcal{A}$.
6. $\mathcal{A}$ outputs a guess coin' for coin. The challenger outputs coin' as the final output of the experiment.

We will now prove the following lemma:

Lemma 5.2. $\forall j \in \{0, \dots, q-1\}$ *and* $\mathsf{coin} \in \{0,1\}$: $\mathsf{Hyb}_j^{\mathsf{coin}} \approx \mathsf{Hyb}_{j+1}^{\mathsf{coin}}$.

Proof. Suppose $\mathsf{Hyb}_j^{\mathsf{coin}} \not\approx \mathsf{Hyb}_{j+1}^{\mathsf{coin}}$. Let $\mathcal{D}$ be a corresponding distinguisher. We will construct a reduction $\mathcal{R}$ that breaks the IND-CVA-CD security of the BB84-based SKE-CD scheme SKECD. The execution of $\mathcal{R}^{\mathcal{D}}$ in the experiment $\mathsf{Exp}_{\mathsf{SKECD},\mathcal{R}}^{\mathsf{ind\text{-}cva\text{-}cd}}(1^\lambda, b)$ proceeds as follows:

Execution of $\mathcal{R}^{\mathcal{D}}$ in $\mathsf{Exp}_{\mathsf{SKECD},\mathcal{R}}^{\mathsf{ind\text{-}cva\text{-}cd}}(1^\lambda, b)$:

1. The challenger $\mathcal{Ch}$ computes $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$.
2. $\mathcal{R}$ samples $(r_0, r_1) \leftarrow \{0,1\}^{\ell_\mathsf{m}} \times \{0,1\}^{\ell_\mathsf{m}}$ and sends it to $\mathcal{Ch}$.
3. $\mathcal{Ch}$ computes $(\mathsf{skecd}.\mathit{ct}^\star, \mathsf{skecd.vk}^\star) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, r_b)$ and sends $\mathsf{skecd}.\mathit{ct}^\star$ to $\mathcal{R}$.
4. $\mathcal{R}$ initializes $\mathcal{D}$ with 1^λ. $\mathcal{D}$ requests q keys for some polynomial q.
5. For each $k \in [j]$, $\mathcal{R}$ computes $\mathcal{dk}_k$ as follows:
   - Sample a random value $\widetilde{r} \leftarrow \{0,1\}^{\ell_\mathsf{m}}$.
   - Compute $(\mathsf{skecd}.\widetilde{\mathit{ct}}, \mathsf{skecd}.\widetilde{\mathsf{vk}}) \leftarrow O_{\mathcal{Enc}}(\widetilde{r})$.
   - Compute $\mathcal{dk}_k$ by executing Steps 2.(c)-2.(g) as in Hyb_j^0, but using $\mathsf{skecd}.\widetilde{\mathit{ct}}$ in place of $\mathsf{skecd}.\mathit{ct}$.
6. $\mathcal{R}$ computes $\mathcal{dk}_{j+1}$ by executing Steps 2.(c)-2.(g) as in Hyb_j^0, but using $\mathsf{skecd}.\mathit{ct}^\star$ in place of $\mathsf{skecd}.\mathit{ct}$.
7. For each $k \in [j+2, q]$, $\mathcal{R}$ computes $\mathcal{dk}_k$ as follows:
   - Compute $(\mathsf{skecd}.\widetilde{\mathit{ct}}, \mathsf{skecd}.\widetilde{\mathsf{vk}}) \leftarrow O_{\mathcal{Enc}}(r_1)$.
   - Compute $\mathcal{dk}_k$ by executing Steps 2.(c)-2.(g) as in Hyb_j^0, but using $\mathsf{skecd}.\widetilde{\mathit{ct}}$ in place of $\mathsf{skecd}.\mathit{ct}$.
8. $\mathcal{R}$ sends $\mathcal{dk}_1, \dots, \mathcal{dk}_q$ to $\mathcal{D}$ and initializes $V_k = \bot$ for every $k \in [q]$.
9. If $k \neq j+1$, $\mathcal{R}$ simulates the response of oracle $O_{\mathcal{Vrfy}}(k, \widetilde{\mathcal{dk}})$ as follows:
   - Parse $\widetilde{\mathsf{vk}} = (x, \theta, S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_\mathsf{ct}] : \theta[i]=1})$ and $\widetilde{\mathcal{dk}} = (\rho_i)_{i \in [\ell_\mathsf{ct}]}$.
   - For every $i \in [\ell_\mathsf{ct}]$, measure ρ_i in the Hadamard basis to get outcomes c_i, d_i corresponding to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ respectively.
   - Compute $\mathsf{cert}[i] = c_i \oplus d_i \cdot (s_{i,0} \oplus s_{i,1})$ for every $i \in [\ell_\mathsf{ct}]$.
   - If $x[i] = \mathsf{cert}[i]$ holds for every $i \in [\ell_\mathsf{ct}] : \theta[i] = 1$, then update $V_k = \top$ and send $\top$ to $\mathcal{D}$. Else, send $\bot$.
10. If $k = j+1$, $\mathcal{R}$ simulates the response of oracle $O_{\mathcal{Vrfy}}(k, \widetilde{\mathcal{dk}})$ as follows:
    - Compute $\mathsf{cert} = \mathsf{cert}[1] \| \dots \| \mathsf{cert}[\ell_\mathsf{ct}]$, where each $\mathsf{cert}[i]$ is computed as in Step 9.
    - Send cert to $\mathcal{Ch}$. If $\mathcal{Ch}$ returns $\mathsf{skecd.sk}$, send $\top$ to $\mathcal{D}$ and update $V_{j+1} = \top$. Else if $\mathcal{Ch}$ returns $\bot$, send $\bot$ to $\mathcal{D}$.

11. $\mathcal{D}$ sends $(\mathsf{m}_0^\star, \mathsf{m}_1^\star) \in \{0,1\}^{\ell_{\mathsf{m}}} \times \{0,1\}^{\ell_{\mathsf{m}}}$ to $\mathcal{R}$. If $V_i = \bot$ for any $i \in [q]$, $\mathcal{R}$ sends 0 to $\mathcal{Ch}$. $\mathcal{R}$ computes $\mathsf{ct}^\star = (\mathsf{skecd.sk}, r_1 \oplus \mathsf{m}_{\mathsf{coin}}^\star)$, where $\mathsf{skecd.sk}$ is obtained from $\mathcal{Ch}$ in Step 10. $\mathcal{R}$ sends ct^* to $\mathcal{D}$.
12. $\mathcal{D}$ outputs a guess b' which $\mathcal{R}$ forwards to $\mathcal{Ch}$. $\mathcal{Ch}$ outputs b' as the final output of the experiment.

We will first argue that when $b = 1$, the view of $\mathcal{D}$ is exactly the same as its view in the hybrid $\mathsf{Hyb}_j^{\mathsf{coin}}$. Notice that the reduction computes the first j decryption keys by querying the encryption oracle on random plaintexts. Hyb_j on the other hand, directly computes them but there is no difference in the output ciphertexts. A similar argument holds for the keys $\mathcal{dk}_{j+2}, \ldots, \mathcal{dk}_q$, which contain encryptions of the same random value r_1. Moreover, if $b = 1$, the value encrypted as part of the key $\mathcal{dk}_{j+1}$ is also r_1. This is the same as in $\mathsf{Hyb}_j^{\mathsf{coin}}$. As for the verification oracle queries, notice that they are answered similarly by the reduction and $\mathsf{Hyb}_j^{\mathsf{coin}}$ for all but the $j+1$-th key. For the $j+1$-th key, the reduction works differently in that it forwards the certificate cert to the verification oracle. However, the verification procedure of the BB84-based SKE-CD scheme checks the validity of the value cert in the same way as the reduction, so there is no difference.

Finally, notice that when $b = 0$, the encrypted value is random and independent of r_1, similar to the hybrid $\mathsf{Hyb}_{j+1}^{\mathsf{coin}}$. Consequently, $\mathcal{R}$ breaks the IND-CVA-CD security of SKECD with non-negligible probability, a contradiction. ☐

Notice now that the hybrid $\mathsf{Hyb}_0^{\mathsf{coin}}$ is the same as the experiment $\mathsf{Exp}_{\mathsf{SKE\text{-}CR\text{-}SKL},\mathcal{A}}^{\mathsf{ot\text{-}ind\text{-}kla}}(1^\lambda, \mathsf{coin})$. From the previous lemma, we have that $\mathsf{Hyb}_0^{\mathsf{coin}} \approx \mathsf{Hyb}_q^{\mathsf{coin}}$. However, we have that $\mathsf{Hyb}_q^0 \approx \mathsf{Hyb}_q^1$ because Hyb_q^0 and Hyb_q^1 do not encrypt r at all as part of the decryption keys, but they mask the plaintext with r. Consequently, we have that $\mathsf{Hyb}_0^0 \approx \mathsf{Hyb}_0^1$, which completes the proof. ☐

5.3 Proof of Key-Testability

First, we will argue the correctness requirement. Recall that $\mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}$ applies the following map to a BB84 state $|x\rangle_\theta$, where $(x, \theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, for every $i \in [\ell_{\mathsf{ct}}]$:

$$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \to |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_i}}$$

where $\mathsf{SKECD.CT_i}$ denotes the register holding the i-th qubit of $|x\rangle_\theta$ and $\mathsf{S_i}$ is a register initialized to $|0\ldots0\rangle_{\mathsf{S_i}}$.

Consider applying the algorithm $\mathsf{SKE\text{-}CR\text{-}SKL.KeyTest}$ in superposition to the resulting state, i.e., performing the following map, where $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ and $\mathsf{S} = \mathsf{S}_1 \otimes \cdots \otimes \mathsf{S}_{\ell_{\mathsf{ct}}}$, and KT is initialized to $|0\rangle$:

$$|u\rangle_{\mathsf{SKECD.CT}} \otimes |v\rangle_{\mathsf{S}} \otimes |\beta\rangle_{\mathsf{KT}} \to |u\rangle_{\mathsf{SKECD.CT}} \otimes |v\rangle_{\mathsf{S}} \otimes |\beta \oplus \mathsf{SKE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{tk}, u\|v)\rangle_{\mathsf{KT}}$$

where $\mathsf{tk} = T = t_{1,0}\|t_{1,1}\| \cdots \|t_{\ell_{\mathsf{ct}},0}\|t_{\ell_{\mathsf{ct}},1}$. Recall that $\mathsf{SKE\text{-}CR\text{-}SKL.KeyTest}$ outputs 1 if and only if $\mathsf{Check}[t_{i,0}, t_{i,1}](u_i, v_i) = 1$ for every $i \in [\ell_{\mathsf{ct}}]$, where u_i, v_i denote the states of the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ respectively. Recall that $\mathsf{Check}[t_{i,0}, t_{i,1}](u_i, v_i)$ computes $f(v_i)$ and checks if it equals t_{i,u_i}. Since the construction chooses t_{i,u_i} such that $f(s_{i,u_i}) = t_{i,u_i}$, this check always passes. Consequently, measuring register KT always produces outcome 1.

We will now argue that the security requirement holds by showing the following reduction to the security of the OWF f. Let $\mathcal{A}$ be an adversary that breaks the key-testability of $\mathsf{SKE\text{-}CR\text{-}SKL}$. Consider a QPT reduction $\mathcal{R}$ that works as follows in the OWF security experiment:

Execution of $\mathcal{R}^{\mathcal{A}}$ in $\mathsf{Expt}_{f,\mathcal{R}}^{\mathsf{owf}}(1^\lambda)$:

1. The challenger chooses $s \leftarrow \{0,1\}^\lambda$ and sends $y := f(s)$ to $\mathcal{R}$.
2. $\mathcal{R}$ runs $\mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$ and initializes $\mathcal{A}$ with input msk.

3. $\mathcal{A}$ requests q decryption keys for some polynomial q. $\mathcal{R}$ picks a random index $k^\star \in [q]$. For every $k \neq k^\star$, $\mathcal{R}$ generates $(\mathcal{dk}_k, \mathsf{vk}_k, \mathsf{tk}_k)$ by computing the function f as needed. For the index $k^\star$, $\mathcal{R}$ computes $(\mathcal{dk}_{k^\star}, \mathsf{vk}_{k^\star}, \mathsf{tk}_{k^\star})$ as follows (the difference is colored in red):
   (a) Parse $\mathsf{msk} = (\mathsf{skecd.sk}, r)$.
   (b) Generate $(\mathsf{skecd}.\mathcal{ct}, \mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, r)$. $\mathsf{skecd.vk}$ is of the form $(x, \theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, $\mathsf{skecd}.\mathcal{ct}$ is of the form $|\psi_1\rangle_{\mathsf{SKECD.CT}_1} \otimes \cdots \otimes |\psi_{\ell_{\mathsf{ct}}}\rangle_{\mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}}$.
   (c) Choose an index $i^\star \in [\ell_{\mathsf{ct}}]$ such that $\theta[i^\star] = 0$. For every $i \in [\ell_{\mathsf{ct}}]$ such that $i \neq i^\star$, generate $s_{i,b} \leftarrow \{0,1\}^\lambda$ and compute $t_{i,b} \leftarrow f(s_{i,b})$ for every $b \in \{0,1\}$. For $i = i^\star$, set $t_{i^\star,1-x[i^\star]} := y$. Then, generate $s_{i^\star,x[i^\star]} \leftarrow \{0,1\}^\lambda$ and compute $t_{i^\star,x[i^\star]} = f(s_{i^\star,x[i^\star]})$. Set $T := t_{1,0} \| t_{1,1} \| \cdots \| t_{\ell_{\mathsf{ct}},0} \| t_{\ell_{\mathsf{ct}},1}$ and $S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_{\mathsf{ct}}] : \theta[i]=1}$.
   (d) Prepare a register S_i that is initialized to $|0 \cdots 0\rangle_{\mathsf{S}_\mathsf{i}}$ for every $i \in [\ell_{\mathsf{ct}}]$.
   (e) For every $i \in [\ell_{\mathsf{ct}}]$, apply the map
   $$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \to |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_i}}$$
   to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ and obtain the resulting state ρ_i.
   (f) Compute $\mathcal{dk}_{k^\star} = (\rho_i)_{i \in [\ell_{\mathsf{ct}}]}$, $\mathsf{vk}_{k^\star} = (x, \theta, S)$, and $\mathsf{tk}_{k^\star} = T$.
4. $\mathcal{R}$ sends $(\mathcal{dk}_i, \mathsf{vk}_i, \mathsf{tk}_i)$ to $\mathcal{A}$ for every $i \in [q]$.
5. $\mathcal{A}$ sends $(k, \mathsf{dk}, \mathsf{m})$ to $\mathcal{R}$. If $k \neq k^\star$, $\mathcal{R}$ aborts. Otherwise, $\mathcal{R}$ parses dk as a string over the registers $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ and $\mathsf{S} = \mathsf{S}_1 \otimes \cdots \otimes \mathsf{S}_{\ell_{\mathsf{ct}}}$ and measures the register $\mathsf{S}_{\mathsf{i}^\star}$ to obtain an outcome $s_{i^\star}$. $\mathcal{R}$ then sends $s_{i^\star}$ to the challenger.

Notice that the view of $\mathcal{A}$ is the same as its view in the key-testability experiment, as only the value $t_{i^\star,1-x[i^\star]}$ is generated differently by forwarding the value y, but this value is distributed identically to the original value. Note that in both cases, $\mathcal{A}$ receives no information about a pre-image of $t_{i^\star,1-x[i^\star]}$. Now, $\mathcal{R}$ guesses the index k that $\mathcal{A}$ targets with probability $\frac{1}{q}$. By assumption, we have that $\mathsf{CDec}(\mathsf{dk}, \mathsf{ct}) \neq \mathsf{m}$ where $\mathsf{ct} = \mathsf{Enc}(\mathsf{msk}, \mathsf{m})$. The value dk can be parsed as a string over the registers $\mathsf{SKECD.CT}$ and S. Let $\widetilde{\mathsf{dk}}$ be the sub-string of dk on the register $\mathsf{SKECD.CT}$. Recall that CDec invokes the algorithm $\mathsf{SKECD.CDec}$ on input $\widetilde{\mathsf{dk}}$. We will now recall a property of $\mathsf{SKECD.CDec}$ that was specified in Definition 3.12:

Let $(\mathcal{ct}, \mathsf{vk} = (x, \theta)) \leftarrow \mathsf{SKECD.Enc}(\mathsf{skecd.sk}, r)$ where $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$. Now, let u be any arbitrary value such that $u[i] = x[i]$ for all $i : \theta[i] = 0$. Then, the following holds:

$$\Pr\left[\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u) = r\right] \geq 1 - \mathsf{negl}(\lambda)$$

Consequently, if $\widetilde{\mathsf{dk}}$ is such that $\widetilde{\mathsf{dk}}[i] = x[i]$ for all $i : \theta[i] = 0$, where (x, θ) are specified by $\mathsf{vk}_{k^\star}$, then $\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, \widetilde{\mathsf{dk}})$ outputs the value r with high probability. Since $\mathsf{CDec}(\mathsf{dk}, \mathsf{ct} = (\mathsf{skecd.sk}, r \oplus \mathsf{m}))$ outputs $r \oplus \mathsf{m} \oplus \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, \widetilde{\mathsf{dk}})$, we have that $\mathsf{CDec}(\mathsf{dk}, \mathsf{ct} = (\mathsf{skecd.sk}, r \oplus \mathsf{m})) = \mathsf{m}$. Therefore, it must be the case that there exists some index i for which $\widetilde{\mathsf{dk}}[i] \neq x[i]$. With probability $\frac{1}{\ell_{\mathsf{ct}}}$, this happens to be the guessed value $i^\star$. In this case, $\mathcal{A}$ must output $s_{i^\star}$ on register $\mathsf{S_i}$ such that $f(s_{i^\star}) = t_{i^\star,1-x[i^\star]} = y$. This concludes the proof. ☐

Since we have proved OT-IND-KLA security (Section 5.2) and Key-Testability (Section 5.3), we can now state the following theorem:

6 PKE-CR-SKL from LWE

In this section, we show how to achieve PKE-CR-SKL from SKE-CR-SKL, standard ABE, and compute-and-compare obfuscation.

6.1 Construction

We construct a PKE-CR-SKL scheme $\mathsf{PKE\text{-}CR\text{-}SKL} = \mathsf{PKE\text{-}CR\text{-}SKL}.(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy})$ with message space $\mathcal{M} = \{0,1\}^{\ell}$ using the following building blocks.

- ABE scheme $\mathsf{ABE} = \mathsf{ABE}.(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ for the following relation R.

  $R(x, y)$**:** Interpret x as a circuit. Then, output 0 (decryptable) if $\bot = x(y)$ and otherwise 1.

- Compute-and-Compare Obfuscation $\mathsf{CC.Obf}$ with the simulator $\mathsf{CC.Sim}$.
- SKE-CR-SKL scheme with Key Testability $\mathsf{SKE\text{-}CR\text{-}SKL} = \mathsf{SKE\text{-}CR\text{-}SKL}.(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Vrfy}, \mathsf{KeyTest})$. It also has the classical decryption algorithm $\mathsf{SKE\text{-}CR\text{-}SKL.CDec}$.

The construction is as follows.

$\mathsf{PKE\text{-}CR\text{-}SKL.Setup}(1^{\lambda})$**:**

- Generate $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^{\lambda})$.
- Generate $\mathsf{ske.msk} \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^{\lambda})$.
- Output $\mathsf{ek} := \mathsf{abe.pk}$ and $\mathsf{msk} := (\mathsf{abe.msk}, \mathsf{ske.msk})$.

$\mathsf{PKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk})$**:**

- Parse $\mathsf{msk} = (\mathsf{abe.msk}, \mathsf{ske.msk})$.
- Generate $(\mathsf{ske}.\mathit{dk}, \mathsf{ske.vk}, \mathsf{ske.tk}) \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{ske.msk})$. We denote the register holding $\mathsf{ske}.\mathit{dk}$ as $\mathsf{SKE.DK}$.
- Prepare a register $\mathsf{ABE.SK}$ that is initialized to $|0\cdots0\rangle_{\mathsf{ABE.SK}}$.
- Choose explicit randomness $\mathsf{k} \leftarrow \{0,1\}^{\lambda}$.
- Apply the map $|u\rangle_{\mathsf{SKE.DK}} |v\rangle_{\mathsf{ABE.SK}} \rightarrow |u\rangle_{\mathsf{SKE.DK}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, u, \mathsf{k})\rangle_{\mathsf{ABE.SK}}$ to the registers $\mathsf{SKE.DK}$ and $\mathsf{ABE.SK}$, and obtain dk over the registers $\mathsf{SKE.DK}$ and $\mathsf{ABE.SK}$.
- Output dk and $\mathsf{vk} := (\mathsf{abe.msk}, \mathsf{ske.vk}, \mathsf{ske.tk}, \mathsf{k})$.

$\mathsf{PKE\text{-}CR\text{-}SKL.Enc}(\mathsf{ek}, \mathsf{m})$**:**

- Parse $\mathsf{ek} = \mathsf{abe.pk}$.
- Generate $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^{\lambda}, \mathsf{pp}_D, 1)$, where pp_D consists of circuit parameters of D defined in the security proof.
- Generate $\mathsf{abe.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, \widetilde{C}, \mathsf{m})$.
- Output $\mathsf{ct} := \mathsf{abe.ct}$.

$\mathsf{PKE\text{-}CR\text{-}SKL}.\mathcal{Dec}(\mathit{dk}, \mathsf{ct})$**:**

- Parse $\mathsf{ct} = \mathsf{abe.ct}$. We denote the register holding dk as $\mathsf{SKE.DK} \otimes \mathsf{ABE.SK}$.
- Prepare a register MSG that is initialized to $|0\cdots0\rangle_{\mathsf{MSG}}$
- Apply the map $|v\rangle_{\mathsf{ABE.SK}} |w\rangle_{\mathsf{MSG}} \rightarrow |v\rangle_{\mathsf{ABE.SK}} |w \oplus \mathsf{ABE.Dec}(v, \mathsf{abe.ct})\rangle_{\mathsf{MSG}}$ to the registers $\mathsf{ABE.SK}$ and MSG.
- Measure the register MSG in the computational basis and output the result m'.

$\mathsf{PKE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{vk}, \mathit{dk}')$**:**

- Parse $\mathsf{vk} = (\mathsf{abe.msk}, \mathsf{ske.vk}, \mathsf{ske.tk}, \mathsf{k})$. We denote the register holding dk' as $\mathsf{SKE.DK} \otimes \mathsf{ABE.SK}$.

- Prepare a register $\mathsf{SKE.KT}$ that is initialized to $|0\rangle_{\mathsf{SKE.KT}}$.
- Apply the map $|u\rangle_{\mathsf{SKE.DK}} |\beta\rangle_{\mathsf{SKE.KT}} \to |u\rangle_{\mathsf{SKE.DK}} |\beta \oplus \mathsf{SKE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{ske.tk}, u)\rangle_{\mathsf{SKE.KT}}$ to the registers $\mathsf{SKE.DK}$ and $\mathsf{SKE.KT}$.
- Measure $\mathsf{SKE.KT}$ in the computational basis and output $\perp$ if the result is 0. Otherwise, go to the next step.
- Apply the map $|u\rangle_{\mathsf{SKE.DK}} |v\rangle_{\mathsf{ABE.SK}} \to |u\rangle_{\mathsf{SKE.DK}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, u, \mathsf{k})\rangle_{\mathsf{ABE.SK}}$ to the registers $\mathsf{SKE.DK}$ and $\mathsf{ABE.SK}$.
- Trace out the register $\mathsf{ABE.SK}$ and obtain $\mathsf{ske}.\mathcal{dk}'$ over $\mathsf{SKE.DK}$.
- Output $\top$ if $\top = \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{ske.vk}, \mathsf{ske}.\mathcal{dk}')$ and $\perp$ otherwise.

Decryption correctness. The key $\mathcal{dk}$ output by $\mathsf{PKE\text{-}CR\text{-}SKL}.\mathcal{KG}$ is of the form $\sum_u \alpha_u |u\rangle_{\mathsf{SKE.DK}} |\mathsf{abe.sk}_u\rangle_{\mathsf{ABE.SK}}$, where $\mathsf{abe.sk}_u \leftarrow \mathsf{ABE.KG}(\mathsf{abe.msk}, u, \mathsf{k})$. Let $\mathsf{ct} \leftarrow \mathsf{PKE\text{-}CR\text{-}SKL.Enc}(\mathsf{ek}, \mathsf{m})$. On applying $|v\rangle_{\mathsf{ABE.SK}} |w\rangle_{\mathsf{MSG}} \to |v\rangle_{\mathsf{ABE.SK}} |w \oplus \mathsf{ABE.Dec}(v, \mathsf{abe.ct})\rangle_{\mathsf{MSG}}$ to $\sum_u \alpha_u |u\rangle_{\mathsf{SKE.DK}} |\mathsf{abe.sk}_u\rangle_{\mathsf{ABE.SK}} \otimes |0\cdots 0\rangle_{\mathsf{MSG}}$, with overwhelming probability, the result is negligibly close to

$$\sum_u \alpha_u |u\rangle_{\mathsf{SKE.DK}} |\mathsf{abe.sk}_u\rangle_{\mathsf{ABE.SK}} \otimes |\mathsf{m}\rangle_{\mathsf{MSG}}$$

since $\widetilde{C}(u) = \perp$ and thus $R(\widetilde{C}, u) = 0$ for $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$ and almost every string u. Therefore, we see that $\mathsf{PKE\text{-}CR\text{-}SKL}$ satisfies decryption correctness.

Verification correctness. Let $\mathcal{dk} \leftarrow \mathsf{PKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk})$. It is clear that the state $\mathsf{ske}.\mathcal{dk}'$ obtained when computing $\mathsf{PKE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{vk}, \mathcal{dk})$ is the same as $\mathsf{ske}.\mathcal{dk}$ generated when generating $\mathcal{dk}$. Therefore, the verification correctness of $\mathsf{PKE\text{-}CR\text{-}SKL}$ follows from that of $\mathsf{SKE\text{-}CR\text{-}SKL}$.

6.2 Proof of IND-KLA Security

Let $\mathcal{A}$ be an adversary for the IND-KLA security of $\mathsf{PKE\text{-}CR\text{-}SKL}$. We consider the following sequence of experiments.

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

1. The challenger $\mathcal{Ch}$ generates $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and $\mathsf{ske.msk} \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$, and sends $\mathsf{ek} := \mathsf{abe.pk}$ to $\mathcal{A}$.
2. $\mathcal{A}$ requests q decryption keys for some polynomial q. $\mathcal{Ch}$ generates $\mathcal{dk}_i$ as follows for every $i \in [q]$:
   - Generate $(\mathsf{ske}.\mathcal{dk}_i, \mathsf{ske.vk}_i, \mathsf{ske.tk}_i) \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{ske.msk})$. We denote the register holding $\mathsf{ske}.\mathcal{dk}_i$ as $\mathsf{SKE.DK_i}$.
   - Prepare a register $\mathsf{ABE.SK_i}$ that is initialized to $|0\cdots 0\rangle_{\mathsf{ABE.SK_i}}$.
   - Choose explicit randomness $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
   - Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |v\rangle_{\mathsf{ABE.SK_i}} \to |u\rangle_{\mathsf{SKE.DK_i}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, u, \mathsf{k}_i)\rangle_{\mathsf{ABE.SK_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.SK_i}$, and obtain $\mathcal{dk}_i$ over the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.SK_i}$.

   $\mathcal{Ch}$ sends $\mathcal{dk}_1, \ldots, \mathcal{dk}_q$ to $\mathcal{A}$.
3. $\mathcal{A}$ can get access to the following (stateful) verification oracle $O_{\mathcal{Vrfy}}$ where V_i is initialized to be $\perp$:

   $O_{\mathcal{Vrfy}}(i, \widetilde{\mathcal{dk}})$**:** It computes d as follows.

   (a) Let the register holding $\widetilde{\mathcal{dk}}$ be $\mathsf{SKE.DK_i} \otimes \mathsf{ABE.SK_i}$.

   (b) Prepare a register $\mathsf{SKE.KT_i}$ that is initialized to $|0\rangle_{\mathsf{SKE.KT_i}}$.

   (c) Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |\beta\rangle_{\mathsf{SKE.KT_i}} \to |u\rangle_{\mathsf{SKE.DK_i}} |\beta \oplus \mathsf{SKE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{ske.tk}_i, u)\rangle_{\mathsf{SKE.KT_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{SKE.KT_i}$.

(d) Measure $\mathsf{SKE.KT_i}$ in the computational basis and set $d := \bot$ if the result is 0. Otherwise, go to the next step.

(e) Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |v\rangle_{\mathsf{ABE.SK_i}} \to |u\rangle_{\mathsf{SKE.DK_i}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, u, \mathsf{k}_i)\rangle_{\mathsf{ABE.SK_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.SK_i}$.

(f) Trace out the register $\mathsf{ABE.SK_i}$ and obtain $\mathsf{ske}.dk'$ over $\mathsf{SKE.DK_i}$.

(g) Set $d := \top$ if $\top = \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{V}rfy(\mathsf{ske.vk}_i, \mathsf{ske}.dk')$ and set $d := \bot$ otherwise. It returns d to $\mathcal{A}$. Finally, if $V_i = \bot$ and $d = \top$, it updates $V_i := \top$.

4. $\mathcal{A}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ to $\mathcal{C}h$. If $V_i = \bot$ for some $i \in [q]$, $\mathcal{C}h$ outputs 0 as the final output of this experiment. Otherwise, $\mathcal{C}h$ generates $\widetilde{C}^* \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$ and $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, \widetilde{C}^*, \mathsf{m}_{\mathsf{coin}}^*)$, and sends $\mathsf{ct}^* := \mathsf{abe.ct}^*$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs coin'. $\mathcal{C}h$ outputs coin' as the final output of the experiment.

$\mathsf{Hyb}_1^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_0^{\mathsf{coin}}$ except that $\widetilde{C}^*$ is generated as $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{ske.ct}^*], \mathsf{lock}, 0)$, where $\mathsf{ske.ct}^* \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Enc}(\mathsf{ske.msk}, 0^\lambda)$, $\mathsf{lock} \leftarrow \{0,1\}^\lambda$, and $D[\mathsf{ske.ct}^*](x)$ is a circuit that has $\mathsf{ske.ct}^*$ hardwired and outputs $\mathsf{SKE\text{-}CR\text{-}SKL.CDec}(x, \mathsf{ske.ct}^*)$.

We pick lock as a uniformly random string that is completely independent of other variables such as $\mathsf{ske.ct}^*$. Thus, from the security of CC.Obf, we have $\mathsf{Hyb}_0^{\mathsf{coin}} \approx \mathsf{Hyb}_1^{\mathsf{coin}}$.

$\mathsf{Hyb}_2^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_1^{\mathsf{coin}}$ except that $\mathsf{ske.ct}^*$ hardwired into the obfuscated circuit $\widetilde{C}^*$ is generated as $\mathsf{ske.ct}^* \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Enc}(\mathsf{ske.msk}, \mathsf{lock})$.

From the OT-IND-KLA security of $\mathsf{SKE\text{-}CR\text{-}SKL}$, we can show that $\mathsf{Hyb}_1^{\mathsf{coin}} \approx \mathsf{Hyb}_2^{\mathsf{coin}}$. Suppose that $\mathsf{Hyb}_1^{\mathsf{coin}} \not\approx \mathsf{Hyb}_2^{\mathsf{coin}}$ and $\mathcal{D}$ is a corresponding distinguisher. We consider the following reduction $\mathcal{R}$:

Execution of $\mathcal{R}^{\mathcal{D}}$ in $\mathsf{Exp}_{\mathsf{SKE\text{-}CR\text{-}SKL},\mathcal{R}}^{\mathsf{ot\text{-}ind\text{-}kla}}(1^\lambda, b)$:

1. $\mathcal{C}h$ runs $\mathsf{ske.msk} \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$ and initializes $\mathcal{R}$ with the security parameter 1^λ.
2. $\mathcal{R}$ generates $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and sends $\mathsf{ek} := \mathsf{abe.pk}$ to $\mathcal{D}$.
3. $\mathcal{D}$ requests q decryption keys for some polynomial q. $\mathcal{R}$ requests q decryption keys. $\mathcal{C}h$ generates $(\mathsf{ske}.dk_i, \mathsf{ske.vk}_i, \mathsf{ske.tk}_i) \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk})$ for every $i \in [q]$ and sends $(\mathsf{ske}.dk_i, \mathsf{ske.tk}_i)_{i\in[q]}$ to $\mathcal{R}$.
4. $\mathcal{R}$ computes $dk_1, \ldots, dk_q$ as in Step 2. of $\mathsf{Hyb}_0^{\mathsf{coin}}$, except that the received values $(\mathsf{ske}.dk_i)_{i\in[q]}$ are used instead of the original ones.
5. $\mathcal{R}$ simulates the access to $O_{\mathcal{V}rfy}(i, \widetilde{dk})$ for $\mathcal{D}$ as follows:

   $O_{\mathcal{V}rfy}(i, \widetilde{dk})$:

   (a) Perform Step 3.(a)-3.(f) of $\mathsf{Hyb}_0^{\mathsf{coin}}$ to obtain $\mathsf{ske}.dk'$, but using the received value $\mathsf{ske.tk}_i$ instead of the original one.

   (b) Set $d := \top$ if $\top = \mathsf{SKE\text{-}CR\text{-}SKL}.O_{\mathcal{V}rfy}(i, \mathsf{ske}.dk')$ and set $d := \bot$ otherwise. It returns d to $\mathcal{D}$. Finally, if $V_i = \bot$ and $d = \top$, it updates $V_i = \top$.
6. $\mathcal{R}$ sends $(\mathsf{ske.m}_0^*, \mathsf{ske.m}_1^*) := (0^\lambda, \mathsf{lock})$ to $\mathcal{C}h$ and receives $\mathsf{ske.ct}^* \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Enc}(\mathsf{ske.msk}, \mathsf{ske.m}_b^*)$.
7. $\mathcal{D}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ to $\mathcal{R}$. If $V_i = \bot$ for some $i \in [q]$, $\mathcal{R}$ outputs 0. Otherwise, $\mathcal{R}$ generates $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{ske.ct}^*], \mathsf{lock}, 0)$ and $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, \widetilde{C}^*, \mathsf{m}_{\mathsf{coin}}^*)$ and sends $\mathsf{ct}^* := \mathsf{abe.ct}^*$ to $\mathcal{D}$.
8. $\mathcal{D}$ outputs a bit b'. $\mathcal{R}$ outputs b' and $\mathcal{C}h$ outputs b' as the final output of the experiment.

It is easy to see that the view of $\mathcal{D}$ is the same as that in $\mathsf{Hyb}_2^{\mathsf{coin}}$ when lock is encrypted in $\mathsf{ske.ct}^*$ and that of $\mathsf{Hyb}_1^{\mathsf{coin}}$ when 0^λ is encrypted. Moreover, for $\mathcal{D}$ to distinguish between the two hybrids, it must be the case that $V_i = \top$ for all $i \in [q]$, which directly implies that the q analogous values checked by $\mathsf{SKE\text{-}CR\text{-}SKL}.O_{\mathcal{V}rfy}$ must also be $\top$. Consequently, $\mathcal{R}$ breaks the OT-IND-KLA security of $\mathsf{SKE\text{-}CR\text{-}SKL}$. Therefore, it must be that $\mathsf{Hyb}_1^{\mathsf{coin}} \approx \mathsf{Hyb}_2^{\mathsf{coin}}$.

$\mathsf{Hyb}_3^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_2^{\mathsf{coin}}$ except that $\mathcal{dk}_i$ is generated as follows for every $i \in [q]$. (The difference is colored in red.)

- Generate $(\mathsf{ske}.\mathcal{dk}_i, \mathsf{ske.vk}_i, \mathsf{ske.tk}_i) \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{ske.msk})$. We denote the register holding $\mathsf{ske}.\mathcal{dk}_i$ as $\mathsf{SKE.DK_i}$.
- Prepare a register $\mathsf{ABE.R_i}$ that is initialized to $|0\rangle_{\mathsf{ABE.R_i}}$.
- Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |\beta\rangle_{\mathsf{ABE.R_i}} \rightarrow |u\rangle_{\mathsf{SKE.DK_i}} \left|\beta \oplus R(\widetilde{\mathsf{C}}^*, u)\right\rangle_{\mathsf{ABE.R_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.R_i}$. (Note that we can generate $\widetilde{\mathsf{C}}^*$ at the beginning of the game.)
- Measure $\mathsf{ABE.R_i}$ in the computational basis and set $\mathcal{dk}_i := \bot$ if the result is 0. Otherwise, go to the next step.
- Prepare a register $\mathsf{ABE.SK_i}$ that is initialized to $|0 \cdots 0\rangle_{\mathsf{ABE.SK_i}}$.
- Sample explicit randomness $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
- Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |v\rangle_{\mathsf{ABE.SK_i}} \rightarrow |u\rangle_{\mathsf{SKE.DK_i}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, u, \mathsf{k}_i)\rangle_{\mathsf{ABE.SK_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.SK_i}$, and obtain $\mathcal{dk}_i$ over the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.SK_i}$.

From the decryption correctness of SKE-CR-SKL, the added procedure that checks $R(\widetilde{\mathsf{C}}^*, u)$ in superposition does not affect the final state $\mathcal{dk}_i$ with overwhelming probability since $R(\widetilde{\mathsf{C}}^*, u) = 1$ in this hybrid for any u that appears in $\mathsf{ske}.\mathcal{dk}_i$ when describing it in the computational basis. Therefore, we have $\mathsf{Hyb}_2^{\mathsf{coin}} \approx \mathsf{Hyb}_3^{\mathsf{coin}}$.

$\mathsf{Hyb}_4^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_3^{\mathsf{coin}}$ except that the oracle $O_{\mathcal{Vrfy}}$ behaves as follows. (The difference is colored in red.)

$O_{\mathcal{Vrfy}}(i, \widetilde{\mathcal{dk}})$**:** It computes d as follows.

(a) Let the register holding $\widetilde{\mathcal{dk}}$ be $\mathsf{SKE.DK_i} \otimes \mathsf{ABE.SK_i}$.
(b) Prepare a register $\mathsf{SKE.KT_i}$ that is initialized to $|0\rangle_{\mathsf{SKE.KT_i}}$.
(c) Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |\beta\rangle_{\mathsf{SKE.KT_i}} \rightarrow |u\rangle_{\mathsf{SKE.DK_i}} |\beta \oplus \mathsf{SKE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{ske.tk}_i, u)\rangle_{\mathsf{SKE.KT_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{SKE.KT_i}$.
(d) Measure $\mathsf{SKE.KT_i}$ in the computational basis and set $d := \bot$ if the result is 0. Otherwise, go to the next step.
(e) Prepare a register $\mathsf{ABE.R_i}$ that is initialized to $|0\rangle_{\mathsf{ABE.R_i}}$.
(f) Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |\beta\rangle_{\mathsf{ABE.R_i}} \rightarrow |u\rangle_{\mathsf{SKE.DK_i}} \left|\beta \oplus R(\widetilde{\mathsf{C}}^*, u)\right\rangle_{\mathsf{ABE.R_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.R_i}$.
(g) Measure $\mathsf{ABE.R_i}$ in the computational basis and set $d := \bot$ if the result is 0. Otherwise, go to the next step.
(h) Apply the map $|u\rangle_{\mathsf{SKE.DK_i}} |v\rangle_{\mathsf{ABE.SK_i}} \rightarrow |u\rangle_{\mathsf{SKE.DK_i}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, u, \mathsf{k}_i)\rangle_{\mathsf{ABE.SK_i}}$ to the registers $\mathsf{SKE.DK_i}$ and $\mathsf{ABE.SK_i}$.
(i) Trace out the register $\mathsf{ABE.SK_i}$ and obtain $\mathsf{ske}.\mathcal{dk}'$ over $\mathsf{SKE.DK_i}$.
(j) Set $d := \top$ if $\top = \mathsf{SKE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{ske.vk}_i, \mathsf{ske}.\mathcal{dk}')$ and set $d := \bot$ otherwise. Return d to $\mathcal{A}$. Finally, if $V_i = \bot$ and $d = \top$, update $V_i := \top$.

Suppose there exists a QPT distinguisher $\mathcal{D}$ that has non-negligible advantage in distinguishing $\mathsf{Hyb}_3^{\mathsf{coin}}$ and $\mathsf{Hyb}_4^{\mathsf{coin}}$. Let $\mathcal{D}$ make $q = \mathrm{poly}(\lambda)$ many queries to the oracle $O_{\mathcal{Vrfy}}(\cdot,\cdot)$. We will now consider the following QPT algorithm $\mathcal{A}_{\mathsf{o2h}}$ with access to an oracle $O_{\mathcal{KG}}$ and an oracle $\mathcal{O}$ that runs $\mathcal{D}$ as follows:

$\underline{\mathcal{A}_{\mathsf{o2h}}^{O_{\mathcal{KG}}, \mathcal{O}}(\mathsf{abe.pk}, \mathsf{ske.msk})}$:

1. $\mathcal{A}_{\mathsf{o2h}}$ initializes $\mathcal{D}$ with input $\mathsf{ek} = \mathsf{abe.pk}$.

2. When $\mathcal{D}$ requests q decryption keys, $\mathcal{A}_{\mathsf{o2h}}$ queries $O_{\mathcal{KG}}$ on input q and receives $(\{\mathcal{dk}_i\}_{i\in[q]}, \{\mathsf{vk}_i\}_{i\in[q]})$. It forwards $\{\mathcal{dk}_i\}_{i\in[q]}$ to $\mathcal{D}$.
3. $\mathcal{A}_{\mathsf{o2h}}$ simulates the access of $O_{\mathcal{Vrfy}}$ for $\mathcal{D}$ as follows:

   $\underline{O_{\mathcal{Vrfy}}(y, \widetilde{\mathcal{dk}})}$:

   (a) Execute Steps (a)-(e) of $O_{\mathcal{Vrfy}}$ as in $\mathsf{Hyb}_4^{\mathsf{coin}}$.
   (b) Apply the map $|u\rangle_{\mathsf{SKE.DK_y}} |\beta\rangle_{\mathsf{ABE.R_y}} \rightarrow |u\rangle_{\mathsf{SKE.DK_y}} |\beta \oplus \mathcal{O}(u)\rangle_{\mathsf{ABE.R_y}}$ to the registers $\mathsf{SKE.DK_y}$ and $\mathsf{ABE.R_y}$.
   (c) Execute Steps (g)-(j) of $O_{\mathcal{Vrfy}}$ as in $\mathsf{Hyb}_4^{\mathsf{coin}}$.
4. $\mathcal{D}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ to $\mathcal{A}_{\mathsf{o2h}}$. If $V_i = \bot$ for some $i \in [q]$, $\mathcal{A}_{\mathsf{o2h}}$ outputs 0. Otherwise, $\mathcal{A}_{\mathsf{o2h}}$ generates $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{ske.ct}^*], \mathsf{lock}, 0)$, where $\mathsf{ske.ct}^* \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Enc}(\mathsf{ske.msk}, \mathsf{lock})$. It then generates $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, \widetilde{C}^*, \mathsf{m}^*_{\mathsf{coin}})$ and sends $\mathsf{ct}^* := \mathsf{abe.ct}^*$ to $\mathcal{D}$.
5. $\mathcal{D}$ outputs a guess b'. $\mathcal{A}_{\mathsf{o2h}}$ outputs b'.

Let H be an oracle that for every input u, outputs 1. Consider now the extractor $\mathcal{B}_{\mathsf{o2h}}^{O_{\mathcal{KG}},H}$ as specified by the O2H Lemma (Lemma 3.2). We will now construct a reduction $\mathcal{R}$ that runs $\mathcal{B}_{\mathsf{o2h}}$ by simulating the oracles $O_{\mathcal{KG}}$ and H for $\mathcal{B}_{\mathsf{o2h}}$, and breaks the key-testability of the SKE-CR-SKL scheme.

Execution of $\mathcal{R}$ in $\mathsf{Exp}^{\mathsf{key\text{-}test}}_{\mathsf{SKE\text{-}CR\text{-}SKL},\mathcal{R}}(1^\lambda)$:

1. The challenger $\mathcal{Ch}$ runs $\mathsf{ske.msk} \leftarrow \mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$ and initializes $\mathcal{R}$ with input $\mathsf{ske.msk}$.
2. $\mathcal{R}$ samples $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and initializes $\mathcal{B}_{\mathsf{o2h}}$ with the input $(\mathsf{abe.pk}, \mathsf{ske.msk})$.
3. When $\mathcal{B}_{\mathsf{o2h}}$ queries input q to $O_{\mathcal{KG}}$, $\mathcal{R}$ generates the decryption-keys $\mathcal{dk}_1, \ldots, \mathcal{dk}_q$ in the same way as in $\mathsf{Hyb}_3^{\mathsf{coin}}$ and sends them to $\mathcal{B}_{\mathsf{o2h}}$.
4. When $\mathcal{B}_{\mathsf{o2h}}$ queries an input u to H, $\mathcal{R}$ responds with 1.
5. $\mathcal{B}_{\mathsf{o2h}}$ outputs measured index y and measurement outcome dk. $\mathcal{R}$ sends $(y, \mathsf{dk}, \mathsf{lock})$ to $\mathcal{Ch}$.

We will now claim that with non-negligible probability, $\mathcal{R}$ obtains values dk and y such that $R(\widetilde{C}^*, \mathsf{dk}) = 0$. By the definition of R and $\widetilde{C}^*$ and the decryption correctness of SKE-CR-SKL, this will imply that $\mathsf{SKE\text{-}CR\text{-}SKL.CDec}(\mathsf{dk}, \mathsf{ske.ct}^\star) \neq \mathsf{lock}$. Moreover, $\mathsf{KeyTest}(\mathsf{ske.tk}_y, \mathsf{dk})$ also holds. Consequently, this will imply $\mathcal{R}$ breaks the key-testability of SKE-CR-SKL. To prove this, we will rely on the One-Way to Hiding (O2H) Lemma (Lemma 3.2). Consider an oracle G which takes as input u and outputs $R(\widetilde{C}^*, u)$ and an oracle H which takes as input u and outputs 1. Notice that if the oracle $\mathcal{O} = G$, then the view of $\mathcal{D}$ as run by $\mathcal{A}_{\mathsf{o2h}}$ is the same as in $\mathsf{Hyb}_4^{\mathsf{coin}}$, while if $\mathcal{O} = H$, the view of $\mathcal{D}$ is the same as in $\mathsf{Hyb}_3^{\mathsf{coin}}$. By the O2H Lemma, we have the following, where $z = (\mathsf{abe.pk}, \mathsf{ske.msk})$.

$$\left|\Pr\left[\mathcal{A}_{\mathsf{o2h}}^{O_{\mathcal{KG}},H}(z) = 1\right] - \Pr\left[\mathcal{A}_{\mathsf{o2h}}^{O_{\mathcal{KG}},G}(z) = 1\right]\right| \leq 2q \cdot \sqrt{\Pr\left[\mathcal{B}_{\mathsf{o2h}}^{O_{\mathcal{KG}},H}(z) \in S\right]} \ .$$

where S is a set where the oracles H and G differ, which happens only for inputs u s.t. $R(\widetilde{C}^*, u) = 0$. Since $\mathcal{R}$ obtains dk and y as the output of $\mathcal{B}_{\mathsf{o2h}}^{O_{\mathcal{KG}},H}(z)$, the argument follows that $\mathsf{Hyb}_3^{\mathsf{coin}} \approx \mathsf{Hyb}_4^{\mathsf{coin}}$.

$\mathsf{Hyb}_5^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_4^{\mathsf{coin}}$ except that $\mathsf{ct}^* := \mathsf{abe.ct}^*$ is generated as $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, \widetilde{C}^*, 0^{\ell_{\mathsf{m}}})$.

The view of $\mathcal{A}$ in $\mathsf{Hyb}_4^{\mathsf{coin}}$ and $\mathsf{Hyb}_5^{\mathsf{coin}}$ can be simulated with $\mathsf{abe.pk}$ and the access to the quantum key generation oracle O_{qkg}. This is because before $\mathsf{ABE.KG}$ is required to be applied in both the generation of $\{\mathcal{dk}_i\}_{i\in[q]}$ and to compute the responses of $O_{\mathcal{Vrfy}}$, the relation check $R(\widetilde{C}^*, u)$ is already applied in superposition. Thus, we have $\mathsf{Hyb}_4^{\mathsf{coin}} \approx \mathsf{Hyb}_5^{\mathsf{coin}}$.

Lastly, Hyb_5^0 and Hyb_5^1 are exactly the same experiment and thus we have $\left|\Pr\left[\mathsf{Hyb}_5^0 = 1\right] - \Pr\left[\mathsf{Hyb}_5^1 = 1\right]\right| = \mathsf{negl}(\lambda)$. Then, from the above arguments, we obtain

$$\begin{aligned}&\left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{PKE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda,0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}kla}}_{\mathsf{PKE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda,1) = 1\right]\right|\\ =&\left|\Pr\left[\mathsf{Hyb}_0^0 = 1\right] - \Pr\left[\mathsf{Hyb}_0^1 = 1\right]\right| \le \mathsf{negl}(\lambda).\end{aligned}$$

This completes the proof. □

Given the fact that SKE-CR-SKL with Key-Testability (implied by BB84-based SKE-CD and OWFs), Compute-and-Compare Obfuscation, and Ciphertext-Policy ABE for General Circuits are all implied by the LWE assumption, we state the following theorem:

Theorem 6.1. *There exists a PKE-CR-SKL scheme satisfying IND-KLA security, assuming the polynomial hardness of the LWE assumption.*

7 ABE-CR-SKL from LWE

In this section, we show how to achieve ABE-CR-SKL from the LWE assumption. To this end, we also introduce SKFE-CR-SKL with classical decryption and key-testability. First, we recall the standard SKFE.

Definition 7.1 (Secret-Key Functional Encryption). *An SKFE scheme* SKFE *for the functionality* $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ *is a tuple of four PPT algorithms* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

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

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

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

$\mathsf{Dec}(\mathsf{sk}_y, \mathsf{ct}) \to z$**:** *The decryption algorithm takes a functional decryption key* sk_y *and a ciphertext* ct*, and outputs* $z \in \{\bot\} \cup \mathcal{Z}$.

Correctness: *We require that for every* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$*, we have that*

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

Definition 7.2 (Selective Single-Ciphertext Security). *We formalize the experiment* $\mathsf{Exp}^{\mathsf{adp\text{-}ind}}_{\mathsf{SKFE},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger for an SKFE scheme for the functionality* $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ *as follows:*

1. *Initialized with* 1^λ, $\mathcal{A}$ *outputs* (x_0^*, x_1^*). *The challenger* Ch *runs* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{msk}, x^*_{\mathsf{coin}})$ *to* $\mathcal{A}$.
2. $\mathcal{A}$ *can get access to the following oracle.*

   $O_{\mathcal{KG}}(y)$**:** *Given* y*, if* $F(x_0^*, y) \neq F(x_1^*, y)$*, returns* $\bot$*. Otherwise, it returns* $\mathsf{sk}_y \leftarrow \mathsf{KG}(\mathsf{msk}, y)$.
3. $\mathcal{A}$ *outputs a guess* coin' *for* coin. $\mathcal{Ch}$ *outputs* coin' *as the final output of the experiment.*

We say that SKFE *satisfies selective single-ciphertext security if, for any QPT* $\mathcal{A}$*, it holds that*

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

Theorem 7.3 ([GVW12]). *Assuming the existence of OWFs, there exists selective single-ciphertext secure SKFE.*

7.1 Definitions of SKFE-CR-SKL

We consider the classical decryption property and key-testability for SKFE-CR-SKL as in the case of SKE-CR-SKL.

Definition 7.4 (SKFE-CR-SKL). *An SKFE-CR-SKL scheme* $\mathsf{SKFE\text{-}CR\text{-}SKL}$ *for the functionality* $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$ *is a tuple of five algorithms* $(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy)$.

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

$\mathcal{KG}(\mathsf{msk}, y) \rightarrow (\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk})$**:** *The key generation algorithm takes a master secret key* msk *and a string* $y \in \mathcal{Y}$*, and outputs a functional secret key* $\mathcal{sk}_y$*, a certificate verification key* vk*, and a testing key* tk.

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

$\mathcal{D}ec(\mathcal{sk}_y, \mathsf{ct}) \rightarrow z$**:** *The decryption algorithm takes a functional secret key* $\mathcal{sk}_y$ *and a ciphertext* ct*, and outputs a value* $z \in \mathcal{Z} \cup \{\bot\}$.

$\mathcal{V}rfy(\mathsf{vk}, \widetilde{\mathcal{sk}_y}) \rightarrow \top/\bot$**:** *The verification algorithm takes a verification key* vk *and a (possibly malformed) functional secret key* $\widetilde{\mathcal{sk}_y}$*, and outputs* $\top$ *or* $\bot$.

Decryption correctness: *For all* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$*, we have*

$$\Pr\left[\mathcal{D}ec(\mathcal{sk}_y, \mathsf{ct}) = F(x, y) \ : \ \begin{array}{c}\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, x)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Verification correctness: *We have*

$$\Pr\left[\mathcal{V}rfy(\mathsf{vk}, \mathcal{sk}_y) = \top \ : \ \begin{array}{c}\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 7.5 (Classical Decryption Property). *We say that* $\mathsf{SKFE\text{-}CR\text{-}SKL} = (\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy)$ *has the classical decryption property if there exists a deterministic polynomial time algorithm* CDec *such that given* $\mathcal{sk}_y$ *in the register* SK *and ciphertext* ct*,* $\mathcal{D}ec$ *applies the map* $|u\rangle_{\mathsf{SK}} |v\rangle_{\mathsf{OUT}} \rightarrow |u\rangle_{\mathsf{SK}} |v \oplus \mathsf{CDec}(u, \mathsf{ct})\rangle_{\mathsf{OUT}}$ *and outputs the measurement result of the register* OUT *in the computational basis, where* OUT *is initialized to* $|0 \cdots 0\rangle_{\mathsf{OUT}}$.

Definition 7.6 (Key-Testability). *We say that an SKFE-CR-SKL scheme* $\mathsf{SKFE\text{-}CR\text{-}SKL}$ *with the classical decryption property satisfies key testability, if there exists a classical deterministic algorithm* $\mathsf{KeyTest}$ *that satisfies the following conditions:*

- ***Syntax:*** $\mathsf{KeyTest}$ *takes as input a testing key* tk *and a classical string* sk *as input. It outputs* 0 *or* 1.
- ***Correctness:*** *Let* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y)$ *where* y *is a string. We denote the register holding* $\mathcal{sk}_y$ *as* SK*. Let* KT *be a register that is initialized to* $|0\rangle_{\mathsf{KT}}$*. If we apply the map* $|u\rangle_{\mathsf{SK}} |\beta\rangle_{\mathsf{KT}} \rightarrow |u\rangle_{\mathsf{SK}} |\beta \oplus \mathsf{KeyTest}(\mathsf{tk}, u)\rangle_{\mathsf{KT}}$ *to the registers* SK *and* KT *and then measure* KT *in the computational basis, we obtain* 1 *with overwhelming probability.*
- ***Security:*** *Consider the following experiment* $\mathsf{Exp}^{\mathsf{key\text{-}test}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda)$.
  1. *The challenger* $\mathcal{Ch}$ *runs* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *and initializes* $\mathcal{A}$ *with input* msk.
  2. $\mathcal{A}$ *can get access to the following oracle, where the list* $L_{\mathcal{KG}}$ *used by the oracles is initialized to an empty list.*

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

3. *$\mathcal{A}$ sends a tuple of classical strings (y, sk, x) to $\mathcal{C}h$. $\mathcal{C}h$ outputs $\bot$ if there is no entry of the form (y, tk) in $L_{\mathcal{KG}}$ for some tk. Otherwise, $\mathcal{C}h$ generates $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, x)$, and outputs $\top$ if $\mathsf{KeyTest}(\mathsf{tk}, \mathsf{sk}) = 1$ and $\mathsf{CDec}(\mathsf{sk}, \mathsf{ct}) \neq F(x, y)$, and outputs $\bot$ otherwise.*

For all QPT $\mathcal{A}$, the following must hold:

$$\mathsf{Adv}^{\mathsf{key\text{-}test}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda) := \Pr\Big[\mathsf{Exp}^{\mathsf{key\text{-}test}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda) \to \top\Big] \leq \mathsf{negl}(\lambda).$$

Definition 7.7 (Selective Single-Ciphertext KLA Security). *We say that an SKFE-CR-SKL scheme $\mathsf{SKFE\text{-}CR\text{-}SKL}$ is selective single-ciphertext secure, if it satisfies the following requirement, formalized from the experiment $\mathsf{Exp}^{\mathsf{sel\text{-}1ct\text{-}kla}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda, \mathsf{coin})$ between an adversary $\mathcal{A}$ and a challenger:*

1. *Initialized with 1^λ, $\mathcal{A}$ outputs (x_0^*, x_1^*). The challenger $\mathcal{C}h$ runs $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. *$\mathcal{A}$ can get access to the following (stateful) oracles, where the list $L_{\mathcal{KG}}$ used by the oracles is initialized to an empty list:*

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

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

3. *$\mathcal{A}$ requests the challenge ciphertext. If there exists an entry (y, vk, V) in $L_{\mathcal{KG}}$ such that $F(x_0^*, y) \neq F(x_1^*, y)$ and $V = \bot$, $\mathcal{C}h$ outputs 0 as the final output of this experiment. Otherwise, $\mathcal{C}h$ generates $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{msk}, x^*_{\mathsf{coin}})$ and sends ct^* to $\mathcal{A}$.*
4. *$\mathcal{A}$ continues to make queries to $O_{\mathcal{KG}}$. However, $\mathcal{A}$ is not allowed to send y such that $F(x_0^*, y) \neq F(x_1^*, y)$ to $O_{\mathcal{KG}}$.*
5. *$\mathcal{A}$ outputs a guess coin' for coin. $\mathcal{C}h$ outputs coin' as the final output of the experiment.*

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

$$\mathsf{Adv}^{\mathsf{sel\text{-}1ct\text{-}kla}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda) := \Big|\Pr\Big[\mathsf{Exp}^{\mathsf{sel\text{-}1ct\text{-}kla}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda, 0) \to 1\Big] - \Pr\Big[\mathsf{Exp}^{\mathsf{sel\text{-}1ct\text{-}kla}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda, 1) \to 1\Big]\Big| \leq \mathsf{negl}(\lambda).$$

In Appendix C, we prove the following theorem:

Theorem 7.8. *Assuming the existence of a BB84-based SKE-CD scheme and the existence of OWFs, there exists a selective single-ciphertext KLA secure SKFE-CR-SKL scheme satisfying the key-testability property.*

7.2 Definitions of ABE-CR-SKL

In this section, we recall definitions of ABE-CR-SKL by Agrawal et al. [AKN+23].

Definition 7.9 (ABE-CR-SKL). *An ABE-CR-SKL scheme $\mathsf{ABE\text{-}CR\text{-}SKL}$ is a tuple of five algorithms $(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy)$. Below, let $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_\lambda$, and $R = \{R_\lambda : \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}\}_\lambda$ be the ciphertext attribute space, the key attribute space, and the associated relation of $\mathsf{ABE\text{-}CR\text{-}SKL}$, respectively. Let $\mathcal{M}$ denote the message space.*

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

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

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

$\mathcal{Dec}(\mathcal{sk}_y, \mathsf{ct}) \to \mathsf{m}'$**:** *The decryption algorithm takes a user secret key* $\mathcal{sk}_y$ *and a ciphertext* ct*. It outputs a value* $\mathsf{m}' \in \{\bot\} \cup \mathcal{M}$*.*

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

Decryption correctness: *For every* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *satisfying* $R(x, y) = 0$ *and* $\mathsf{m} \in \mathcal{M}$*, we have*

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

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

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

Definition 7.10 (Adaptive IND-KLA Security). *We say that an ABE-CR-SKL scheme* $\mathsf{ABE\text{-}CR\text{-}SKL}$ *for relation* $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ *is adaptively IND-KLA secure, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ada\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{Ch}$*:*

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

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

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

2. *When* $\mathcal{A}$ *sends* $(x^*, \mathsf{m}_0, \mathsf{m}_1)$ *to* $\mathcal{Ch}$*, it checks that for every entry* (y, vk, V) *in* $L_{\mathcal{KG}}$ *such that* $R(x^*, y) = 0$*, it holds that* $V = \top$*. If so,* $\mathcal{Ch}$ *generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x^*, \mathsf{m}_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$*. Otherwise, it 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* $R(x^*, y) = 0$ *to* $O_{\mathcal{KG}}$*.*

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

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

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

Remark 7.11. Although we can handle the situation where multiple keys for the same attribute y are generated using an index management such as $(y, 1, vk_1, V_1)$, $(y, 2, vk_2, V_2)$, we use the simplified definition as Agrawal et al. [AKN+23] did.

We also consider relaxed versions of the above security notion.

Definition 7.12 (Selective IND-KLA Security). *We consider the same security game as that for adaptive IND-KLA security except that the adversary* $\mathcal{A}$ *should declare its target* x^* *at the beginning of the game (even before it is given* pk*).*

We then define the advantage $\mathsf{Adv}^{\mathsf{sel\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda)$ *for the selective case similarly. We say* $\mathsf{ABE\text{-}CR\text{-}SKL}$ *is selectively IND-KLA secure if for any QPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{sel\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}CR\text{-}SKL},\mathcal{A}}(1^\lambda)$ *is negligible.*

7.3 Construction

We construct an ABE-CR-SKL scheme $\mathsf{ABE\text{-}CR\text{-}SKL} = \mathsf{ABE\text{-}CR\text{-}SKL}.(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy)$ for the relation R with the message space $\mathcal{M} := \{0,1\}^{\ell_{\mathsf{m}}}$ using the following building blocks.

- ABE scheme $\mathsf{ABE} = \mathsf{ABE}.(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ for the following relation R'.

  $R'(x', y')$**:** Interpret $x' := x \| C$ and $y' := y \| z$, where C is a circuit. Then, output 0 if $R(x,y) = 0$ and $C(z) = \bot$, and otherwise output 1.

- Compute-and-Compare Obfuscation $\mathsf{CC.Obf}$ with the simulator $\mathsf{CC.Sim}$.

- SKFE-CR-SKL scheme with key-testability $\mathsf{SKFE\text{-}CR\text{-}SKL} = \mathsf{SKFE\text{-}CR\text{-}SKL}.(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy, \mathsf{KeyTest})$ for the following functionality F. It also has the classical decryption algorithm $\mathsf{SKFE\text{-}CR\text{-}SKL.CDec}$.

  $F(\widetilde{x}, \widetilde{y})$**:** Interpret $\widetilde{x} := x \| z$ and $\widetilde{y} := y$. Then, output z if $R(x,y) = 0$, and otherwise output $\bot$.

The construction is as follows.

$\mathsf{ABE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$**:**

- Generate $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$.
- Generate $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$.
- Output $\mathsf{pk} := \mathsf{abe.pk}$ and $\mathsf{msk} := (\mathsf{abe.msk}, \mathsf{skfe.msk})$.

$\mathsf{ABE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk}, y)$**:**

- Parse $\mathsf{msk} = (\mathsf{abe.msk}, \mathsf{skfe.msk})$.
- Generate $(\mathsf{skfe}.\mathpzc{sk}, \mathsf{skfe.vk}, \mathsf{skfe.tk}) \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{skfe.msk}, y)$. We denote the register holding $\mathsf{skfe}.\mathpzc{sk}$ as $\mathsf{SKFE.SK}$.
- Sample explicit randomness $\mathsf{k} \leftarrow \{0,1\}^\lambda$.
- Prepare a register $\mathsf{ABE.SK}$ that is initialized to $|0 \cdots 0\rangle_{\mathsf{ABE.SK}}$.
- Apply the map $|u\rangle_{\mathsf{SKFE.SK}} |v\rangle_{\mathsf{ABE.SK}} \rightarrow |u\rangle_{\mathsf{SKFE.SK}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, y \| u, \mathsf{k})\rangle_{\mathsf{ABE.SK}}$ to the registers $\mathsf{SKFE.SK}$ and $\mathsf{ABE.SK}$, and obtain $\mathpzc{sk}$ over the registers $\mathsf{SKFE.SK}$ and $\mathsf{ABE.SK}$.
- Output $\mathpzc{sk}$ and $\mathsf{vk} := (y, \mathsf{abe.msk}, \mathsf{skfe.vk}, \mathsf{skfe.tk}, \mathsf{k})$.

$\mathsf{ABE\text{-}CR\text{-}SKL.Enc}(\mathsf{pk}, x, \mathsf{m})$**:**

- Parse $\mathsf{pk} = \mathsf{abe.pk}$.
- Generate $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$, where pp_D consists of circuit parameters of D defined in the security proof.
- Generate $\mathsf{abe.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, x \| \widetilde{C}, \mathsf{m})$.
- Output $\mathsf{ct} := \mathsf{abe.ct}$.

$\mathsf{ABE\text{-}CR\text{-}SKL}.\mathcal{D}ec(\mathpzc{sk}, \mathsf{ct})$**:**

- Parse $\mathsf{ct} = \mathsf{abe.ct}$. We denote the register holding $\mathpzc{sk}$ as $\mathsf{SKFE.SK} \otimes \mathsf{ABE.SK}$.
- Prepare a register MSG that is initialized to $|0 \cdots 0\rangle_{\mathsf{MSG}}$
- Apply the map $|v\rangle_{\mathsf{ABE.SK}} |w\rangle_{\mathsf{MSG}} \rightarrow |v\rangle_{\mathsf{ABE.SK}} |w \oplus \mathsf{ABE.Dec}(v, \mathsf{abe.ct})\rangle_{\mathsf{MSG}}$ to the registers $\mathsf{ABE.SK}$ and MSG.
- Measure the register MSG in the computational basis and output the result m'.

$\mathsf{ABE\text{-}CR\text{-}SKL}.\mathcal{V}rfy(\mathsf{vk}, \mathpzc{sk}')$**:**

- Parse $\mathsf{vk} = (y, \mathsf{abe.msk}, \mathsf{skfe.vk}, \mathsf{skfe.tk}, \mathsf{k})$. We denote the register holding $\mathcal{sk}'$ as $\mathsf{SKFE.SK} \otimes \mathsf{ABE.SK}$.
- Prepare a register $\mathsf{SKFE.KT}$ that is initialized to $|0\rangle_{\mathsf{SKFE.KT}}$.
- Apply the map $|u\rangle_{\mathsf{SKFE.SK}} |\beta\rangle_{\mathsf{SKFE.KT}} \to |u\rangle_{\mathsf{SKFE.SK}} |\beta \oplus \mathsf{SKFE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{skfe.tk}, u)\rangle_{\mathsf{SKFE.KT}}$ to the registers $\mathsf{SKFE.SK}$ and $\mathsf{SKFE.KT}$.
- Measure $\mathsf{SKFE.KT}$ in the computational basis and output $\bot$ if the result is 0. Otherwise, go to the next step.
- Apply the map $|u\rangle_{\mathsf{SKFE.SK}} |v\rangle_{\mathsf{ABE.SK}} \to |u\rangle_{\mathsf{SKFE.SK}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, y\|u, \mathsf{k})\rangle_{\mathsf{ABE.SK}}$ to the registers $\mathsf{SKFE.SK}$ and $\mathsf{ABE.SK}$.
- Trace out the register $\mathsf{ABE.SK}$ and obtain $\mathsf{skfe}.\mathcal{sk}'$ over register $\mathsf{SKFE.SK}$.
- Output $\top$ if $\top = \mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{skfe.vk}, \mathsf{skfe}.\mathcal{sk}')$ and $\bot$ otherwise.

Decryption correctness: Let x and y be a ciphertext-attribute and a key-attribute, respectively, such that $R(x, y) = 0$. The secret-key $\mathcal{sk}$ output by $\mathsf{ABE\text{-}CR\text{-}SKL}.\mathcal{KG}$ is of the form $\sum_u \alpha_u |u\rangle_{\mathsf{SKFE.SK}} \left|\mathsf{abe.sk}_{y\|u}\right\rangle_{\mathsf{ABE.SK}}$, where $\mathsf{abe.sk}_{y\|u} \leftarrow \mathsf{ABE.KG}(\mathsf{abe.msk}, y\|u, \mathsf{k})$. Let $\mathsf{ct} \leftarrow \mathsf{ABE\text{-}CR\text{-}SKL.Enc}(\mathsf{pk}, x, \mathsf{m})$, where $\mathsf{ct} = \mathsf{abe.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, x\|\widetilde{C}, \mathsf{m})$ and $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$. If we apply the map

$$|v\rangle_{\mathsf{ABE.SK}} |w\rangle_{\mathsf{MSG}} \to |v\rangle_{\mathsf{ABE.SK}} |w \oplus \mathsf{ABE.Dec}(v, \mathsf{abe.ct})\rangle_{\mathsf{MSG}}$$

to $\sum_u \alpha_u |u\rangle_{\mathsf{SKFE.SK}} \left|\mathsf{abe.sk}_{y\|u}\right\rangle_{\mathsf{ABE.SK}} \otimes |0\cdots 0\rangle_{\mathsf{MSG}}$, the result is

$$\sum_u \alpha_u |u\rangle_{\mathsf{SKFE.SK}} \left|\mathsf{abe.sk}_{y\|u}\right\rangle_{\mathsf{ABE.SK}} \otimes |\mathsf{m}\rangle_{\mathsf{MSG}}$$

since $\widetilde{C}(u) = \bot$ and thus $R'(x\|\widetilde{C}, y\|u) = 0$ for every string u. Therefore, $\mathsf{ABE\text{-}CR\text{-}SKL}$ satisfies decryption correctness.

Verification correctness. Let y be a key attribute and $(\mathcal{sk}, \mathsf{vk}) \leftarrow \mathsf{ABE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk}, y)$. It is clear that the state $\mathsf{skfe}.\mathcal{sk}'$ obtained when computing $\mathsf{ABE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{vk}, \mathcal{sk})$ is the same as $\mathsf{skfe}.\mathcal{sk}$ generated when generating $\mathcal{sk}$. Therefore, the verification correctness of $\mathsf{ABE\text{-}CR\text{-}SKL}$ follows from that of $\mathsf{SKFE\text{-}CR\text{-}SKL}$.

7.4 Proof of Selective IND-KLA Security

Let $\mathcal{A}$ be an adversary for the selective IND-KLA security of $\mathsf{ABE\text{-}CR\text{-}SKL}$. We consider the following sequence of experiments.

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

1. $\mathcal{A}$ declares the challenge ciphertext attribute x^*. The challenger $\mathcal{Ch}$ generates $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$, and sends $\mathsf{pk} := \mathsf{abe.pk}$ to $\mathcal{A}$.
2. $\mathcal{A}$ can get access to the following (stateful) oracles, where the list $L_{\mathcal{KG}}$ used by the oracles is initialized to an empty list:

   $O_{\mathcal{KG}}(y)$**:** Given y, it finds an entry of the form (y, vk_y, V) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\bot$. Otherwise, it generates $\mathcal{sk}_y, \mathsf{vk}_y$ as follows.

   - Generate $(\mathsf{skfe}.\mathcal{sk}_y, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y) \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{skfe.msk}, y)$. We denote the register holding $\mathsf{skfe}.\mathcal{sk}_y$ as $\mathsf{SKFE.SK_y}$.
   - Prepare a register $\mathsf{ABE.SK_y}$ that is initialized to $|0\cdots 0\rangle_{\mathsf{ABE.SK_y}}$.
   - Sample explicit randomness $\mathsf{k}_y \leftarrow \{0,1\}^\lambda$.
   - Apply the map $|u\rangle_{\mathsf{SKFE.SK_y}} |v\rangle_{\mathsf{ABE.SK_y}} \to |u\rangle_{\mathsf{SKFE.SK_y}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, y\|u, \mathsf{k}_y)\rangle_{\mathsf{ABE.SK_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.SK_y}$, and obtain $\mathcal{sk}_y$ over the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.SK_y}$.

- Set $\mathsf{vk}_y := (y, \mathsf{abe.msk}, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y, \mathsf{k}_y)$.

It returns sk_y to $\mathcal{A}$ and adds the entry $(y, \mathsf{vk}_y, \bot)$ to $L_{\mathcal{KG}}$.

$O_{\mathcal{Vrfy}}(y, \widetilde{\mathit{sk}})$**:** It finds an entry (y, vk_y, V) from $L_{\mathcal{KG}}$. (If there is no such entry, it returns $\bot$.) It parses $\mathsf{vk}_y = (y, \mathsf{abe.msk}, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y, \mathsf{k}_y)$ and computes d as follows.

- Let the register holding $\widetilde{\mathit{sk}}$ be $\mathsf{SKFE.SK_y} \otimes \mathsf{ABE.SK_y}$.
- Prepare a register $\mathsf{SKE.KT_y}$ that is initialized to $|0\rangle_{\mathsf{SKE.KT_y}}$.
- Apply $|u\rangle_{\mathsf{SKFE.SK_y}} |\beta\rangle_{\mathsf{SKE.KT_y}} \rightarrow |u\rangle_{\mathsf{SKFE.SK_y}} |\beta \oplus \mathsf{SKFE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{skfe.tk}_y, u)\rangle_{\mathsf{SKE.KT_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{SKE.KT_y}$.
- Measure $\mathsf{SKE.KT_y}$ in the computational basis and set $d := \bot$ if the result is 0. Otherwise, go to the next step.
- Apply the map $|u\rangle_{\mathsf{SKFE.SK_y}} |v\rangle_{\mathsf{ABE.SK_y}} \rightarrow |u\rangle_{\mathsf{SKFE.SK_y}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, y\|u, \mathsf{k}_y)\rangle_{\mathsf{ABE.SK_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.SK_y}$.
- Trace out the register $\mathsf{ABE.SK_y}$ and obtain $\mathsf{skfe}.\mathit{sk}'$ over $\mathsf{SKFE.SK_y}$.
- Set $d := \top$ if $\top = \mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{skfe.vk}_y, \mathsf{skfe}.\mathit{sk}')$ and set $d := \bot$ otherwise. It returns d to $\mathcal{A}$. Finally, if $V = \bot$, it updates the entry (y, vk_y, V) to (y, vk_y, d).

3. $\mathcal{A}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ to $\mathcal{Ch}$. $\mathcal{Ch}$ checks if for every entry (y, vk_y, V) in $L_{\mathcal{KG}}$ such that $R(x^*, y) = 0$, it holds that $V = \top$. If so, it generates $\widetilde{C}^* \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$ and $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, x^*\|\widetilde{C}^*, \mathsf{m}^*_{\mathsf{coin}})$, and sends $\mathsf{ct}^* := \mathsf{abe.ct}^*$ to $\mathcal{A}$. Otherwise, it outputs 0.
4. $\mathcal{A}$ outputs coin'. The challenger outputs coin' as the final output of the experiment.

$\mathsf{Hyb}_1^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_0^{\mathsf{coin}}$ except that $\widetilde{C}^*$ is generated as $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{skfe.ct}^*], \mathsf{lock}, 0)$, where $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Enc}(\mathsf{skfe.msk}, x^*\|0^\lambda)$, $\mathsf{lock} \leftarrow \{0,1\}^\lambda$, and $D[\mathsf{skfe.ct}^*](x)$ is a circuit that has $\mathsf{skfe.ct}^*$ hardwired and outputs $\mathsf{SKFE\text{-}CR\text{-}SKL.CDec}(x, \mathsf{skfe.ct}^*)$.

We pick lock as a uniformly random string that is completely independent of other variables such as $\mathsf{skfe.ct}^*$. Thus, from the security of $\mathsf{CC.Obf}$, we have $\left|\Pr\left[\mathsf{Hyb}_0^{\mathsf{coin}} = 1\right] - \Pr\left[\mathsf{Hyb}_1^{\mathsf{coin}} = 1\right]\right| = \mathsf{negl}(\lambda)$.

$\mathsf{Hyb}_2^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_1^{\mathsf{coin}}$ except that $\mathsf{skfe.ct}^*$ hardwired into the circuit D is generated as $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Enc}(\mathsf{skfe.msk}, x^*\|\mathsf{lock})$.

From the selective single-ciphertext security of $\mathsf{SKFE\text{-}CR\text{-}SKL}$, we can show that $\mathsf{Hyb}_1^{\mathsf{coin}} \approx \mathsf{Hyb}_2^{\mathsf{coin}}$. Suppose that $\mathsf{Hyb}_1^{\mathsf{coin}} \not\approx \mathsf{Hyb}_2^{\mathsf{coin}}$ and $\mathcal{D}$ is a corresponding distinguisher. We consider the following reduction $\mathcal{R}$:

Execution of $\mathcal{R}^{\mathcal{D}}$ in $\mathsf{Exp}^{\mathsf{sel\text{-}1ct\text{-}kla}}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{R}}(1^\lambda, b)$:

1. $\mathcal{Ch}$ runs $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$ and initializes $\mathcal{R}$ with the security parameter 1^λ.
2. $\mathcal{D}$ declares the challenge ciphertext-attribute x^*. $\mathcal{R}$ generates $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and sends $\mathsf{ek} := \mathsf{abe.pk}$ to $\mathcal{D}$.
3. $\mathcal{R}$ chooses $\mathsf{lock} \leftarrow \{0,1\}^\lambda$ and sends $(\mathsf{skfe.m}_0^*, \mathsf{skfe.m}_1^*) := (x^*\|0^\lambda, x^*\|\mathsf{lock})$ to $\mathcal{Ch}$.
4. $\mathcal{R}$ simulates the access to $O_{\mathcal{KG}}(y)$ for $\mathcal{D}$ as follows, where $L_{\mathcal{KG}}$ is a list initialized to be empty:

   $O_{\mathcal{KG}}(y)$: Given y, it finds an entry of the form (y, vk_y, V) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\bot$. Otherwise, it generates $(\mathit{sk}_y, \mathsf{vk}_y)$ similar to $\mathsf{Hyb}_0^{\mathsf{coin}}$ except that the values $(\mathsf{skfe}.\mathit{sk}_y, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y)$ are generated as $(\mathsf{skfe}.\mathit{sk}_y, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y) \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL}.O_{\mathcal{KG}}(y)$ instead. It adds $(y, \mathsf{vk}_y, \bot)$ to $L_{\mathcal{KG}}$ and returns sk_y.
5. $\mathcal{R}$ simulates the access to $O_{\mathcal{Vrfy}}(y, \widetilde{\mathit{sk}})$ as follows:

$O_{\mathcal{Vrfy}}(y, \widetilde{sk})$: Given $(y, \widetilde{sk})$, it finds an entry (y, vk_y, V) from $L_{\mathcal{KG}}$ (If there is no such entry, it returns $\bot$). It then executes a procedure similar to that in $\mathsf{Hyb}_0^{\mathsf{coin}}$, except that $\mathsf{SKFE\text{-}CR\text{-}SKL}.O_{\mathcal{Vrfy}}(y, \mathsf{skfe}.\mathcal{sk}')$ is executed instead of $\mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{skfe.vk}_y, \mathsf{skfe}.\mathcal{sk}')$. The corresponding output d is returned as the output of the oracle.

6. $\mathcal{R}$ requests the challenge ciphertext from $\mathcal{Ch}$ and receives $\mathsf{skfe.ct}^* \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Enc}(\mathsf{skfe.msk}, \mathsf{skfe.m}_b^*)$.
7. $\mathcal{D}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ to $\mathcal{R}$. $\mathcal{R}$ checks that for every entry (y, vk_y, V) such that $R(x^*, y) = 0$, it holds that $V = \top$. If so, it generates $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{skfe.ct}^*], \mathsf{lock}, 0)$ and $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, x^* \| \widetilde{C}^*, \mathsf{m}_{\mathsf{coin}}^*)$ and sends $\mathsf{ct}^* := \mathsf{abe.ct}^*$ to $\mathcal{D}$. Else, it outputs 0.
8. $\mathcal{D}$ outputs a bit b'. $\mathcal{R}$ outputs b' and $\mathcal{Ch}$ outputs b' as the final output of the experiment.

It is easy to see that the view of $\mathcal{D}$ is the same as that in $\mathsf{Hyb}_2^{\mathsf{coin}}$ when lock is encrypted in $\mathsf{skfe.ct}^*$ and that of $\mathsf{Hyb}_1^{\mathsf{coin}}$ when 0^λ is encrypted. Moreover, for $\mathcal{D}$ to distinguish between the two hybrids, it must be the case that $V = \top$ for all entries (y, vk_y, V) such that $R(x^*, y) = 0$, which directly implies that the analogous values checked by $\mathsf{SKFE\text{-}CR\text{-}SKL}.O_{\mathcal{Vrfy}}$ must also be $\top$. If $R(x^*, y) = 1$, $F(x^* \| 0^\lambda, y) = F(x^* \| \mathsf{lock}, y) = \bot$. Consequently, $\mathcal{R}$ breaks the selective single-ciphertext security of SKFE-CR-SKL. Therefore, it must be that $\mathsf{Hyb}_1^{\mathsf{coin}} \approx \mathsf{Hyb}_2^{\mathsf{coin}}$.

$\mathsf{Hyb}_3^{\mathsf{coin}}$: This is the same as $\mathsf{Hyb}_2^{\mathsf{coin}}$ except that $O_{\mathcal{KG}}$ behaves as follows. (The difference is red colored.)

$O_{\mathcal{KG}}(y)$**:** Given y, it finds an entry of the form (y, vk_y, V) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\bot$. Otherwise, it generates $\mathcal{sk}_y, \mathsf{vk}_y$ as follows.

- Generate $(\mathsf{skfe}.\mathcal{sk}_y, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y) \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{skfe.msk}, y)$. We denote the register holding $\mathsf{skfe}.\mathcal{sk}_y$ as $\mathsf{SKFE.SK_y}$.
- Prepare a register $\mathsf{ABE.R'_y}$ that is initialized to $|0\rangle_{\mathsf{ABE.R'_y}}$.
- Apply the map $|u\rangle_{\mathsf{SKFE.SK_y}} |\beta\rangle_{\mathsf{ABE.R'_y}} \rightarrow |u\rangle_{\mathsf{SKFE.SK_y}} \left|\beta \oplus R'(x^* \| \widetilde{C}^*, y \| u)\right\rangle_{\mathsf{ABE.R'_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.R'_y}$.
- Measure $\mathsf{ABE.R'_y}$ in the computational basis and output $\bot$ if the result is 0. Otherwise, go to the next step.
- Prepare a register $\mathsf{ABE.SK_y}$ that is initialized to $|0 \cdots 0\rangle_{\mathsf{ABE.SK_y}}$.
- Sample explicit randomness $\mathsf{k}_y \leftarrow \{0,1\}^\lambda$.
- Apply the map $|u\rangle_{\mathsf{SKFE.SK_y}} |v\rangle_{\mathsf{ABE.SK_y}} \rightarrow |u\rangle_{\mathsf{SKFE.SK_y}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, y \| u, \mathsf{k}_y)\rangle_{\mathsf{ABE.SK_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.SK_y}$, and obtain $\mathcal{sk}_y$ over the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.SK_y}$.
- Set $\mathsf{vk}_y := (y, \mathsf{abe.msk}, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y, \mathsf{k}_y)$.

It returns $\mathcal{sk}_y$ to $\mathcal{A}$ and adds the entry $(y, \mathsf{vk}_y, \bot)$ to $L_{\mathcal{KG}}$.

With overwhelming probability, we observe that the added procedure that checks $R'(x^* \| \widetilde{C}^*, y \| u)$ in superposition affects the final state $\mathcal{sk}_y$ at most negligibly. We consider the following two cases.

- The first case is where $R(x^*, y) = 1$. In this case, we have $R'(x^* \| \widetilde{C}^*, y \| u) = 1$ for any u. Hence, we see that the added procedure does not affect the state $\mathsf{skfe}.\mathcal{sk}_y$.
- The second case is where $R(x^*, y) = 0$. Let $\mathsf{skfe}.\mathcal{sk}_y = \sum_u \alpha_u |u\rangle_{\mathsf{SKFE.SK_y}}$. From the classical decryption property of SKFE-CR-SKL, we have that for any u, it holds that $\mathsf{SKFE\text{-}CR\text{-}SKL.CDec}(u, \mathsf{skfe.ct}^*) = \mathsf{lock}$. Thus, $\widetilde{C}^*(u) \neq \bot$ and $R'(x^* \| \widetilde{C}^*, y \| u) = 1$ with overwhelming probability from the correctness of CC.Obf. Hence, we see that the added procedure affects the state $\mathsf{skfe.sk}_y$ at most negligibly.

Therefore, we have $\left|\Pr\left[\mathsf{Hyb}_2^{\mathsf{coin}} = 1\right] - \Pr\left[\mathsf{Hyb}_3^{\mathsf{coin}} = 1\right]\right| = \mathsf{negl}(\lambda)$.

$\mathsf{Hyb}_4^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_3^{\mathsf{coin}}$ except that the oracle $O_{\mathcal{Vrfy}}$ behaves as follows. (The difference is red colored.)

$O_{\mathcal{Vrfy}}(y, \widetilde{sk})$**:** It finds an entry (y, vk_y, V) from $L_{\mathcal{KG}}$. (If there is no such entry, it returns $\bot$.) It parses $\mathsf{vk}_y = (y, \mathsf{abe.msk}, \mathsf{skfe.vk}_y, \mathsf{skfe.tk}_y, \mathsf{k}_y)$. If $R(x^*, y) = 1$, then it behaves exactly as in $\mathsf{Hyb}_3^{\mathsf{coin}}$ and otherwise, it computes d as follows:

(a) Let the register holding $\widetilde{sk}$ be $\mathsf{SKFE.SK_y} \otimes \mathsf{ABE.SK_y}$.

(b) Prepare a register $\mathsf{SKFE.KT_y}$ that is initialized to $|0\rangle_{\mathsf{SKFE.KT_y}}$.

(c) Apply $|u\rangle_{\mathsf{SKFE.SK_y}} |\beta\rangle_{\mathsf{SKFE.KT_y}} \rightarrow |u\rangle_{\mathsf{SKFE.SK_y}} |\beta \oplus \mathsf{SKFE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{skfe.tk}_y, u)\rangle_{\mathsf{SKFE.KT_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{SKFE.KT_y}$.

(d) Measure $\mathsf{SKFE.KT_y}$ in the computational basis and set $d := \bot$ if the result is 0. Otherwise, go to the next step.

(e) Prepare a register $\mathsf{ABE.R'_y}$ that is initialized to $|0\rangle_{\mathsf{ABE.R'_y}}$.

(f) Apply the map $|u\rangle_{\mathsf{SKFE.SK_y}} |\beta\rangle_{\mathsf{ABE.R'_y}} \rightarrow |u\rangle_{\mathsf{SKFE.SK_y}} \left|\beta \oplus R'(x^* \| \widetilde{C}^*, y \| u)\right\rangle_{\mathsf{ABE.R'_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.R'_y}$.

(g) Measure $\mathsf{ABE.R'_y}$ in the computational basis and set $d := \bot$ if the result is 0. Otherwise, go to the next step.

(h) Apply the map $|u\rangle_{\mathsf{SKFE.SK_y}} |v\rangle_{\mathsf{ABE.SK_y}} \rightarrow |u\rangle_{\mathsf{SKFE.SK_y}} |v \oplus \mathsf{ABE.KG}(\mathsf{abe.msk}, y \| u, \mathsf{k})\rangle_{\mathsf{ABE.SK_y}}$ to the registers $\mathsf{SKFE.SK_y}$ and $\mathsf{ABE.SK_y}$.

(i) Trace out the register $\mathsf{ABE.SK_y}$ and obtain $\mathsf{skfe}.\widetilde{sk}'$ over $\mathsf{SKFE.SK_y}$.

(j) Set $d := \top$ if $\top = \mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{skfe.vk}_y, \mathsf{skfe}.\widetilde{sk}')$ and set $d := \bot$ otherwise. It returns d to $\mathcal{A}$. Finally, if $V = \bot$, it updates the entry (y, vk_y, V) into (y, vk_y, d).

Suppose there exists a QPT distinguisher $\mathcal{D}$ that has non-negligible advantage in distinguishing $\mathsf{Hyb}_3^{\mathsf{coin}}$ and $\mathsf{Hyb}_4^{\mathsf{coin}}$. Let $\mathcal{D}$ make $q = \mathrm{poly}(\lambda)$ many queries to the oracle $O_{\mathcal{Vrfy}}(\cdot, \cdot)$. We will now consider the following QPT algorithm $\mathcal{A}_{\mathsf{o2h}}$ with access to an oracle $\widetilde{O_{\mathcal{KG}}}$ and an oracle $\mathcal{O}$ that runs $\mathcal{D}$ as follows:

$\underline{\mathcal{A}_{\mathsf{o2h}}^{\widetilde{O_{\mathcal{KG}}}, \mathcal{O}}(\mathsf{abe.pk}, \mathsf{skfe.msk})}$:

1. $\mathcal{A}_{\mathsf{o2h}}$ runs $\mathcal{D}$ who sends the challenge ciphertext-attribute x^* to $\mathcal{A}_{\mathsf{o2h}}$.
2. $\mathcal{A}_{\mathsf{o2h}}$ sends $\mathsf{ek} = \mathsf{abe.pk}$ to $\mathcal{D}$ and initializes $L_{\mathcal{KG}}$ to be an empty list.
3. When $\mathcal{D}$ queries $O_{\mathcal{KG}}$ on input y, $\mathcal{A}_{\mathsf{o2h}}$ queries the oracle $\widetilde{O_{\mathcal{KG}}}(y)$ in order to obtain its response $|\phi\rangle$ and a list $\widetilde{L_{\mathcal{KG}}}$. It updates $L_{\mathcal{KG}} = \widetilde{L_{\mathcal{KG}}}$. It sends $|\phi\rangle$ to $\mathcal{D}$.
4. $\mathcal{A}_{\mathsf{o2h}}$ simulates the access of $O_{\mathcal{Vrfy}}$ for $\mathcal{D}$ as follows:

   $\underline{O_{\mathcal{Vrfy}}(y, \widetilde{sk})}$:

   (a) Execute Steps (a)-(e) of $O_{\mathcal{Vrfy}}$ as in $\mathsf{Hyb}_4^{\mathsf{coin}}$.

   (b) Apply the map $|u\rangle_{\mathsf{SKE.SK_y}} |\beta\rangle_{\mathsf{ABE.R_y}} \rightarrow |u\rangle_{\mathsf{SKE.SK_y}} |\beta \oplus \mathcal{O}(u)\rangle_{\mathsf{ABE.R_y}}$ to the registers $\mathsf{SKE.SK_y}$ and $\mathsf{ABE.R_y}$.

   (c) Execute Steps (g)-(j) of $O_{\mathcal{Vrfy}}$ as in $\mathsf{Hyb}_4^{\mathsf{coin}}$.
5. $\mathcal{D}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ to $\mathcal{A}_{\mathsf{o2h}}$. $\mathcal{A}_{\mathsf{o2h}}$ checks if for every entry (y, vk_y, V) in $L_{\mathcal{KG}}$ such that $R(x^*, y) = 0$, it holds that $V = \top$. If so, it generates $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{skfe.ct}^*], \mathsf{lock}, 0)$ and $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, x^* \| \widetilde{C}^*, \mathsf{m}_{\mathsf{coin}}^*)$, where $\mathsf{skfe.ct}^* = \mathsf{SKFE\text{-}CR\text{-}SKL.Enc}(\mathsf{skfe.msk}, x^* \| 0^\lambda)$and sends $\mathsf{ct}^* := \mathsf{abe.ct}^*$ to $\mathcal{D}$. Otherwise, it outputs 0.
6. $\mathcal{D}$ outputs a guess b'. $\mathcal{A}_{\mathsf{o2h}}$ outputs b'.

Let H be an oracle that for every input u, outputs 1. Consider now the extractor $\mathcal{B}_{\mathsf{o2h}}^{\widetilde{O_{\mathcal{KG}}},H}$ as specified by the O2H Lemma (Lemma 3.2). We will now construct a reduction $\mathcal{R}$ that runs $\mathcal{B}_{\mathsf{o2h}}$ by simulating the oracles $\widetilde{O_{\mathcal{KG}}}$ and H for $\mathcal{B}_{\mathsf{o2h}}$, and breaks key-testability of the SKFE-CR-SKL scheme.

Execution of $\mathcal{R}$ in $\mathsf{Exp}_{\mathsf{SKFE\text{-}CR\text{-}SKL},\mathcal{R}}^{\mathsf{key\text{-}test}}(1^\lambda)$:

1. The challenger $\mathcal{C}h$ runs $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$ and initializes $\mathcal{R}$ with input $\mathsf{skfe.msk}$.
2. $\mathcal{R}$ samples $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and initializes $\mathcal{B}_{\mathsf{o2h}}$ with the input $(\mathsf{abe.pk}, \mathsf{skfe.msk})$.
3. $\mathcal{R}$ simulates the access to $\widetilde{O_{\mathcal{KG}}}(y)$ for $\mathcal{B}$ as follows, where $L_{\mathcal{KG}}$ is a list initialized to be empty:

   $\widetilde{O_{\mathcal{KG}}}(y)$: Given y, it finds an entry of the form (y, vk_y, V) from $L_{\mathcal{KG}}$ (If there is such an entry, it returns $(\bot, L_{\mathcal{KG}})$). Otherwise, it generates (sk_y, vk_y) similar to $\mathsf{Hyb}_3^{\mathsf{coin}}$. It adds $(y, \mathsf{vk}_y, \bot)$ to $L_{\mathcal{KG}}$ and returns $(sk_y, L_{\mathcal{KG}})$.
4. When $\mathcal{B}_{\mathsf{o2h}}$ queries an input u to H, $\mathcal{R}$ responds with 1.
5. $\mathcal{B}_{\mathsf{o2h}}$ outputs values y and sk. $\mathcal{R}$ sends $(\mathsf{sk}, y, \mathsf{lock})$ to $\mathcal{C}h$.

We will now claim that with non-negligible probability, $\mathcal{R}$ obtains values sk and y such that $R'(x^* \| \widetilde{C}^*, y \| \mathsf{sk}) = 0$. Recall that the hybrids differ only when $R(x^*, y) = 0$. By the definition of R' and $\widetilde{C}^*$ and the decryption correctness of SKFE-CR-SKL, this will imply that $\mathsf{SKFE\text{-}CR\text{-}SKL.CDec}(\mathsf{sk}, \mathsf{skfe.ct}^\star) \neq F(x^* \| \mathsf{lock}, y)$ since $F(x^* \| \mathsf{lock}, y) = \mathsf{lock}$, and $R'(x^* \| \widetilde{C}^*, y \| \mathsf{sk}) = 0$ requires $\widetilde{C}^*(\mathsf{sk}) = \bot$. Moreover, $\mathsf{KeyTest}(\mathsf{skfe.tk}_y, \mathsf{sk})$ also holds. Consequently, this will imply $\mathcal{R}$ breaks the key-testability of SKFE-CR-SKL. To prove this, we will rely on the One-Way to Hiding (O2H) Lemma (Lemma 3.2). Consider an oracle G which takes as input u and outputs $R'(x^* \| \widetilde{C}^\star, y \| u)$ and an oracle H which takes as input u and outputs 1. Notice that if the oracle $\mathcal{O} = G$, then the view of $\mathcal{D}$ as run by $\mathcal{A}_{\mathsf{o2h}}$ is the same as in $\mathsf{Hyb}_4^{\mathsf{coin}}$, while if $\mathcal{O} = H$, the view of $\mathcal{D}$ is the same as in $\mathsf{Hyb}_3^{\mathsf{coin}}$. By the O2H Lemma, we have the following, where $z = (\mathsf{abe.pk}, \mathsf{skfe.msk})$.

$$\left|\Pr\left[\mathcal{A}_{\mathsf{o2h}}^{\widetilde{O_{\mathcal{KG}}},H}(z) = 1\right] - \Pr\left[\mathcal{A}_{\mathsf{o2h}}^{\widetilde{O_{\mathcal{KG}}},G}(z) = 1\right]\right| \leq 2q \cdot \sqrt{\Pr\left[\mathcal{B}_{\mathsf{o2h}}^{\widetilde{O_{\mathcal{KG}}},H}(z) \in S\right]} \ .$$

where S is a set where the oracles H and G differ, which happens only for inputs u s.t. $R'(x^* \| \widetilde{C}^\star, y \| u) = 0$. Since $\mathcal{R}$ obtains sk and y as the output of $\mathcal{B}_{\mathsf{o2h}}^{\widetilde{O_{\mathcal{KG}}},H}(z)$, the argument follows that $\mathsf{Hyb}_3^{\mathsf{coin}} \approx \mathsf{Hyb}_4^{\mathsf{coin}}$.

$\mathsf{Hyb}_5^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_4^{\mathsf{coin}}$ except that $\mathsf{ct}^* := \mathsf{abe.ct}^*$ is generated as $\mathsf{abe.ct}^* \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.pk}, x^* \| \widetilde{C}^*, 0^{\ell_{\mathsf{m}}})$.

The view of $\mathcal{A}$ in $\mathsf{Hyb}_4^{\mathsf{coin}}$ and $\mathsf{Hyb}_5^{\mathsf{coin}}$ can be simulated with $\mathsf{abe.pk}$ and the access to the quantum key generation oracle O_{qkg}. This is because before $\mathsf{ABE.KG}$ is required to be applied in the simulation of oracles $O_{\mathcal{KG}}$ and $O_{\mathcal{V}rfy}$, the relation check $R'(x^* \| \widetilde{C}^\star, y \| u)$ is already applied in superposition. Thus, we have $\left|\Pr[\mathsf{Hyb}_4^{\mathsf{coin}} = 1] - \Pr[\mathsf{Hyb}_5^{\mathsf{coin}} = 1]\right| = \mathsf{negl}(\lambda)$.

Lastly, Hyb_5^0 and Hyb_5^1 are exactly the same experiment and thus we have $\left|\Pr\left[\mathsf{Hyb}_5^0 = 1\right] - \Pr\left[\mathsf{Hyb}_5^1 = 1\right]\right| = \mathsf{negl}(\lambda)$. Then, from the above arguments, we obtain

$$\begin{aligned}&\left|\Pr\left[\mathsf{Exp}_{\mathsf{ABE\text{-}CR\text{-}SKL},\mathcal{A}}^{\mathsf{sel\text{-}ind\text{-}kla}}(1^\lambda, 0) = 1\right] - \mathsf{Exp}_{\mathsf{ABE\text{-}CR\text{-}SKL},\mathcal{A}}^{\mathsf{sel\text{-}ind\text{-}kla}}(1^\lambda, 1) = 1]\right| \\ =&\left|\Pr\left[\mathsf{Hyb}_0^0 = 1\right] - \Pr\left[\mathsf{Hyb}_0^1 = 1\right]\right| \leq \mathsf{negl}(\lambda).\end{aligned}$$

This completes the proof. ☐

Recall now that SKFE-CR-SKL with Key-Testability is implied by BB84-based SKE-CD, OWFs and adaptively single-ciphertext function-private SKFE. Since this notion of SKFE is implied by OWFs (Theorem 7.3), we have that SKFE-CR-SKL with Key-Testability is implied by LWE. Since, Compute-and-Compare Obfuscation and quantum-secure Ciphertext-Policy ABE for Circuits are both implied by LWE, this gives us the following theorem:

Theorem 7.13. *There exists an ABE-CR-SKL scheme satisfying Selective IND-KLA security, assuming the polynomial hardness of the LWE assumption.*

8 ABE-CR^2-SKL from Multi-Input ABE

In this section, we show how to achieve ABE-CR-SKL with classical certificates (ABE-CR^2-SKL) from MI-ABE.

8.1 Definitions of Multi-Input ABE

First, we recall the definitions of MI-ABE. The syntax of MI-ABE is as follows:

Definition 8.1. *A Multi-Input (Ciphertext-Policy) Attribute-Based Encryption scheme* $\mathsf{MI\text{-}ABE}$ *is a tuple of four PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*. Let* $k = \mathrm{poly}(\lambda)$ *and let* $\{\mathcal{X}_\lambda\}_\lambda$*,* $\{(\mathcal{Y}_\lambda)^k\}_\lambda$ *and* $R = \{R_\lambda : \mathcal{X}_\lambda \times (\mathcal{Y}_\lambda)^k \to \{0,1\}\}$ *be the ciphertext attribute space, key attribute space and the relation associated with* $\mathsf{MI\text{-}ABE}$ *respectively. Let* $\{0,1\}^\ell$ *be the message space. Let* $s(\lambda)$ *denote the maximum bit string length required to describe PPT circuits with input size* $k \cdot n(\lambda)$ *and depth* $d(\lambda)$ *for polynomials* s, n *and* d*. Let* $\mathcal{X}_\lambda = \{0,1\}^{s(\lambda)}$ *and* $\mathcal{Y} = \{0,1\}^{n(\lambda)}$*. Let* R *be such that* $R(x, y_1, \ldots, y_k) = 0 \iff x(y_1, \ldots, y_k) = \perp$ *where* $x \in \mathcal{X}$ *is parsed as a* k*-input circuit and* $(y_1, \ldots, y_k) \in \mathcal{Y}^k$*.*

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

$\mathsf{KG}(\mathsf{msk}, i, y_i) \to \mathsf{sk}_i$**:** *The key generation algorithm takes the master secret key* msk*, an index* $i \in [k]$*, and a key-attribute* $y_i \in \mathcal{Y}$ *as input. It outputs a secret-key* sk_i*.*

$\mathsf{Enc}(\mathsf{pk}, x, \mathsf{m}) \to \mathsf{ct}$**:** *The encryption algorithm takes the public key* pk*, a ciphertext attribute* $x \in \mathcal{X}$*, and a message* $\mathsf{m} \in \{0,1\}^\ell$ *as input. It outputs a ciphertext* ct*.*

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}_1, \ldots \mathsf{sk}_k) \to \widetilde{\mathsf{m}}$**:** *The decryption algorithm takes as input a ciphertext* ct *and* k *secret-keys* $\mathsf{sk}_1, \ldots, \mathsf{sk}_k$*. It outputs a value* $\widetilde{\mathsf{m}} \in \{0,1\}^\ell \cup \perp$*.*

Correctness: *We require that*

$$\Pr\left[\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}_1, \ldots, \mathsf{sk}_k) = \mathsf{m} \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \forall i \in [k] : \mathsf{sk}_i \leftarrow \mathsf{KG}(\mathsf{msk}, i, y_i), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x, \mathsf{m}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

holds for all $x \in \mathcal{X}$ *and* $(y_1, \ldots, y_k) \in \mathcal{Y}^k$ *such that* $R(x, y_1, \ldots, y_k) = 0$ *and* $\mathsf{m} \in \{0,1\}^\ell$*.*

Remark 8.2. We say that an MI-ABE scheme has *polynomial-arity*, if it allows k to be an arbitrary polynomial in λ.

Definition 8.3 (Post-Quantum Selective Security for MI-ABE:). *We say that* $\mathsf{MI\text{-}ABE}$ *is a selective-secure MI-ABE scheme for relation* $R : \mathcal{X} \times \mathcal{Y}^k \to \{0,1\}$*, if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{sel\text{-}ind}}_{\mathsf{MI\text{-}ABE},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and a challenger* Ch*:*

1. *$\mathcal{A}$ declares the challenge ciphertext attribute x.* Ch *runs* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$*.*
2. *$\mathcal{A}$ can get access to the following key generation oracle.*

   $O_{\mathsf{kg}}(i, y_i)$**:** *It outputs* $\mathsf{sk}_i \leftarrow \mathsf{KG}(\mathsf{msk}, i, y_i)$ *to* $\mathcal{A}$*.*
3. *At some point,* $\mathcal{A}$ *sends* $(\mathsf{m}_0, \mathsf{m}_1)$ *to* Ch*. Then,* Ch *generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x, \mathsf{m}_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$*.*
4. *Again,* $\mathcal{A}$ *can get access to the oracle* O_{kg}*.*
5. *$\mathcal{A}$ outputs a guess* coin' *for* coin *and the experiment outputs* coin'*.*

For an adversary to be admissible, we require that for all tuples $(y_1, \ldots, y_k) \in \mathcal{Y}^k$ *received by* $\mathcal{A}$*, it must hold that* $R(x, y_1, \ldots, y_k) = 1$ *(non-decrytable). Given this constraint, we say* MI-ABE *satisfies selective security if for all QPT* $\mathcal{A}$*, the following holds:*

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

Remark 8.4. We do not require quantum security in Definition 8.3 unlike Definition 3.5.

Comparison with the Definition of Agrawal et. al [ARYY23]. Their definition considers a key-policy variant of MI-ABE. It consists of algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KG}_1, \cdots, \mathsf{KG}_{k-1}, \mathsf{KG}_k, \mathsf{Dec})$ where Enc works in a similar way as our definition, i.e., $\mathsf{Enc}(\mathsf{pk}, x_0, \mathsf{m}) \to \mathsf{ct}$ for message m and attribute x_0. For each $i \in [k-1]$, KG_i works as $\mathsf{KG}_i(\mathsf{msk}, x_i) \to \mathsf{sk}_i$ for the i-th key-attribute x_i. On the other hand, KG_k works as $\mathsf{KG}_k(\mathsf{msk}, f) \to \mathsf{sk}_f$ where f is an arbitrary k-input function. The guarantee is that decryption is feasible if and only if $f(x_0, \ldots, x_{k-1}) = \perp$. The algorithm Dec has the syntax $\mathsf{Dec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_1, \ldots, \mathsf{sk}_{k-1}, \mathsf{sk}_f) \to \mathsf{m}'$.

It is easy to see that this definition directly implies a $(k-1)$-input ciphertext-policy MI-ABE with algorithms $(\mathsf{Setup}', \mathsf{Enc}', \mathsf{KG}', \mathsf{Dec}')$. Specifically, $\mathsf{Setup}'(1^\lambda) := \mathsf{Setup}(1^\lambda)$, $\mathsf{Enc}'(\mathsf{pk}, x_0, \mathsf{m}) := \big(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, x_0, \mathsf{m})\big)$ and KG' can be defined as $\mathsf{KG}'(\mathsf{msk}, i, x_i) := \mathsf{KG}_i(\mathsf{msk}, x_i)$ for all $i \in [k-1]$. Finally, Dec' is defined as $\mathsf{Dec}'(\mathsf{ct} = (\mathsf{pk}, \widetilde{\mathsf{ct}}), \mathsf{sk}_1, \ldots, \mathsf{sk}_{k-1}) := \mathsf{Dec}(\mathsf{pk}, \widetilde{\mathsf{ct}}, \mathsf{sk}_1, \ldots, \mathsf{sk}_{k-1}, \mathsf{sk}_f)$ for the function $f(x_0, \ldots, x_{k-1}) = C_{x_0}(x_1, \ldots, x_{k-1})$ where C_{x_0} is the circuit described by ciphertext-attribute x_0.

Unfortunately, [ARYY23] only allows for a constant k, and hence we do not currently know of a construction where $k = \mathrm{poly}(\lambda)$. In the recent work of [AKY24], a variant of MI-ABE (for $k = \mathrm{poly}(\lambda)$) was achieved from the LWE and evasive LWE assumptions, which is non-standard. However, the scheme it implies would require the encryption algorithm Enc' to utilize the master secret-key msk.

Disucssion on MI-ABE and IO. We focus on MI-ABE satisfying Definition 8.1 in this paragraph. MI-ABE is not stronger than IO since IO is equivalent to multi-input functional encryption (MIFE) [GGG+14] and MIFE trivially implies MI-ABE. Although we do not know whether MIFE is separated from MI-ABE, it is likely since PKFE is separated from ABE [GMM17] and constructing FE schemes is significantly more challenging than ABE schemes. Moreover, MI-ABE is not currently known from weaker assumptions than IO. They are qualitatively different primitives, and MI-ABE could be constructed from weaker assumptions in the future. We also know that MI-ABE implies witness encryption [BJK+18], and witness encryption is achieved from the evasive LWE assumption [Tsa22, VWW22]. Hence, MI-ABE is somewhere between witness encryption and IO. Achieving MI-ABE from lattice assumptions is an interesting open problem.

8.2 Definitions of ABE-CR^2-SKL

The syntax of ABE-CR^2-SKL is defined as follows.

Definition 8.5 (ABE-CR^2-SKL). *An ABE-CR^2-SKL scheme* ABE-CR^2-SKL *is a tuple of six algorithms* $(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$. *Below, let* $\mathcal{M}$ *be the message space of* ABE-CR^2-SKL, $\mathcal{X}$ *be the ciphertext-attribute space, and* $\mathcal{Y}$ *the key-attribute space.*

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

$\mathcal{KG}(\mathsf{msk}, y) \to (\mathcal{sk}, \mathsf{vk})$**:** *The key generation algorithm takes the master secret-key* msk *and a key-attribute* $y \in \mathcal{Y}$ *as inputs, and outputs a decryption key* $\mathcal{sk}$ *and a verification key* vk.

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

$\mathcal{Dec}(\mathcal{sk}, \mathsf{ct}) \to \widetilde{\mathsf{m}}$**:** *The decryption algorithm takes a decryption key* $\mathcal{sk}$ *and a ciphertext* ct*, and outputs a value* $\widetilde{\mathsf{m}}$ *or* $\perp$.

$\mathcal{D}el(s\!k) \to \mathsf{cert}$**:** *The deletion algorithm takes a decryption key* $s\!k$ *and outputs a deletion certificate* cert.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}') \to \top/\bot$**:** *The verification algorithm takes a verification key* vk *and a certificate* cert', *and outputs* $\top$ *or* $\bot$.

Decryption correctness: *For every* $\mathsf{m} \in \mathcal{M}$, $\widetilde{x} \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *such that* $\widetilde{x}(y) = 0$ *(decryptable), we have*

$$\Pr\left[\mathcal{D}ec(s\!k, \mathsf{ct}) = \mathsf{m} \ : \ \begin{array}{c}(\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (s\!k, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \widetilde{x}, \mathsf{m})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

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

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) = \top \ : \ \begin{array}{c}(\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (s\!k, \mathsf{vk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y) \\ \mathsf{cert} \leftarrow \mathcal{D}el(s\!k)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Remark 8.6. We use the same experiment identifier sel-ind-kla to refer to the selective IND-KLA security experiments of ABE-CR-SKL and ABE-CR^2-SKL. This is for the sake of simplicity, as the exact experiment will be clear from the context.

Selective IND-KLA Security: *This security notion for an ABE-CR^2-SKL scheme is defined in the same way as for ABE-CR-SKL, except that instead of access to the oracle* $O_{\mathcal{V}rfy}$ *in the experiment,* $\mathcal{A}$ *receives access to the following oracle:*

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

8.3 Construction of ABE-CR^2-SKL from MI-ABE

Construction overview. We first provide the main idea behind our ABE-CR^2-SKL based on MI-ABE, which follows along the lines of our ABE-CR-SKL construction with some key-differences. In this case, we directly rely on a BB84-based SKE-CD scheme as a building block, instead of needing something similar to SKFE-CR-SKL. Consider now an SKE-CD ciphertext $\mathsf{skecd}.ct$ of the plaintext 0^λ. Let its corresponding verification-key $\mathsf{skecd.vk}$ be of the form $\mathsf{skecd.vk} = (x, \theta)$, where x and θ are k-bit strings. Then, $\mathsf{skecd}.ct$ is of the form $|\psi_1\rangle \otimes \cdots \otimes |\psi_k\rangle$, where for $i \in [k]$, $|\psi_i\rangle$ is of the following form:

$$|\psi_i\rangle = \begin{cases} |x[i]\rangle & if \ \ \theta[i] = 0 \\ |0\rangle + (-1)^{x[i]}\,|1\rangle & if \ \ \theta[i] = 1 \end{cases}$$

Now, for each $i \in \{2, \ldots, k\}$ and $u \in \{0, 1\}$, consider the attribute-key $\mathsf{abe.sk}_{i,u}$ generated by the MI-ABE key-generation algorithm for slot i and attribute $t\|u$. Here, t is a value chosen at random and is common for all slots corresponding to a given decryption-key $s\!k$. Additionally, for $u \in \{0, 1\}$, consider the keys $\mathsf{abe.sk}_{1,u}$ corresponding to the attributes $t\|u\|y$ where y is the actual ABE key-attribute.

Consider now the following state ρ_i for each $i \in [k]$:

$$\rho_i = \begin{cases} |x[i]\rangle \left|\mathsf{abe.sk}_{i,x[i]}\right\rangle & if \ \ \theta[i] = 0 \\ |0\rangle\,|\mathsf{abe.sk}_{i,0}\rangle + (-1)^{x[i]}\,|1\rangle\,|\mathsf{abe.sk}_{i,1}\rangle & if \ \ \theta[i] = 1, \end{cases}$$

The quantum decryption-key of our scheme will be the tuple $s\!k = (\rho_i)_{i \in [k]}$. The encryption algorithm is similar to the ABE-CR-SKL scheme, and outputs an MI-ABE encryption of message m under a policy $C_{\mathsf{ABE}}\|\widetilde{C}$

where $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda)$ is a simulator of a compute and compare obfuscator and C_{ABE} is the actual ABE policy. However, the MI-ABE relation is a little different. The relation is such that for a set of k attributes $y_1 = t_1 \| u_1 \| y$ and $y_2 = t_2 \| u_2, \ldots, y_k = t_k \| u_k$, it outputs 0 (decryptable) whenever $C_{\mathsf{ABE}}(y) = 0$ (the ABE relation is satisfied) AND $t_1 = \ldots = t_k$ AND $\widetilde{C}(u_1 \| \ldots \| u_k) = \bot$. Otherwise, it outputs 1. As in ABE-CR-SKL, the idea is that $\widetilde{C}$ is indistinguishable from $\widetilde{C}^*$ that is an obfuscation of the compute-and-compare circuit $\mathbf{CC}[D[\mathsf{lock} \oplus r, \mathsf{skecd.sk}], \mathsf{lock}, 0]$. Here, lock and r are random values, and $\mathsf{skecd.sk}$ is the secret-key of the SKE-CD scheme. The circuit D is such that $D(u)$ outputs $\mathsf{lock} \oplus r \oplus \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u)$, where $\mathsf{SKECD.CDec}$ is the classical decryption algorithm of the SKE-CD scheme.

Consequently, when every $\mathcal{sk} = (\rho_i)_{i \in [k]}$ the adversary receives is generated using an SKE-CD encryption of r (instead of 0^λ as in the scheme), then any tuple of k MI-ABE keys of the adversary having attributes $(t_1 \| u_1 \| y, t_2 \| u_2 \ldots, t_k \| u_k)$ satisfies one of the three conditions:

- The values $t_1 \ldots t_k$ are not all the same.
- $C_{\mathsf{ABE}}(y) \neq 0$.
- For $u = u_1 \| \ldots \| u_k$, $D[\mathsf{lock} \oplus r, \mathsf{skecd.sk}](u)$ returns lock.

Notice that the former condition ensures that the adversary cannot interleave keys corresponding to different decryption-keys. The last condition holds because the adversary never receives $\mathsf{abe.sk}_{i,1-x[i]}$ for any i such that $\theta[i] = 0$. Consequently, u and x are the same at all positions where $\theta[i] = 0$. This means that $\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u)$ outputs r by the classical decryption property of SKECD (Definition 3.12). It is important to note that the positions i where $\theta[i] = 1$ (the Hadamard positions) have no effect on the value output by $\mathsf{SKECD.CDec}$ as their purpose is just in the verification of deletion. As a result, the security of MI-ABE allows to simulate the adversary's view in this hybrid, as no "decryptable" set of k keys is given out.

Importantly, the switch from SKE-CD encryptions of 0^λ to r is indistinguishable, given that the adversary deletes all the information in the SKE-CD ciphertexts. To enforce this, the deletion algorithm requires the adversary to measure both the SKE-CD and MI-ABE registers for each slot to obtain values $(c_i, d_i)_{i \in [k]}$. Then, given the values $\{\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1}\}_{i \in [k]}$ as part of the verification key, the verification checks whether $x[i] = c_i \oplus d_i \cdot (\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1})$ holds for every $i \in [k]$ such that $\theta[i] = 1$. As a result, we are able use a standard hybrid argument to turn any distinguisher (of the 0^λ and r hybrids) into an attack on the certified deletion security of the SKE-CD scheme.

Construction. We will construct an ABE-CR^2-SKL scheme $\mathsf{ABE\text{-}CR^2\text{-}SKL} = \mathsf{ABE\text{-}CR^2\text{-}SKL}.(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ using the following building blocks:

- Multi-Input (Ciphertext-Policy) ABE Scheme $\mathsf{MI\text{-}ABE} = \mathsf{MI\text{-}ABE}.(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ for the following relation:

  $R(\widetilde{x} \| z, y_1, \cdots, y_k)$**:** Let $\widetilde{x}$ be interpreted as a circuit and z as a k-input circuit.
  - Parse $y_1 = t_1 \| u_1 \| y$.
  - Parse $y_i = t_i \| u_i$ for every $i \in 2, \ldots, k$.
  - If $t_i \neq t_j$ for some $i, j \in [k]$, output 1. Otherwise, go to the next step.
  - If $z(u_1 \| \cdots \| u_k) = \bot$ AND $\widetilde{x}(y) = 0$, output 0 (decryptable). Else, output 1.

- Compute-and-Compare Obfuscation $\mathsf{CC.Obf}$ with the simulator $\mathsf{CC.Sim}$.
- BB84-based SKE-CD scheme $\mathsf{SKECD} = \mathsf{SKECD}.(\mathsf{KG}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ with the classical decryption algorithm $\mathsf{SKECD.CDec}$.

The description of each algorithm of $\mathsf{ABE\text{-}CR^2\text{-}SKL}$ is as follows.

$\mathsf{ABE\text{-}CR^2\text{-}SKL.Setup}(1^\lambda)$**:**

- Generate $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{MI\text{-}ABE.Setup}(1^\lambda)$.
- Generate $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$.
- Output $\mathsf{ek} := \mathsf{abe.pk}$ and $\mathsf{msk} := (\mathsf{abe.msk}, \mathsf{skecd.sk})$.

$\mathsf{ABE\text{-}CR^2\text{-}SKL}.\mathcal{KG}(\mathsf{msk}, y)$**:**

- Parse $\mathsf{msk} = (\mathsf{abe.msk}, \mathsf{ske.msk})$.
- Sample a random value $t \leftarrow \{0,1\}^\lambda$.
- Generate $(\mathsf{skecd}.\mathit{ct}, \mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, 0^\lambda)$. $\mathsf{skecd.vk}$ is of the form $(x, \theta) \in \{0,1\}^k \times \{0,1\}^k$, and $\mathsf{skecd}.\mathit{ct}$ is of the form $|\psi_1\rangle_{\mathsf{SKECD.CT_1}} \otimes \cdots \otimes |\psi_k\rangle_{\mathsf{SKECD.CT_k}}$.
- Generate $\mathsf{abe.sk}_{1,b} \leftarrow \mathsf{MI\text{-}ABE.KG}(\mathsf{abe.msk}, 1, t\|b\|y)$ for each $b \in \{0,1\}$.
- For every $i \in 2, \ldots, k$, do the following:
  - Generate $\mathsf{abe.sk}_{i,b} \leftarrow \mathsf{MI\text{-}ABE.KG}(\mathsf{abe.msk}, i, t\|b)$ for each $b \in \{0,1\}$.
- For every $i \in [k]$, do the following:
  - Prepare a register $\mathsf{ABE.SK_i}$ that is initialized to $|0 \cdots 0\rangle_{\mathsf{ABE.SK_i}}$.
  - Apply the map $|u\rangle_{\mathsf{SKECD.CT_i}} |v\rangle_{\mathsf{ABE.SK_i}} \rightarrow |u\rangle_{\mathsf{SKECD.CT_i}} |v \oplus \mathsf{abe.sk}_{i,u}\rangle_{\mathsf{ABE.SK_i}}$ to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{ABE.SK_i}$, and obtain the resulting state ρ_i.
- Output $\mathit{sk} := (\rho_i)_{i \in [k]}$ and $\mathsf{vk} := \left(\mathsf{skecd.vk}, \{\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1}\}_{i \in [k]:\theta[i]=1}\right)$.

$\mathsf{ABE\text{-}CR^2\text{-}SKL.Enc}(\mathsf{ek}, \widetilde{x}, \mathsf{m})$**:**

- Parse $\mathsf{ek} = \mathsf{abe.pk}$.
- Generate $\widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$, where pp_D consists of circuit parameters of D defined in the security proof.
- Generate $\mathsf{abe.ct} \leftarrow \mathsf{MI\text{-}ABE.Enc}(\mathsf{abe.pk}, \widetilde{x}\|\widetilde{C}, \mathsf{m})$.
- Output $\mathsf{ct} := \mathsf{abe.ct}$.

$\mathsf{ABE\text{-}CR^2\text{-}SKL}.\mathcal{Dec}(\mathit{sk}, \mathsf{ct})$**:**

- Parse $\mathit{sk} = (\rho_i)_{i \in [k]}$ and $\mathsf{ct} = \mathsf{abe.ct}$. We denote the register holding ρ_i as $\mathsf{SKECD.CT_i} \otimes \mathsf{ABE.SK_i}$.
- Prepare a register MSG that is initialized to $|0 \cdots 0\rangle_{\mathsf{MSG}}$
- To the registers $\bigotimes_{i \in [k]} \mathsf{ABE.SK_i}$ and MSG, apply $\bigotimes_{i \in [k]} |v_i\rangle_{\mathsf{ABE.SK_i}} \otimes |w\rangle_{\mathsf{MSG}} \rightarrow \bigotimes_{i \in [k]} |v_i\rangle_{\mathsf{ABE.SK_i}} \otimes |w \oplus \mathsf{MI\text{-}ABE.Dec}(\mathsf{abe.ct}, v_1, \cdots, v_k)\rangle_{\mathsf{MSG}}$.
- Measure the register MSG in the computational basis and output the result m'.

$\mathsf{ABE\text{-}CR^2\text{-}SKL}.\mathcal{Del}(\mathit{sk})$**:**

- Parse $\mathit{sk} = (\rho_i)_{i \in [k]}$. Let the register holding ρ_i be denoted as $\mathsf{SKECD.CT_i} \otimes \mathsf{ABE.SK_i}$.
- For each $i \in [k]$, measure the registers $\mathsf{SKECD_i}$ and $\mathsf{ABE.SK_i}$ in the Hadamard basis to obtain outcomes c_i and d_i.
- Output $\mathsf{cert} = (c_i, d_i)_{i \in [k]}$.

$\mathsf{ABE\text{-}CR^2\text{-}SKL.Vrfy}(\mathsf{vk}, \mathsf{cert})$**:**

- Parse $\mathsf{vk} = \left(\mathsf{skecd.vk} = (x, \theta), \{\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1}\}_{i \in [k]:\theta[i]=1}\right)$ and $\mathsf{cert} = (c_i, d_i)_{i \in [k]}$.
- Output $\top$ if $x[i] = c_i \oplus d_i \cdot (\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1})$ holds for every $i \in [k]$ such that $\theta[i] = 1$ and $\bot$ otherwise.

Let $\mathcal{sk} \leftarrow \mathsf{ABE\text{-}CR^2\text{-}SKL}.\mathcal{KG}(\mathsf{msk}, y)$. $\mathcal{sk}$ is of the form $(\rho_i)_{i\in[k]}$, where ρ_i is of the following form, where (x, θ) is the verification key of a BB84-based SKECD scheme:

$$\rho_i = \begin{cases} |x[i]\rangle \left|\mathsf{abe.sk}_{i,x[i]}\right\rangle & if \;\; \theta[i] = 0 \\ |0\rangle |\mathsf{abe.sk}_{i,0}\rangle + (-1)^{x[i]} |1\rangle |\mathsf{abe.sk}_{i,1}\rangle & if \;\; \theta[i] = 1, \end{cases}$$

Recall that $\mathsf{abe.sk}_{i,b} = \mathsf{MI\text{-}ABE.KG}(\mathsf{abe.msk}, i, t\|b)$ for every $i \in \{2, \ldots, k\}$ and $b \in \{0, 1\}$. Moreover, $\mathsf{abe.sk}_{1,b} = \mathsf{MI\text{-}ABE.KG}(\mathsf{abe.msk}, 1, t\|b\|y)$ for each $b \in \{0, 1\}$.

Decryption correctness. The MI-ABE relation defined in the construction is as follows:

$R(\widetilde{x}\|z, y_1, \cdots, y_k)$**:** Let $\widetilde{x}$ be interpreted as a circuit and z as a k-input circuit.

- Parse $y_1 = t_1\|u_1\|y$.
- Parse $y_i = t_i\|u_i$ for every $i \in 2, \ldots, k$.
- If $t_i \neq t_j$ for some $i, j \in [k]$, output 1. Otherwise, go to the next step.
- If $z(u_1\| \cdots \|u_k) = \bot$ AND $\widetilde{x}(y) = 0$, output 0 (decryptable). Else, output 1.

Clearly, $z := \widetilde{C} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$ always outputs $\bot$ and $t_1 = \ldots = t_k$ holds by construction. Hence, the guarantee follows from the decryption correctness of MI-ABE, as long as $\widetilde{x}(y) = 0$ holds, as desired.

Verification correctness. Recall that the verification only checks the Hadamard basis positions, i.e, positions $i \in [k]$ such that $\theta[i] = 1$. Consider the outcome (c_i, d_i) obtained by measuring the state $|0\rangle |\mathsf{abe.sk}_{i,0}\rangle + (-1)^{x[i]} |1\rangle |\mathsf{abe.sk}_{i,1}\rangle$ in the Hadamard basis, where c_i denotes the first bit of the outcome. It is easy to see that $x[i] = c_i \oplus d_i \cdot (\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1})$ is satisfied. Hence, the verification correctness follows.

8.4 Proof of Selective IND-KLA Security

Let $\mathcal{A}$ be an adversary for the selective IND-KLA security of $\mathsf{ABE\text{-}CR^2\text{-}SKL}$. We consider the following sequence of experiments.

$\mathsf{Hyb}_0^{\mathsf{coin}}$**:** This is $\mathsf{Exp}^{\mathsf{sel\text{-}ind\text{-}kla}}_{\mathsf{ABE\text{-}CR^2\text{-}SKL},\mathcal{A}}(1^\lambda, \mathsf{coin})$.

1. $\mathcal{A}$ declares the challenge ciphertext attribute $x^* \in \mathcal{X}$.
2. The challenger $\mathcal{Ch}$ generates $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{MI\text{-}ABE.Setup}(1^\lambda)$ and $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$, and sends $\mathsf{ek} := \mathsf{abe.pk}$ to $\mathcal{A}$.
3. $\mathcal{A}$ can get access to the following (stateful) oracles, where the list $L_{\mathcal{KG}}$ used by the oracles is initialized to an empty list:

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

   $O_{\mathsf{Vrfy}}(y, \mathsf{cert})$**:** Given (y, cert), it finds an entry (y, vk_y, V_y) from $L_{\mathcal{KG}}$. (If there is no such entry, it returns $\bot$.) It then runs $d := \mathsf{Vrfy}(\mathsf{vk}_y, \mathsf{cert})$ and returns d to $\mathcal{A}$. If $V_y = \bot$, it updates the entry into (y, vk_y, d).
4. $\mathcal{A}$ sends $(\mathsf{m}_0^*, \mathsf{m}_1^*) \in \mathcal{M}^2$ to the challenger. If $V_j = \bot$ for some $j \in [q]$, $\mathcal{Ch}$ outputs 0 as the final output of this experiment. Otherwise, $\mathcal{Ch}$ generates $\widetilde{C}^* \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_D, 1)$ and $\mathsf{abe.ct}^* \leftarrow \mathsf{MI\text{-}ABE.Enc}(\mathsf{abe.pk}, x^*\|\widetilde{C}^*, \mathsf{m}^*_{\mathsf{coin}})$, and sends $\mathsf{ct}^* := \mathsf{abe.ct}^*$ to $\mathcal{A}$.
5. $\mathcal{A}$ outputs coin'. $\mathcal{Ch}$ outputs coin' as the final output of the experiment.

$\mathsf{Hyb}_1^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_0^{\mathsf{coin}}$ except that $\widetilde{C}^*$ is generated as $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{lock} \oplus r, \mathsf{skecd.sk}], \mathsf{lock}, 0)$, where $\mathsf{lock} \leftarrow \{0,1\}^\lambda$, $r \leftarrow \{0,1\}^\lambda$ and $D[\mathsf{lock} \oplus r, \mathsf{skecd.sk}](x)$ is a circuit that has $\mathsf{lock} \oplus r$ and $\mathsf{skecd.sk}$ hardwired and outputs $\mathsf{lock} \oplus r \oplus \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, x)$.

Since lock is chosen at random independently of all other variables, from the security of compute-and-compare obfuscation, we have that $\mathsf{Hyb}_0^{\mathsf{coin}} \approx \mathsf{Hyb}_1^{\mathsf{coin}}$.

$\mathsf{Hyb}_2^{\mathsf{coin}}$**:** This is the same as $\mathsf{Hyb}_1^{\mathsf{coin}}$ except that $\mathsf{skecd}.ct$ generated as part of $\mathsf{ABE\text{-}CR^2\text{-}SKL}.\mathcal{KG}(\mathsf{msk}, y)$ such that $x^*(y) = 0$, is generated as $(\mathsf{skecd}.ct, \mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{E}nc(\mathsf{skecd.sk}, r)$.

In the previous step, we changed the distribution of $\widetilde{C}^*$ used to generate the challenge ciphertext so that $\widetilde{C}^*$ has $\mathsf{skecd.sk}$ hardwired. However, the ciphertext is given to $\mathcal{A}$ after $\mathcal{A}$ deletes all the leased secret keys satisfying the ABE relation, and thus the corresponding ciphertetexts of SKECD. Thus, we can use the security of SKECD to argue that $\mathsf{Hyb}_1^{\mathsf{coin}} \approx \mathsf{Hyb}_2^{\mathsf{coin}}$. Let q be the number of key-queries made to $O_{\mathcal{KG}}$ that satisfy the relation wrt x^*. Consider now $q+1$ hybrids $\mathsf{Hyb}_{1,0}^{\mathsf{coin}}, \cdots, \mathsf{Hyb}_{1,q}^{\mathsf{coin}}$ where for every $l \in [q]$, $\mathsf{Hyb}_{1,l}^{\mathsf{coin}}$ is such that the first l of the q keys are generated using SKECD encryptions of r (instead of 0^λ). Notice that $\mathsf{Hyb}_{1,0}^{\mathsf{coin}} \equiv \mathsf{Hyb}_1^{\mathsf{coin}}$ and $\mathsf{Hyb}_{1,q}^{\mathsf{coin}} \equiv \mathsf{Hyb}_2^{\mathsf{coin}}$. Suppose that $\mathsf{Hyb}_{1,l}^{\mathsf{coin}} \not\approx \mathsf{Hyb}_{1,l+1}^{\mathsf{coin}}$ for some $l \in [q]$. Let $\mathcal{D}$ be a distinguisher with non-negligible advantage in distinguishing the hybrids. We will construct the following reduction to the IND-CVA-CD security of SKECD:

Execution of $\mathcal{R}^{\mathcal{D}}$ in $\mathsf{Exp}_{\mathsf{SKECD},\mathcal{R}}^{\mathsf{ind\text{-}cva\text{-}cd}}(1^\lambda, b)$:

1. The challenger $\mathcal{C}h$ computes $\mathsf{sk} \leftarrow \mathsf{KG}(1^\lambda)$.
2. $\mathcal{D}$ declares a challenge ciphertext attribute x^* to $\mathcal{R}$. $\mathcal{R}$ generates $(\mathsf{abe.pk}, \mathsf{abe.msk}) \leftarrow \mathsf{MI\text{-}ABE.Setup}(1^\lambda)$ and sends $\mathsf{ek} := \mathsf{abe.pk}$ to $\mathcal{D}$.
3. $\mathcal{R}$ sends $(r_0, r_1) := (0^\lambda, r)$ to $\mathcal{C}h$.
4. $\mathcal{C}h$ computes $(ct^\star, \mathsf{vk}^\star) \leftarrow \mathcal{E}nc(\mathsf{sk}, r_b)$ and sends $ct^\star$ to $\mathcal{R}$.
5. $\mathcal{R}$ simulates the oracle $O_{\mathcal{KG}}(y)$ for $\mathcal{D}$ as follows. Given y, it finds an entry (y, vk_y, V_y) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\bot$. Otherwise, it proceeds as follows:
   - For the j-th query (of the q satisfying key-queries), if $j \neq l+1$, $\mathcal{R}$ computes $s\mathcal{k}_y$ and vk_y as in $\mathsf{Hyb}_0^{\mathsf{coin}}$, except that the values $(\mathsf{skecd}.ct, \mathsf{skecd.vk})$ are computed using $O_{\mathsf{Enc}}(r)$ for $j \in [l]$ and using $O_{\mathsf{Enc}}(0^\lambda)$ for $j \in \{l+2, \ldots, q\}$.
   - For the j-th query (of the q satisfying key-queries), if $j = l+1$, $\mathcal{R}$ computes $s\mathcal{k}_y$ as in $\mathsf{Hyb}_0^{\mathsf{coin}}$, except that the value $ct^\star$ is used instead of $\mathsf{skecd}.ct$. Consider the values $\{\mathsf{abe.sk}_{i,0}, \mathsf{abe.sk}_{i,1}\}_{i \in [k]: \theta[i]=1}$ computed during the computation of $s\mathcal{k}_y$. $\mathcal{R}$ computes the value $\widetilde{\mathsf{vk}}_y = \{\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1}\}_{i \in [k]}$.
   - If $x^*(y) \neq 0$ (unsatisfying key-queries), $\mathcal{R}$ computes $s\mathcal{k}_y$ and vk_y as in $\mathsf{Hyb}_0^{\mathsf{coin}}$, except that $(\mathsf{skecd}.ct, \mathsf{skecd.vk})$ are computed using $O_{\mathsf{Enc}}(0^\lambda)$.
6. $\mathcal{R}$ simulates the view of oracle $O_{\mathsf{Vrfy}}(y, \mathsf{cert})$ for $\mathcal{D}$ as follows. Given (y, cert), it finds an entry (y, vk_y, V_y) from $L_{\mathcal{KG}}$. If there is no such entry, it returns $\bot$. Otherwise, it proceeds as follows:
   - If $x^*(y) \neq 0$, $\mathcal{R}$ simulates access to O_{Vrfy} as in $\mathsf{Hyb}_0^{\mathsf{coin}}$.
   - If y corresponds to the j-th satisfying key-query to $O_{\mathcal{KG}}$ and $j \neq l+1$, $\mathcal{R}$ simulates access to O_{Vrfy} as in $\mathsf{Hyb}_0^{\mathsf{coin}}$.
   - If y corresponds to the j-th satisfying key-query to $O_{\mathcal{KG}}$ and $j = l+1$, $\mathcal{R}$ simulates access to $O_{\mathsf{Vrfy}}(y, \mathsf{cert})$ as follows:
     (a) Parse $\widetilde{\mathsf{vk}}_y = \{\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1}\}_{i \in [k]}$ and $\mathsf{cert} = (c_i, d_i)_{i \in [k]}$.
     (b) Compute $x[i] = c_i \oplus d_i \cdot (\mathsf{abe.sk}_{i,0} \oplus \mathsf{abe.sk}_{i,1})$ for every $i \in [k]$.
     (c) If $O_{\mathsf{Vrfy}}(x) = \mathsf{sk}$, set $d := \top$. Else, set $d := \bot$.

   It returns d to $\mathcal{D}$. If $V_y = \bot$, it updates the entry in $L_{\mathcal{KG}}$ into (y, vk_y, d).

7. $\mathcal{D}$ sends $(m_0^*, m_1^*) \in \mathcal{M}^2$ to $\mathcal{R}$. $\mathcal{R}$ computes ct^* as in Step 4. of $\mathsf{Hyb}_1^{\mathsf{coin}}$ using $\mathsf{skecd.sk} := \mathsf{sk}$ obtained from $\mathcal{Ch}$. It sends ct^* to $\mathcal{D}$ if $V_y = \top$ for every entry of the form (y, vk_y, V_y) in $L_{\mathcal{KG}}$ such that $x^*(y) = 0$. Else, it outputs $\perp$.
8. $\mathcal{D}$ guesses a bit b'. $\mathcal{R}$ sends b' to $\mathcal{Ch}$.

Notice that the view of $\mathcal{D}$ in the reduction is that of $\mathsf{Hyb}_{1,l}^{\mathsf{coin}}$ if $b = 0$ and $\mathsf{Hyb}_{1,l+1}^{\mathsf{coin}}$ if $b = 1$. Moreover, $\mathcal{D}$ can only distinguish when $V_y = \top$ for all entries $(y, \mathsf{vk}_y, V_y) \in L_{\mathcal{KG}}$ such that $x^*(y) = 0$. This means that V_y corresponding to the $(l+1)$-th satisfying query to $O_{\mathcal{KG}}$ also satisfies $V_y = \top$. Consequently, the corresponding verification check of $\mathcal{Ch}$ is also $\top$. Hence, $\mathcal{R}$ succeeds in breaking IND-CVA-CD security of SKECD. Therefore, we have $\mathsf{Hyb}_1^{\mathsf{coin}} \approx \mathsf{Hyb}_2^{\mathsf{coin}}$.

We will now bound the distinguishing gap between Hyb_2^0 and Hyb_2^1 using the security of $\mathsf{MI\text{-}ABE}$. Consider a decryption key $\mathit{sk}_y = (\rho_i)_{i\in[k]}$ such that $x^*(y) = 0$. Recall that a ciphertext of BB84-based SKECD is of the form $|\psi_1\rangle \otimes \cdots \otimes |\psi_k\rangle$, where

$$|\psi_i\rangle = \begin{cases} |x[i]\rangle & \text{if } \theta[i] = 0 \\ |0\rangle + (-1)^{x[i]}\,|1\rangle & \text{if } \theta[i] = 1 \end{cases}$$

As a result, we have

$$\rho_i = \begin{cases} |x[i]\rangle \left|\mathsf{abe.sk}_{i,x[i]}\right\rangle & \text{if } \theta[i] = 0 \\ |0\rangle\,|\mathsf{abe.sk}_{i,0}\rangle + (-1)^{x[i]}\,|1\rangle\,|\mathsf{abe.sk}_{i,1}\rangle & \text{if } \theta[i] = 1, \end{cases}$$

where $\mathsf{abe.sk}_{1,b} \leftarrow \mathsf{MI\text{-}ABE.KG}(\mathsf{abe.msk}, 1, t_y \| b \| y)$ for each $b \in \{0,1\}$ and $\mathsf{abe.sk}_{i,b} \leftarrow \mathsf{MI\text{-}ABE.KG}(\mathsf{abe.msk}, i, t_y \| b)$ for each $i \in \{2, \ldots, k\}$ and $b \in \{0,1\}$. Here, t_y is a random value that is common for each of the k "slots" of the decryption-key sk_y. Notice first that no two values t_y, t_w are equal for t_w corresponding to some sk_w except with negligible probability. Consequently, due to the defined relation R and the selective security of $\mathsf{MI\text{-}ABE}$, any set of k secret keys is "decryptable" only if they correspond to the same decryption key sk_y. Now, we notice that $\mathsf{abe.sk}_{i,1-x[i]}$ is not given to $\mathcal{A}$ for any $i \in [k]$ such that $\theta[i] = 0$. This means that for any set of k $\mathsf{MI\text{-}ABE}$ keys corresponding to the decryption-key sk_y, the attributes $(x'[1], \cdots, x'[k])$ satisfy $x'[i] = x[i]$ for all $i : \theta[i] = 0$. Consequently, the classical decryption property (See Definition 3.12) of the BB84-based SKE-CD scheme guarantees that $\mathsf{lock} \oplus r \oplus \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, x') = \mathsf{lock}$ in these hybrids. This means that $\widetilde{C}^*(x') = 0$ (instead of $\perp$). As a result, any subset of $\mathsf{MI\text{-}ABE}$ keys of size k given to $\mathcal{A}$ is a "non-decryptable" set in the hybrids Hyb_2^0 and Hyb_2^1. It is easy to see now that we can reduce to the selective security of $\mathsf{MI\text{-}ABE}$. Specifically, the reduction specifies the target attribute $x^* \| \widetilde{C}^*$ where $\widetilde{C}^* \leftarrow \mathsf{CC.Obf}(1^\lambda, D[\mathsf{lock} \oplus r, \mathsf{skecd.sk}], \mathsf{lock}, 0)$ and simulates the view for $\mathcal{A}$ by querying the key-generation oracle of $\mathsf{MI\text{-}ABE}$ accordingly. Note that the reduction can handle verification queries since it can obtain both $\mathsf{abe.sk}_{i,0}$ and $\mathsf{abe.sk}_{i,1}$ for every $i \in [k]$ such that $\theta[i] = 1$ that are sufficient to perform the verification. (Especially, it does not need $\mathsf{abe.sk}_{i,1-x[i]}$ for $i \in [k]$ such that $\theta[i] = 0$ for verification.) We now state the following theorem:

Theorem 8.7. *There exists an ABE-CR^2-SKL scheme satisfying selective IND-KLA Security, assuming the existence of a selectively-secure Multi-Input ABE scheme for polynomial-arity, Compute-and-Compare Obfuscation, and a BB84-based SKE-CD scheme.*

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010. (Cited on page 14.)

[AGKZ20] Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 255–268. ACM Press, June 2020. (Cited on page 6.)

[AHH24] Prabhanjan Ananth, Zihan Hu, and Zikuan Huang. Quantum key-revocable dual-regev encryption, revisited. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part III*, volume 15366 of *LNCS*, pages 257–288. Springer, Cham, December 2024. (Cited on page 3, 4, 5.)

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Cham, August 2019. (Cited on page 12.)

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Berlin, Heidelberg, August 2015. (Cited on page 3.)

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. (Cited on page 3.)

[AKN$^+$23] Shweta Agrawal, Fuyuki Kitagawa, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Public key encryption with secure key leasing. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 581–610. Springer, Cham, April 2023. (Cited on page 3, 4, 5, 7, 8, 32, 33.)

[AKY24] Shweta Agrawal, Simran Kumari, and Shota Yamada. Pseudorandom multi-input functional encryption and applications. *IACR Cryptol. ePrint Arch.*, page 1720, 2024. (Cited on page 5, 41.)

[AL21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 501–530. Springer, Cham, October 2021. (Cited on page 5.)

[AMP24] Prabhanjan Ananth, Saachi Mutreja, and Alexander Poremba. Revocable encryption, programs, and more: The case of multi-copy security. *IACR Cryptol. ePrint Arch.*, page 1687, 2024. (Cited on page 6.)

[APV23] Prabhanjan Ananth, Alexander Poremba, and Vinod Vaikuntanathan. Revocable cryptography from learning with errors. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 93–122. Springer, Cham, November / December 2023. (Cited on page 3, 4, 5, 7.)

[ARYY23] Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 532–564. Springer, Cham, August 2023. (Cited on page 5, 41.)

[ATY23] Shweta Agrawal, Junichi Tomida, and Anshu Yadav. Attribute-based multi-input FE (and more) for attribute-weighted sums. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 464–497. Springer, Cham, August 2023. (Cited on page 5.)

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 149–178. Springer, Cham, November 2020. (Cited on page 5.)

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 13–43. Springer, Cham, May 2020. (Cited on page 5.)

[AYY22] Shweta Agrawal, Anshu Yadav, and Shota Yamada. Multi-input attribute based encryption and predicate encryption. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 590–621. Springer, Cham, August 2022. (Cited on page 5.)

[BB20] Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *arXiv preprint arXiv:2003.06557*, 2020. (Cited on page 10.)

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. (Cited on page 9, 11, 14, 52, 53.)

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. (Cited on page 6.)

[BGK$^+$24] James Bartusek, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts. Software with certified deletion. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part IV*, volume 14654 of *LNCS*, pages 85–111. Springer, Cham, May 2024. (Cited on page 3, 4, 5, 6.)

[BI20] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 92–122. Springer, Cham, November 2020. (Cited on page 6.)

[BJK$^+$18] Zvika Brakerski, Aayush Jain, Ilan Komargodski, Alain Passelègue, and Daniel Wichs. Non-trivial witness encryption and null-iO from standard assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 425–441. Springer, Cham, September 2018. (Cited on page 41.)

[BJL$^+$21] Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 90–120. Springer, Cham, November 2021. (Cited on page 5.)

[BK23] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 192–223. Springer, Cham, August 2023. (Cited on page 6, 54.)

[BKM$^+$23] James Bartusek, Dakshita Khurana, Giulio Malavolta, Alexander Poremba, and Michael Walter. Weakening assumptions for publicly-verifiable deletion. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 183–197. Springer, Cham, November / December 2023. (Cited on page 6.)

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 418–449. Springer, Singapore, December 2024. (Cited on page 6.)

[BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM*, 65(6):39:1–39:37, 2018. (Cited on page 3.)

[BV22] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. In Mark Braverman, editor, *ITCS 2022*, volume 215, pages 28:1–28:20. LIPIcs, January / February 2022. (Cited on page 6.)

[BZ13a] Dan Boneh and Mark Zhandry. Quantum-secure message authentication codes. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 592–608. Springer, Berlin, Heidelberg, May 2013. (Cited on page 53.)

[BZ13b] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 361–379. Springer, Berlin, Heidelberg, August 2013. (Cited on page 14, 53.)

[ÇG24] Alper Çakan and Vipul Goyal. Unclonable cryptography with unbounded collusions and impossibility of hyperefficient shadow tomography. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part III*, volume 15366 of *LNCS*, pages 225–256. Springer, Cham, December 2024. (Cited on page 3.)

[CGJL23] Orestis Chardouvelis, Vipul Goyal, Aayush Jain, and Jiahui Liu. Quantum key leasing for PKE and FHE with a classical lessor. Cryptology ePrint Archive, Report 2023/1640, 2023. (Cited on page 3, 4, 5.)

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 556–584, Virtual Event, August 2021. Springer, Cham. (Cited on page 6.)

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. *arXiv (CoRR)*, abs/2009.13865, 2020. (Cited on page 5.)

[CV22] Eric Culf and Thomas Vidick. A monogamy-of-entanglement game for subspace coset states. *Quantum*, 6:791, sep 2022. (Cited on page 6.)

[FFMV23] Danilo Francati, Daniele Friolo, Giulio Malavolta, and Daniele Venturi. Multi-key and multi-input predicate encryption from learning with errors. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 573–604. Springer, Cham, April 2023. (Cited on page 5.)

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, Berlin, Heidelberg, August 1994. (Cited on page 5.)

[GGG$^+$14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Berlin, Heidelberg, May 2014. (Cited on page 5, 41.)

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. (Cited on page 6.)

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Berlin, Heidelberg, August 2013. (Cited on page 6.)

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017. (Cited on page 4, 9, 15.)

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 661–695. Springer, Cham, August 2017. (Cited on page 41.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on page 5.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Berlin, Heidelberg, August 2012. (Cited on page 30.)

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877, 2020. (Cited on page 6.)

[HKM$^+$24] Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa. Certified everlasting secure collusion-resistant functional encryption, and more. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part III*, volume 14653 of *LNCS*, pages 434–456. Springer, Cham, May 2024. (Cited on page 6, 11.)

[HMNY21] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 606–636. Springer, Cham, December 2021. (Cited on page 6.)

[KMY24] Fuyuki Kitagawa, Tomoyuki Morimae, and Takashi Yamakawa. A simple framework for secure key leasing. *ArXiv (CoRR)*, abs/2410.03413, 2024. (Cited on page 3, 4, 5.)

[KN22] Fuyuki Kitagawa and Ryo Nishimaki. Functional encryption with secure key leasing. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 569–598. Springer, Cham, December 2022. (Cited on page 3, 6, 16.)

[KN23] Fuyuki Kitagawa and Ryo Nishimaki. One-out-of-many unclonable cryptography: Definitions, constructions, and more. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 246–275. Springer, Cham, November / December 2023. (Cited on page 5.)

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 31–61. Springer, Cham, November 2021. (Cited on page 5.)

[KNY23] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Publicly verifiable deletion from minimal assumptions. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 228–245. Springer, Cham, November / December 2023. (Cited on page 6, 11.)

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979. (Cited on page 11.)

[LLQZ22] Jiahui Liu, Qipeng Liu, Luowen Qian, and Mark Zhandry. Collusion resistant copy-protection for watermarkable functionalities. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 294–323. Springer, Cham, November 2022. (Cited on page 6.)

[MPY23] Tomoyuki Morimae, Alexander Poremba, and Takashi Yamakawa. Revocable quantum digital signatures. Cryptology ePrint Archive, Report 2023/1937, 2023. (Cited on page 3.)

[Por23] Alexander Poremba. Quantum proofs of deletion for learning with errors. In Yael Tauman Kalai, editor, *ITCS 2023*, volume 251, pages 90:1–90:14. LIPIcs, January 2023. (Cited on page 6.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. (Cited on page 5.)

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Berlin, Heidelberg, May 2005. (Cited on page 3.)

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022. (Cited on page 41.)

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Cham, December 2022. (Cited on page 41.)

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Cham, May / June 2022. (Cited on page 6.)

[Wie83] Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983. (Cited on page 10.)

[Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999. (Cited on page 17.)

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017. (Cited on page 4, 9, 15.)

[Zha12] Mark Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, October 2012. (Cited on page 52.)

A Quantum Secure ABE for All Relations from LWE

We show there exists quantum selective-secure ABE for all relations computable in polynomial time based on the LWE assumption.

A.1 Preparation

Definition A.1 (Quantum-Accessible Pseudo-Random Function). *Let* $\{\mathsf{PRF}_k : \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_2} \mid k \in \{0,1\}^\lambda\}$ *be a family of polynomially computable functions, where* ℓ_1 *and* ℓ_2 *are some polynomials of* λ*. We say that* PRF *is a quantum-accessible pseudo-random function (QPRF) family if for any QPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{A}}(\lambda) = \left|\Pr\left[\mathcal{A}^{|\mathsf{PRF}_k(\cdot)\rangle}(1^\lambda) \leftarrow 1 \mid k \leftarrow \{0,1\}^\lambda\right] - \Pr\left[\mathcal{A}^{|\mathsf{R}(\cdot)\rangle}(1^\lambda) \leftarrow 1 \mid \mathsf{R} \leftarrow \mathcal{U}\right]\right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{U}$ *is the set of all functions from* $\{0,1\}^{\ell_1}$ *to* $\{0,1\}^{\ell_2}$.

Theorem A.2 ([Zha12]). *If there exists a OWF, there exists a QPRF.*

A.2 Proofs

We first define Key-Policy ABE for polynomial size circuits and briefly see that it can be used to instantiate ABE for any relations computable in polynomial time, even under quantum selective-security. Then, we prove that the Key-Policy ABE scheme for polynomial size circuits by Boneh et al. [BGG+14] with a light modification using QPRF satisfies quantum selective-security under the LWE assumption.

Key-Policy ABE for Circuits: Let $\mathcal{X}_\lambda = \{0,1\}^{n(\lambda)}$ and $\mathcal{Y}_\lambda$ be the set of all circuits with input space $\{0,1\}^{n(\lambda)}$ and size at most $s(\lambda)$, where n and s are some polynomials. Let R_λ be the following relation:

$$R_\lambda(x,y) = 0 \iff y(x) = 0$$

An ABE scheme for such $\{\mathcal{X}_\lambda\}_\lambda$, $\{\mathcal{Y}_\lambda\}_\lambda$, and $\{R_\lambda\}_\lambda$ is referred to as a Key-Policy ABE scheme for circuits.

Lemma A.3. *If there exists a quantum selective-secure Key-Policy ABE scheme for circuits, then there exists a quantum selective-secure ABE scheme for all relations computable in polynomial time.*

Proof. Let $\mathcal{X}_\lambda \subseteq \{0,1\}^n$, $\mathcal{Y}_\lambda \subseteq \{0,1\}^\ell$, and $R_\lambda : \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}$, where n and ℓ are polynomials and R_λ is computable in polynomial time. We construct ABE scheme $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ with attribute spaces $\{\mathcal{X}_\lambda\}$ and $\{\mathcal{Y}_\lambda\}$ and relation $\{R_\lambda\}$, using Key-Policy ABE scheme for circuits $\mathsf{ABE}' = (\mathsf{Setup}, \mathsf{KG}', \mathsf{Enc}, \mathsf{Dec})$ with the following attribute spaces $\{\mathcal{X}'_\lambda\}$ and $\{\mathcal{Y}'_\lambda\}$.

- $\mathcal{X}'_\lambda = \{0,1\}^n$.
- Let $C_{\lambda,y}$ be the circuit such that $C_{\lambda,y}(x) = R_\lambda(x,y)$ for every $x \in \{0,1\}^n$ and $y \in \{0,1\}^\ell$, and let s be the maximum size of C_y. Note that s is a polynomial in λ. Then, $\mathcal{Y}'_\lambda$ is the set of all circuits with input length n and size at most s'.

The scheme is as follows: $\mathsf{KG}'(\mathsf{msk}, y, r) = \mathsf{KG}(\mathsf{msk}, C_{\lambda,y}, r)$. It is easy to see that the correctness of ABE follows from that of ABE'. For the quantum selective security of ABE', consider a reduction $\mathcal{R}$ to the quantum selective security of ABE.

Execution of $\mathcal{R}^{\mathcal{A}}$ in Experiment $\mathsf{Exp}^{\mathsf{q\text{-}sel\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda, \mathsf{coin})$:

1. $\mathcal{R}$ receives challenge attribute $x^* \in \mathcal{X}_\lambda$ from $\mathcal{A}$ and forwards it to the challenger Ch.
2. For each oracle query (Y, Z) made by $\mathcal{A}$, $\mathcal{R}$ performs the following map on register Z initialized to $\left|0^{s(\lambda)}\right\rangle$:
$$|y\rangle_{\mathsf{Y}} |z\rangle_{\mathsf{Z}} \mapsto |y\rangle_{\mathsf{Y}} |z \oplus C_{\lambda,y}\rangle_{\mathsf{Z}}$$
3. Then, $\mathcal{R}$ queries the registers Y, Z to O_{qkg} followed by returning the registers Y, Z to $\mathcal{A}$.
4. When $\mathcal{A}$ sends $(\mathsf{m}_0, \mathsf{m}_1)$ to $\mathcal{R}$, $\mathcal{R}$ forwards it to Ch.
5. On receiving $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x^*, \mathsf{m}_{\mathsf{coin}})$ from Ch, $\mathcal{R}$ forwards it to $\mathcal{A}$.
6. Finally, when $\mathcal{A}$ outputs a guess coin', $\mathcal{R}$ sends coin' to Ch.

Since the view of $\mathcal{A}$ is identical to that in the quantum selective security experiment for scheme ABE', $\mathcal{R}$ ends up breaking the quantum selective security of ABE. ☐

Theorem A.4. *Assuming the polynomial hardness of the LWE problem, there exists a quantum selective-secure Key-Policy ABE scheme for circuits.*

Proof. We claim that the Key-Policy ABE scheme for circuits by Boneh et al. [BGG+14] based on LWE is quantum selective-secure. Actually, we will alter the key-generation algorithm of their scheme as follows: $\mathsf{KG}(\mathsf{msk}, y, k) = \mathsf{KG}'(\mathsf{msk}, y, \mathsf{PRF}_k(y))$ where KG' is the key-generation algorithm of their construction with explicit random coins $\mathsf{PRF}_k(y)$, and $\{\mathsf{PRF}_k\}_k$ is QPRF. This is a common technique utilized in quantum security proofs (See for Eg. [BZ13a, BZ13b]) that allows one to use a common random value for every term of a superposition. In the following discussion, whenever we discuss a hybrid titled "Game i" for some value i, it refers to the corresponding hybrid in Theorem 4.2 of [BGG+14]. Also, any indistinguishability claims between Game hybrids that are mentioned to be previously established, will refer to their work. Consider the following sequence of hybrids for the aforementioned ABE scheme ABE and a QPT adversary $\mathcal{A}$:

$\mathsf{Hyb}^{\mathsf{coin}}_0$**:** This is the same as the experiment $\mathsf{Exp}^{\mathsf{q\text{-}sel\text{-}ind}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda, \mathsf{coin})$.

$\mathsf{Hyb}^{\mathsf{coin}}_1$**:** This is similar to $\mathsf{Hyb}^{\mathsf{coin}}_0$, except that the public-key pk is generated based on the challenge attribute x^*, as in the hybrid Game 1. It was shown that the hybrids Game 0 (the original experiment) and Game 1 are statistically indistinguishable. By the same argument, it follows that $\mathsf{Hyb}^{\mathsf{coin}}_0 \approx_s \mathsf{Hyb}^{\mathsf{coin}}_1$.

$\mathsf{Hyb}^{\mathsf{coin}}_2$**:** This is similar to $\mathsf{Hyb}^{\mathsf{coin}}_1$, except that the public-key is further modified as in Game 2. Unlike the change made in $\mathsf{Hyb}^{\mathsf{coin}}_1$ though, this change requires the key-queries to be answered differently as in Game 2. More specifically, the hybrid now has a punctured master secret-key that still allows it to simulate every key-query y such that $C_y(x^*) \neq \bot$. It was shown previously that Game 1 $\approx_s$ Game 2. Consider now the intermediate hybrids $\widetilde{\mathsf{Hyb}}^{\mathsf{coin}}_1, \widetilde{\mathsf{Hyb}}^{\mathsf{coin}}_2$ that are similar to $\mathsf{Hyb}^{\mathsf{coin}}_1, \mathsf{Hyb}^{\mathsf{coin}}_2$ respectively, except that all the superposition key-queries in these hybrids are responded using independent (true) randomness in every term of the superposition. We will now restate the following lemma by [BZ13a], which we will use to argue that $\widetilde{\mathsf{Hyb}}^{\mathsf{coin}}_1 \approx_s \widetilde{\mathsf{Hyb}}^{\mathsf{coin}}_2$.

Lemma A.5. *[BZ13a] Let $\mathcal{Y}$ and $\mathcal{Z}$ be sets and for each $y \in \mathcal{Y}$, let D_y and D'_y be distributions on $\mathcal{Z}$ such that $\mathsf{SD}(D_y, D'_y) \leq \epsilon$. Let $O : \mathcal{Y} \to \mathcal{Z}$ and $O' : \mathcal{Y} \to \mathcal{Z}$ be functions such that $O(y)$ outputs $z \leftarrow D_y$ and $O'(y)$ outputs $z' \leftarrow D'_y$. Then, $O(y)$ and $O'(y)$ are ϵ'-statistically indistinguishable by quantum algorithms making q superposition oracle queries, such that $\epsilon' = \sqrt{8C_0 q^3 \epsilon}$ where C_0 is a constant.*

Recall that $\mathcal{Y}$ denotes the set of key-attributes. Let us fix a challenge ciphertext attribute x^* for the following discussion. For each $y \in \mathcal{Y}$ let the distributions $D_y^1[\mathsf{pk}], D_y^2[\mathsf{pk}]$ correspond to how a key for attribute y is sampled in the hybrids $\widetilde{\mathsf{Hyb}}_1^{\mathsf{coin}}, \widetilde{\mathsf{Hyb}}_2^{\mathsf{coin}}$ respectively, conditioned on the public key being pk. Note that for each $i \in [2]$, we consider $D_y^i[\mathsf{pk}]$ to also output the public-key pk along with the secret-key for y. We know that on average over pk, $D_y^1[\mathsf{pk}] \approx_s D_y^2[\mathsf{pk}]$ holds for all $y \in \mathcal{Y}$. It now follows from the above lemma that on average over pk, the analogously defined oracles $O_1[\mathsf{pk}], O_2[\mathsf{pk}]$ are statistically indistinguishable by algorithms making polynomially many superposition queries. Observe that with access to oracle $O_i[\mathsf{pk}]$ for each $i \in [2]$, the view of $\mathcal{A}$ in $\widetilde{\mathsf{Hyb}}_i^{\mathsf{coin}}$ (conditioned on the public-key being pk) can easily be recreated as the oracle outputs pk, which can then be used to compute the challenge ciphertext corresponding to coin. Consequently, it follows that the hybrids $\widetilde{\mathsf{Hyb}}_1^{\mathsf{coin}}$ and $\widetilde{\mathsf{Hyb}}_2^{\mathsf{coin}}$ are statistically indistinguishable. Since $\widetilde{\mathsf{Hyb}}_i^{\mathsf{coin}} \approx \mathsf{Hyb}_i^{\mathsf{coin}}$ holds for each $i \in [2]$ (by the quantum-security of PRF), we have that $\mathsf{Hyb}_1^{\mathsf{coin}} \approx \mathsf{Hyb}_2^{\mathsf{coin}}$.

$\mathsf{Hyb}_3^{\mathsf{coin}}$**:** This is similar to $\mathsf{Hyb}_2^{\mathsf{coin}}$, except that the challenge ciphertext is chosen uniformly at random, as in Game 3.

Observe that $\mathsf{Hyb}_3^0 \equiv \mathsf{Hyb}_3^1$. It was shown previously that Game 3 $\approx$ Game 2 by a reduction to LWE. Specifically, the reduction prepares the setup based on the LWE sample and the challenge ciphertext x^*, and plants the LWE challenge in the challenge ciphertext. It is easy to see that a similar reduction works in our case, thereby showing that $\mathsf{Hyb}_2^{\mathsf{coin}} \approx \mathsf{Hyb}_3^{\mathsf{coin}}$. Consequently, it follows that $\mathsf{Hyb}_0^0 \approx \mathsf{Hyb}_0^1$.

This completes the proof. □

B IND-CVA-CD Secure BB84 Based SKE-CD

To prove Theorem 3.13, we show how to transform IND-CD secure SKE-CD to IND-CVA-CD secure one.

Definition B.1 (IND-CD Security). *We define the security experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *in the same way as* $\mathsf{Exp}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *except that the adversary* $\mathcal{A}$ *is allowed to get access to the verification oracle only once. We say that* SKECD *is IND-CD secure if for any QPT* $\mathcal{A}$*, it holds that*

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

Bartusek and Khurana [BK23] showed the following theorem.

Theorem B.2 ([BK23]). *There exists an IND-CD secure BB84 based SKE-CD scheme assuming just an IND-CPA secure SKE scheme.*

We prove Theorem 3.13 by proving the following theorem.

Theorem B.3. *Let* $\mathsf{SKECD} = (\mathsf{KG}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *be a BB84-based SKE-CD scheme. If* SKECD *is IND-CD secure, then it is also IND-CVA-CD secure.*

Proof. Let $\mathcal{A}$ be a QPT adversary that attacks the IND-CVA-CD security of SKECD with q verification queries. Let $\mathtt{FV}_i$ be the event that the i-th verification query is the first verification query such that the answer for it is not $\perp$. Then, $\mathcal{A}$'s advantage can be described as follows.

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda) &:= \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, 1) = 1\right]\right| \\ &\leq \sum_{i \in [q]} \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, 0) = 1 \wedge \mathtt{FV}_i\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, 1) = 1 \wedge \mathtt{FV}_i\right]\right| \qquad (1)\end{aligned}$$

To bound each term of Equation (1), we construct the following QPT adversary $\mathcal{B}_i$ that attacks the IND-CD security of SKECD using $\mathcal{A}$.

1. $\mathcal{B}_i$ initializes $\mathcal{A}$ with the security parameter 1^λ.

2. $\mathcal{A}$ makes queries to the encryption oracle throughout the experiment.

   $O_{\mathcal{E}nc}(\mathsf{m})$**:** When $\mathcal{A}$ makes an encryption query m, $\mathcal{B}_i$ forwards it to its own encryption oracle, and sends back the answer (vk, ct) from the encryption oracle to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M}^2$, $\mathcal{B}_i$ sends $(\mathsf{m}_0, \mathsf{m}_1)$ to its challenger. On receiving the challenge ciphertext ct^* from the challenger, $\mathcal{B}_i$ measures its classical part cla. This does not affect ct^*. $\mathcal{B}_i$ then forwards ct^* to $\mathcal{A}$.

4. Hereafter, $\mathcal{A}$ can get access to the following oracle.

   $O_{\mathsf{Vrfy}}(\mathsf{cert}_j)$**:** For the j-th query cert_j, if $j < i$, $\mathcal{B}_i$ returns $\bot$ to $\mathcal{A}$. if $j = i$, $\mathcal{B}_i$ queries cert_j to its verification oracle. If the response is $\bot$, $\mathcal{B}_i$ aborts. Otherwise if the answer is sk, $\mathcal{B}_i$ forwards sk to $\mathcal{A}$. $\mathcal{B}_i$ also computes θ from cla and sk, and sets $\mathsf{vk}' = (\theta, \mathsf{cert}_i)$. $\mathcal{B}_i$ checks whether $\mathsf{Vrfy}(\mathsf{vk}', \mathsf{cert}_j) = \bot$ holds for every $j < i$ (that is, whether the i-th query is the first query resulting in the answer other than $\bot$). If not, $\mathcal{B}_i$ aborts. Otherwise, $\mathcal{B}_i$ responds to the subsequent verification queries using vk'.

5. When $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$, $\mathcal{B}_i$ outputs coin'.

Let $\mathsf{vk}^* = (\theta, x)$ be the verification key corresponding to ct^*. For any string cert, if $\mathsf{Vrfy}(\mathsf{vk}^*, \mathsf{cert}) = \top$, $\mathsf{Vrfy}(\mathsf{vk}^*, \cdot)$ and $\mathsf{Vrfy}(\mathsf{vk}', \cdot)$ for $\mathsf{vk}' = (\theta, \mathsf{cert})$ are functionally equivalent, and vk' can be used as an alternative verification key. This is because $\mathsf{Vrfy}(\mathsf{vk}^*, \mathsf{cert}')$ and $\mathsf{Vrfy}(\mathsf{vk}', \mathsf{cert}')$ respectively checks whether $\mathsf{cert}'[i] = x[i]$ and $\mathsf{cert}'[i] = \mathsf{cert}[i]$ holds or not for every $i \in [n]$ such that $\theta[i] = 1$, and for such i, we have $x[i] = \mathsf{cert}[i]$ from the fact that $\mathsf{Vrfy}(\mathsf{vk}^*, \mathsf{cert}) = \top$.

From the above, after the i-th verification query from $\mathcal{A}$ is responded, $\mathcal{B}_i$ can check whether its simulation of $\mathcal{A}$ so far has been successful or not. Moreover, if the simulation has failed, $\mathcal{B}_i$ aborts, and otherwise, $\mathcal{B}_i$ can successfully simulate the remaining steps for $\mathcal{A}$ using the alternative verification key $\mathsf{vk}' = (\theta, \mathsf{cert}_i)$. Then, we have $\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cd}}_{\mathsf{SKECD},\mathcal{B}_i}(1^\lambda, \mathsf{coin}) = 1\right] = \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cva\text{-}cd}}_{\mathsf{SKECD},\mathcal{A}}(1^\lambda, \mathsf{coin}) = 1 \land \mathtt{FV}_i\right]$. Since, SKECD satisfies IND-CD security, it holds that

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cd}}_{\mathsf{SKECD},\mathcal{B}_i}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cd}}_{\mathsf{SKECD},\mathcal{B}_i}(1^\lambda, 1) = 1\right]\right| = \mathsf{negl}(\lambda)$$

for every $i \in [q]$, which shows each term of Equation (1) is negligible. This completes the proof. $\square$

C Construction of SKFE-CR-SKL with Key Testability

C.1 Construction

We construct an SKFE-CR-SKL scheme for the functionality $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ with key testability $\mathsf{SKFE\text{-}CR\text{-}SKL} = \mathsf{SKFE\text{-}CR\text{-}SKL}.(\mathsf{Setup}, \mathcal{KG}, \mathsf{Enc}, \mathcal{D}ec, \mathcal{V}rfy)$ having the additional algorithms $\mathsf{SKFE\text{-}CR\text{-}SKL}.(\mathsf{CDec}, \mathsf{KeyTest})$, using the following building blocks.

- BB84-based SKE-CD scheme (Definition 3.12) $\mathsf{SKECD} = \mathsf{SKECD}.(\mathsf{KG}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ having the classical decryption algorithm $\mathsf{SKECD.CDec}$.
- Classical SKFE scheme $\mathsf{SKFE} = \mathsf{SKFE}.(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ for the functionality $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$.
- OWF $f : \{0,1\}^\lambda \to \{0,1\}^{p(\lambda)}$ for some polynomial p.

The construction is as follows:

$\mathsf{SKE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$**:**

1. Generate $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$.
2. Generate $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$.

3. Output $\mathsf{msk} := (\mathsf{skecd.sk}, \mathsf{skfe.msk})$.

$\mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{KG}(\mathsf{msk}, y)$**:**

1. Parse $\mathsf{msk} = (\mathsf{skecd.sk}, \mathsf{skfe.msk})$.
2. Generate $\mathsf{skfe.sk}_y \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, y)$.
3. Generate $(\mathsf{skecd}.ct, \mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, \mathsf{skfe.sk}_y)$. Recall that $\mathsf{skecd.vk}$ is of the form $(x, \theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, and $\mathsf{skecd}.ct$ is of the form $|\psi_1\rangle_{\mathsf{SKECD.CT}_1} \otimes \cdots \otimes |\psi_{\ell_{\mathsf{ct}}}\rangle_{\mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}}$.
4. Generate $s_{i,b} \leftarrow \{0,1\}^\lambda$ and compute $t_{i,b} \leftarrow f(s_{i,b})$ for every $i \in [\ell_{\mathsf{ct}}]$ and $b \in \{0,1\}$. Set $T := t_{1,0} \| t_{1,1} \| \cdots \| t_{\ell_{\mathsf{ct}},0} \| t_{\ell_{\mathsf{ct}},1}$ and $S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_{\mathsf{ct}}] : \theta[i]=1}$.
5. Prepare a register $\mathsf{S_i}$ that is initialized to $|0^\lambda\rangle_{\mathsf{S_i}}$ for every $i \in [\ell_{\mathsf{ct}}]$.
6. For every $i \in [\ell_{\mathsf{ct}}]$, apply the map
$$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \rightarrow |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_i}}$$
to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ and obtain the resulting state ρ_i.
7. Output $\mathcal{sk}_y = (\rho_i)_{i \in [\ell_{\mathsf{ct}}]}$, $\mathsf{vk} = (x, \theta, S)$, and $\mathsf{tk} = T$.

$\mathsf{SKFE\text{-}CR\text{-}SKL.Enc}(\mathsf{msk}, x)$**:**

1. Parse $\mathsf{msk} = (\mathsf{skecd.sk}, \mathsf{skfe.msk})$.
2. Generate $\mathsf{skfe.ct} \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, x)$.
3. Output $\mathsf{ct} := (\mathsf{skecd.sk}, \mathsf{skfe.ct})$.

$\mathsf{SKFE\text{-}CR\text{-}SKL.CDec}(\mathsf{sk}, \mathsf{ct})$**:**

1. Parse $\mathsf{ct} = (\mathsf{skecd.sk}, \mathsf{skfe.ct})$. Parse sk as a string over the registers $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ and $\mathsf{S} = \mathsf{S}_1 \otimes \cdots \otimes \mathsf{S}_{\ell_{\mathsf{ct}}}$. Let $\widetilde{\mathsf{sk}}$ be the sub-string of sk on register $\mathsf{SKECD.CT}$.
2. Compute $\mathsf{skfe.sk} \leftarrow \mathsf{SKECD.CDec}(\mathsf{skecd.sk}, \widetilde{\mathsf{sk}})$.
3. Output $z \leftarrow \mathsf{SKFE.Dec}(\mathsf{skfe.sk}, \mathsf{skfe.ct})$.

$\mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{Dec}(\mathcal{sk}, \mathsf{ct})$**:**

1. Parse $(\rho_i)_{i \in [\ell_{\mathsf{ct}}]}$. We denote the register holding ρ_i as $\mathsf{SKECD.CT_i} \otimes \mathsf{S_i}$ for every $i \in [\ell_{\mathsf{ct}}]$.
2. Prepare a register MSG of ℓ_{m} qubits that is initialized to $|0 \cdots 0\rangle_{\mathsf{MSG}}$.
3. Apply the map
$$|u\rangle_{\bigotimes_{i \in [\ell_{\mathsf{ct}}]} \mathsf{SKECD.CT_i}} \otimes |w\rangle_{\mathsf{MSG}} \rightarrow |u\rangle_{\bigotimes_{i \in [\ell_{\mathsf{ct}}]} \mathsf{SKECD.CT_i}} \otimes |w \oplus \mathsf{SKFE\text{-}CR\text{-}SKL.CDec}(u, \mathsf{ct})\rangle_{\mathsf{MSG}}$$
to the registers $\bigotimes_{i \in [\ell_{\mathsf{ct}}]} \mathsf{SKECD.CT_i}$ and MSG.
4. Measure MSG in the computational basis and output the result m'.

$\mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{Vrfy}(\mathsf{vk}, \mathcal{sk})$**:**

1. Parse $\mathsf{vk} = (x, \theta, S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_{\mathsf{ct}}] : \theta[i]=1})$ and $\mathcal{sk} = (\rho_i)_{i \in [\ell_{\mathsf{ct}}]}$ where ρ_i is a state on the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$.
2. For every $i \in [\ell_{\mathsf{ct}}]$, measure ρ_i in the Hadamard basis to get outcomes c_i, d_i corresponding to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ respectively.
3. Output $\top$ if $x[i] = c_i \oplus d_i \cdot (s_{i,0} \oplus s_{i,1})$ holds for every $i \in [\ell_{\mathsf{ct}}]$ such that $\theta[i] = 1$. Otherwise, output $\bot$.

$\mathsf{SKFE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{tk}, \mathsf{sk})$:

1. Parse sk as a string over the registers $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ and $\mathsf{S} = \mathsf{S}_1 \otimes \cdots \otimes \mathsf{S}_{\ell_{\mathsf{ct}}}$. Let u_i denote the value on $\mathsf{SKECD.CT}_i$ and v_i the value on S_i. Parse tk as $T = t_{1,0} \| t_{1,1} \| \cdots \| t_{\ell_{\mathsf{ct}},0} \| t_{\ell_{\mathsf{ct}},1}$.
2. Let $\mathsf{Check}[t_{i,0}, t_{i_1}](u_i, v_i)$ be the deterministic algorithm that outputs 1 if $f(v_i) = t_{i,u_i}$ holds and 0 otherwise.
3. Output $\mathsf{Check}[t_{1,0}, t_{1,1}](u_1, v_1) \wedge \mathsf{Check}[t_{2,0}, t_{2,1}](u_2, v_2) \wedge \cdots \wedge \mathsf{Check}[t_{\ell_{\mathsf{ct}},0}, t_{\ell_{\mathsf{ct}},1}](u_{\ell_{\mathsf{ct}}}, v_{\ell_{\mathsf{ct}}})$.

C.2 Proof of Selective Single-Ciphertext KLA Security

Let $\mathcal{A}$ be an adversary for the selective single-ciphertext KLA security of the construction $\mathsf{SKFE\text{-}CR\text{-}SKL}$ that makes use of a BB84-based SKE-CD scheme SKECD. Consider the hybrid $\mathsf{Hyb}_j^{\mathsf{coin}}$ defined as follows:

$\mathsf{Hyb}_j^{\mathsf{coin}}$:

1. Initialized with 1^λ, $\mathcal{A}$ outputs (x_0^*, x_1^*). Sample $\mathsf{msk} \leftarrow \mathsf{SKFE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$.
2. $\mathcal{A}$ can get access to the following (stateful) oracles, where the list $L_{\mathcal{KG}}$ used by the oracles is initialized to an empty list:

   $O_{\mathcal{KG}}(y)$: Given y, it finds an entry of the form (y, vk, V) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\perp$. Otherwise it proceeds as follows:

   (i) If this is the k-th query for $k \leq j$ and $F(x_0^*, y) \neq F(x_1^*, y)$, then compute $(\mathit{sk}, \mathsf{vk}, \mathsf{tk}) \leftarrow \widetilde{\mathcal{KG}}(\mathsf{msk}, y)$ where $\widetilde{\mathcal{KG}}$ is defined below. Otherwise, compute $(\mathit{sk}, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y)$.

   (ii) It sends sk and tk to $\mathcal{A}$ and adds $(y, \mathsf{vk}, \perp)$ to $L_{\mathcal{KG}}$.

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

   $\widetilde{\mathcal{KG}}(\mathsf{msk})$: Differences from $\mathcal{KG}$ are colored in red:

   (a) Parse $\mathsf{msk} = (\mathsf{skecd.sk}, \mathsf{skfe.msk})$.

   (b) Generate $r \leftarrow \{0,1\}^\lambda$.

   (c) Generate $(\mathsf{skecd}.\mathit{ct}, \mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, r)$. $\mathsf{skecd.vk}$ is of the form $(x, \theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, and $\mathsf{skecd}.\mathit{ct}$ is of the form $|\psi_1\rangle_{\mathsf{SKECD.CT}_1} \otimes \cdots \otimes |\psi_{\ell_{\mathsf{ct}}}\rangle_{\mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}}$.

   (d) Generate $s_{i,b} \leftarrow \{0,1\}^\lambda$ and compute $t_{i,b} \leftarrow f(s_{i,b})$ for every $i \in [\ell_{\mathsf{ct}}]$ and $b \in \{0,1\}$. Set $T := t_{1,0} \| t_{1,1} \| \cdots \| t_{\ell_{\mathsf{ct}},0} \| t_{\ell_{\mathsf{ct}},1}$ and $S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_{\mathsf{ct}}] \,:\, \theta[i]=1}$.

   (e) Prepare a register S_i that is initialized to $|0^\lambda\rangle_{\mathsf{S}_i}$ for every $i \in [\ell_{\mathsf{ct}}]$.

   (f) For every $i \in [\ell_{\mathsf{ct}}]$, apply the map

   $$|u_i\rangle_{\mathsf{SKECD.CT}_i} \otimes |v_i\rangle_{\mathsf{S}_i} \to |u_i\rangle_{\mathsf{SKECD.CT}_i} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S}_i}$$

   to the registers $\mathsf{SKECD.CT}_i$ and S_i and obtain the resulting state ρ_i.

   (g) Output $\mathit{sk}_y = (\rho_i)_{i \in [\ell_{\mathsf{ct}}]}$, $\mathsf{vk} = (x, \theta, S)$, and $\mathsf{tk} = T$.
3. If there exists an entry (y, vk, V) in $L_{\mathcal{KG}}$ such that $F(x_0^*, y) \neq F(x_1^*, y)$ and $V = \perp$, output 0. Otherwise, generate $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{msk}, x_{\mathsf{coin}}^*)$ and send ct^* to $\mathcal{A}$.
4. $\mathcal{A}$ continues to make queries to $O_{\mathcal{KG}}$. However, $\mathcal{A}$ is not allowed to send y such that $F(x_0^*, y) \neq F(x_1^*, y)$ to $O_{\mathcal{KG}}$.
5. $\mathcal{A}$ outputs a guess coin' for coin. Output coin'.

Let $\mathcal{A}$ make $q = \mathrm{poly}(\lambda)$ many queries to $O_{\mathcal{KG}}$ before the challenge phase. We will now prove the following lemma:

Lemma C.1. $\forall j \in \{0,\ldots,q-1\}$ *and* $\mathsf{coin} \in \{0,1\}$: $\mathsf{Hyb}_j^{\mathsf{coin}} \approx_c \mathsf{Hyb}_{j+1}^{\mathsf{coin}}$.

Proof. Suppose $\mathsf{Hyb}_j^{\mathsf{coin}} \not\approx \mathsf{Hyb}_{j+1}^{\mathsf{coin}}$. Let $\mathcal{D}$ be a corresponding distinguisher. We will construct a reduction $\mathcal{R}$ that breaks the IND-CVA-CD security of the BB84-based SKE-CD scheme SKECD. The execution of $\mathcal{R}^{\mathcal{D}}$ in the experiment $\mathsf{Exp}_{\mathsf{SKECD},\mathcal{R}}^{\mathsf{ind\text{-}cva\text{-}cd}}(1^\lambda, b)$ proceeds as follows:

Execution of $\mathcal{R}^{\mathcal{D}}$ in $\mathsf{Exp}_{\mathsf{SKECD},\mathcal{R}}^{\mathsf{ind\text{-}cva\text{-}cd}}(1^\lambda, b)$:

1. The challenger $\mathcal{Ch}$ computes $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$.
2. $\mathcal{R}$ initializes $\mathcal{D}$ with 1^λ which outputs (x_0^*, x_1^*).
3. $\mathcal{R}$ samples $\mathsf{skfe.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$. It initializes a list $L_{\mathcal{KG}}$ to an empty list.
4. $\mathcal{R}$ simulates the oracle $O_{\mathcal{KG}}$ for $\mathcal{D}$ as follows:

   $O_{\mathcal{KG}}(y)$: Given y, it finds an entry of the form (y, vk, V) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\bot$. Otherwise, it proceeds as follows for the k-th query:

   (a) If $k \le j$ and $F(x_0^*, y) \neq F(x_1^*, y)$, then compute $(\mathcal{sk}, \mathsf{vk}, \mathsf{tk})$ as in $\widetilde{\mathcal{KG}}$ except that the pair $(\mathsf{skecd}.ct, \mathsf{skecd.vk})$ is obtained as the output of $O_{\mathcal{Enc}}(r)$ instead. Send $\mathcal{sk}$ and tk to $\mathcal{A}$ and add $(y, \mathsf{vk}, \bot)$ to $L_{\mathcal{KG}}$.

   (b) If $k = j+1$ and $F(x_0^*, y) \neq F(x_1^*, y)$, compute $\mathsf{skfe.sk}_y \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, y)$. Then, send $(r^*, \mathsf{skfe.sk}_y)$ to $\mathcal{Ch}$ where r^* is a random value. On receiving $\mathsf{skecd.ct}^*$ from $\mathcal{Ch}$, compute $\mathcal{sk}$ and tk as in $\mathcal{KG}$, except that $\mathsf{skecd.ct}^*$ is used in place of $\mathsf{skecd.ct}$. Send $\mathcal{sk}$ and tk to $\mathcal{A}$ and add $(y, 0^\lambda, \bot)$ to $L_{\mathcal{KG}}$.

   (c) If $k > j+1$ or $F(x_0^*, y) = F(x_1^*, y)$, then compute $(\mathcal{sk}, \mathsf{vk}, \mathsf{tk})$ as in $\mathcal{KG}$ except that the pair $(\mathsf{skecd}.ct, \mathsf{skecd.vk})$ is replaced with the output of $O_{\mathsf{Enc}}(\mathsf{skfe.sk}_y)$ instead. Send $\mathcal{sk}$ and tk to $\mathcal{A}$ and add $(y, \mathsf{vk}, \bot)$ to $L_{\mathcal{KG}}$.
5. $\mathcal{R}$ simulates the oracle $O_{\mathcal{Vrfy}}$ for $\mathcal{D}$ as follows:

   $O_{\mathcal{Vrfy}}(y, \widetilde{\mathcal{sk}})$: Given $(y, \widetilde{\mathcal{sk}})$, it finds an entry (y, vk, V) from $L_{\mathcal{KG}}$ (If there is no such entry, it returns $\bot$.) It then proceeds as follows, if y corresponds to the k-th query made to $O_{\mathcal{KG}}$ and $k \neq j+1$:

   (a) Parse $\mathsf{vk} = (x, \theta, S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_{\mathsf{ct}}] : \theta[i]=1})$ and $\widetilde{\mathcal{sk}} = (\rho_i)_{i \in [\ell_{\mathsf{ct}}]}$.

   (b) For every $i \in [\ell_{\mathsf{ct}}]$, measure ρ_i in the Hadamard basis to get outcomes c_i, d_i corresponding to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ respectively.

   (c) Compute $\mathsf{cert}[i] = c_i \oplus d_i \cdot (s_{i,0} \oplus s_{i,1})$ for every $i \in [\ell_{\mathsf{ct}}]$.

   (d) If $x[i] = \mathsf{cert}[i]$ holds for every $i \in [\ell_{\mathsf{ct}}] : \theta[i] = 1$, then update the entry to $(y, \mathsf{vk}, \top)$ and send $\top$ to $\mathcal{D}$. Else, send $\bot$.

   If $k = j+1$, then it proceeds as follows:

   (a) Compute $\mathsf{cert} = \mathsf{cert}[1] \| \ldots \| \mathsf{cert}[\ell_{\mathsf{ct}}]$ where each $\mathsf{cert}[i]$ is computed as in the previous case.

   (b) Send cert to $\mathcal{Ch}$. If $\mathcal{Ch}$ returns $\mathsf{skecd.sk}$, send $\top$ to $\mathcal{D}$ and update the corresponding entry to $(y, \mathsf{vk}, \top)$. Else, output $\bot$.
6. $\mathcal{D}$ requests the challenge ciphertext. If there exists an entry (y, vk, V) in $L_{\mathcal{KG}}$ such that $F(x_0^*, y) \neq F(x_1^*, y)$ and $V = \bot$, output 0. Otherwise, compute and send $\mathsf{ct}^\star = (\mathsf{skecd.sk}, \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, x_{\mathsf{coin}}^*))$ to $\mathcal{D}$.
7. $\mathcal{D}$ continues to make queries to $O_{\mathcal{KG}}$. However, $\mathcal{A}$ is not allowed to send y such that $F(x_0^*, y) \neq F(x_1^*, y)$ to $O_{\mathcal{KG}}$. Consequently, $\mathcal{R}$ simulates these queries as per Step 4. (c) above.
8. $\mathcal{D}$ outputs a guess coin' for coin which $\mathcal{R}$ forwards to $\mathcal{Ch}$. $\mathcal{Ch}$ outputs coin' as the output of the experiment.

We will first argue that when $b = 1$, the view of $\mathcal{D}$ is exactly the same as its view in the hybrid $\mathsf{Hyb}_j^{\mathsf{coin}}$. Notice that the for the first j queries, if $F(x_0^*, y) \neq F(x_1^*, y)$ for a key-query corresponding to y, then the decryption key is computed by querying the encryption oracle on a random plaintext. If this condition does not hold, then the key is computed by querying the encryption oracle on the corresponding SKFE key $\mathsf{skfe.sk}_y$. The hybrid Hyb_j on the other hand, directly computes these values, but there is no difference in the distribution of the output ciphertexts. A similar argument holds for the keys $\mathcal{dk}_{j+2}, \ldots, \mathcal{dk}_q$, which contain encryptions of the corresponding keys $\mathsf{skfe.sk}_y$. Note that if $F(x_0^*, y^*) = F(x_1^*, y^*)$, where y^* corresponds to the $j+1$-th query, then the hybrids $\mathsf{Hyb}_j^{\mathsf{coin}}$ and $\mathsf{Hyb}_{j+1}^{\mathsf{coin}}$ are identical. Hence, consider the case when $F(x_0^*, y^*) \neq F(x_1^*, y^*)$. In this case, if $b = 1$, the value encrypted as part of the key $\mathcal{sk}_{j+1}$ is $\mathsf{skfe.sk}_{y^*}$. This is the same as in $\mathsf{Hyb}_j^{\mathsf{coin}}$. As for the verification oracle queries, notice that they are answered similarly by the reduction and $\mathsf{Hyb}_j^{\mathsf{coin}}$ for all but the $j+1$-th key. For the $j+1$-th key, the reduction works differently in that it forwards the certificate cert to the verification oracle. However, the verification procedure of the BB84-based SKE-CD scheme checks the validity of the value cert in the same way as the reduction, so there is no difference. Finally, notice that when $b = 0$, the encrypted value is an independent and random value, similar to the hybrid $\mathsf{Hyb}_{j+1}^{\mathsf{coin}}$. Consequently, $\mathcal{R}$ breaks the IND-CVA-CD security of SKECD with non-negligible probability, a contradiction. $\square$

Notice now that the hybrid $\mathsf{Hyb}_0^{\mathsf{coin}}$ is the same as the experiment $\mathsf{Exp}_{\mathsf{SKE\text{-}CR\text{-}SKL},\mathcal{A}}^{\mathsf{sel\text{-}1ct\text{-}kla}}(1^\lambda, \mathsf{coin})$. From the previous lemma, we have that $\mathsf{Hyb}_0^{\mathsf{coin}} \approx_c \mathsf{Hyb}_q^{\mathsf{coin}}$. However, we have that $\mathsf{Hyb}_q^0 \approx_c \mathsf{Hyb}_q^1$ holds from the selective single-ciphertext security of the underlying SKFE scheme SKFE. This is because any key-query corresponding to y after the q-th query is such that $F(x_0^*, y) = F(x_1^*, y)$. For the first q queries, wherever this condition doesn't hold, the SKFE keys have been replaced with random values. Consequently, we have that $\mathsf{Hyb}_0^0 \approx_c \mathsf{Hyb}_0^1$, which completes the proof. $\square$

C.3 Proof of Key-Testability

First, we will argue the correctness requirement. Recall that $\mathsf{SKFE\text{-}CR\text{-}SKL}.\mathcal{KG}$ applies the following map to a BB84 state $|x\rangle_\theta$, where $(x, \theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, for every $i \in [\ell_{\mathsf{ct}}]$:

$$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \to |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_i}}$$

where $\mathsf{SKECD.CT_i}$ denotes the register holding the i-th qubit of $|x\rangle_\theta$ and $\mathsf{S_i}$ is a register initialized to $|0\ldots0\rangle_{\mathsf{S_i}}$.

Consider applying the algorithm $\mathsf{SKFE\text{-}CR\text{-}SKL.KeyTest}$ in superposition to the resulting state, i.e., performing the following map, where $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ and $\mathsf{S} = \mathsf{S}_1 \otimes \cdots \otimes \mathsf{S}_{\ell_{\mathsf{ct}}}$, and KT is initialized to $|0\rangle$:

$$|u\rangle_{\mathsf{SKECD.CT}} \otimes |v\rangle_{\mathsf{S}} \otimes |\beta\rangle_{\mathsf{KT}} \to |u\rangle_{\mathsf{SKECD.CT}} \otimes |v\rangle_{\mathsf{S}} \otimes |\beta \oplus \mathsf{SKFE\text{-}CR\text{-}SKL.KeyTest}(\mathsf{tk}, u\|v)\rangle_{\mathsf{KT}}$$

where $\mathsf{tk} = T = t_{1,0}\|t_{1,1}\| \cdots \|t_{\ell_{\mathsf{ct}},0}\|t_{\ell_{\mathsf{ct}},1}$. Recall that $\mathsf{SKFE\text{-}CR\text{-}SKL.KeyTest}$ outputs 1 if and only if $\mathsf{Check}[t_{i,0}, t_{i,1}](u_i, v_i) = 1$ for every $i \in [\ell_{\mathsf{ct}}]$, where u_i, v_i denote the states of the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ respectively. Recall that $\mathsf{Check}[t_{i,0}, t_{i,1}](u_i, v_i)$ computes $f(v_i)$ and checks if it equals t_{i,u_i}. Since the construction chooses t_{i,u_i} such that $f(s_{i,u_i}) = t_{i,u_i}$, this check always passes. Consequently, measuring register KT always produces outcome 1.

We will now argue that the security requirement holds by showing the following reduction to the security of the OWF f. Let $\mathcal{A}$ be an adversary that breaks the key-testability of $\mathsf{SKFE\text{-}CR\text{-}SKL}$. Consider a QPT reduction $\mathcal{R}$ that works as follows in the OWF security experiment:

Execution of $\mathcal{R}^{\mathcal{A}}$ in $\mathsf{Expt}_{f,\mathcal{R}}^{\mathsf{owf}}(1^\lambda)$:

1. The challenger chooses $s \leftarrow \{0,1\}^\lambda$ and sends $y^* = f(s)$ to $\mathcal{R}$.
2. $\mathcal{R}$ runs $\mathsf{SKFE\text{-}CR\text{-}SKL.Setup}(1^\lambda)$ and initializes $\mathcal{A}$ with input msk.
3. $\mathcal{R}$ picks a random $k^* \in [q]$.
4. $\mathcal{R}$ simulates the access of $\mathcal{A}$ to the oracle $O_{\mathcal{KG}}$ as follows, where the list $L_{\mathcal{KG}}$ is initialized to an empty list:

   $O_{\mathcal{KG}}(y)$**:** For the k-th query, do the following:

(a) Given y, it finds an entry of the form (y, tk) from $L_{\mathcal{KG}}$. If there is such an entry, it returns $\perp$.

(b) Otherwise, if $k \neq k^*$, it generates $(\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk}) \leftarrow \mathcal{KG}(\mathsf{msk}, y)$, sends $(\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk})$ to $\mathcal{A}$, and adds (y, tk) to $L_{\mathcal{KG}}$.

(c) Otherwise, if $k = k^*$, it generates $(\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk}) \leftarrow \widetilde{\mathcal{KG}}(\mathsf{msk}, y)$ (differences from $\mathcal{KG}$ are colored in red). It then sends $(\mathcal{sk}_y, \mathsf{vk}, \mathsf{tk})$ to $\mathcal{A}$ and adds (y, tk) to $L_{\mathcal{KG}}$.

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

(a) Parse $\mathsf{msk} = (\mathsf{skecd.sk}, \mathsf{skfe.msk})$.

(b) Generate $\mathsf{skfe.sk}_y \leftarrow \mathsf{SKFE.KG}(\mathsf{skfe.msk}, y)$.

(c) Generate $(\mathsf{skecd}.ct, \mathsf{skecd.vk}) \leftarrow \mathsf{SKECD}.\mathcal{Enc}(\mathsf{skecd.sk}, \mathsf{skfe.sk}_y)$. Here, $\mathsf{skecd.vk}$ is of the form $(x, \theta) \in \{0,1\}^{\ell_{\mathsf{ct}}} \times \{0,1\}^{\ell_{\mathsf{ct}}}$, and $\mathsf{skecd}.ct$ is of the form $|\psi_1\rangle_{\mathsf{SKECD.CT}_1} \otimes \cdots \otimes |\psi_{\ell_{\mathsf{ct}}}\rangle_{\mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}}$.

(d) Choose an index $i^\star \in [\ell_{\mathsf{ct}}]$ such that $\theta[i^\star] = 0$. For every $i \in [\ell_{\mathsf{ct}}]$ such that $i \neq i^\star$, generate $s_{i,b} \leftarrow \{0,1\}^\lambda$ and compute $t_{i,b} \leftarrow f(s_{i,b})$ for every $b \in \{0,1\}$. For $i = i^\star$, set $t_{i^\star,1-x[i^\star]} = y^*$. Then, generate $s_{i^\star,x[i^\star]} \leftarrow \{0,1\}^\lambda$ and compute $t_{i^\star,x[i^\star]} = f(s_{i^\star,x[i^\star]})$. Set $T := t_{1,0} \| t_{1,1} \| \cdots \| t_{\ell_{\mathsf{ct}},0} \| t_{\ell_{\mathsf{ct}},1}$ and $S = \{s_{i,0} \oplus s_{i,1}\}_{i \in [\ell_{\mathsf{ct}}] : \theta[i]=1}$.

(e) Prepare a register $\mathsf{S_i}$ that is initialized to $|0^\lambda\rangle_{\mathsf{S_i}}$ for every $i \in [\ell_{\mathsf{ct}}]$.

(f) For every $i \in [\ell_{\mathsf{ct}}]$, apply the map

$$|u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i\rangle_{\mathsf{S_i}} \to |u_i\rangle_{\mathsf{SKECD.CT_i}} \otimes |v_i \oplus s_{i,u_i}\rangle_{\mathsf{S_i}}$$

to the registers $\mathsf{SKECD.CT_i}$ and $\mathsf{S_i}$ and obtain the resulting state ρ_i.

(g) Output $\mathcal{sk}_y = (\rho_i)_{i \in [\ell_{\mathsf{ct}}]}$, $\mathsf{vk} = (x, \theta, S)$, and $\mathsf{tk} = T$.

5. $\mathcal{A}$ sends a tuple of classical strings (y, sk, x^*) to $\mathcal{R}$. $\mathcal{R}$ outputs $\perp$ if there is no entry of the form (y, tk) in $L_{\mathcal{KG}}$ for some tk. Also, if $k \neq k^*$, $\mathcal{R}$ outputs $\perp$. Otherwise, $\mathcal{R}$ parses sk as a string over the registers $\mathsf{SKECD.CT} = \mathsf{SKECD.CT}_1 \otimes \cdots \otimes \mathsf{SKECD.CT}_{\ell_{\mathsf{ct}}}$ and $\mathsf{S} = \mathsf{S}_1 \otimes \cdots \otimes \mathsf{S}_{\ell_{\mathsf{ct}}}$ and measures the register $\mathsf{S}_{i^\star}$ to obtain an outcome $s_{i^\star}$. $\mathcal{R}$ then sends $s_{i^\star}$ to the challenger.

Notice that the view of $\mathcal{A}$ is the same as its view in the key-testability experiment, as only the value $t_{i^\star,1-x[i^\star]}$ is generated differently by forwarding the value y, but this value is distributed identically to the original value. Note that in both cases, $\mathcal{A}$ receives no information about a pre-image of $t_{i^\star,1-x[i^\star]}$. Now, $\mathcal{R}$ guesses the index k that $\mathcal{A}$ targets with probability $\frac{1}{q}$. By assumption, we have that $\mathsf{CDec}(\mathsf{sk}, \mathsf{ct}) \neq F(x^*, y)$ where $\mathsf{ct} = \mathsf{Enc}(\mathsf{msk}, x^*)$. The value sk can be parsed as a string over the registers $\mathsf{SKECD.CT}$ and S. Let $\widetilde{\mathsf{sk}}$ be the sub-string of sk on the register $\mathsf{SKECD.CT}$. Recall that CDec invokes the algorithm $\mathsf{SKECD.CDec}$ on input $\widetilde{\mathsf{sk}}$. We will now recall a property of $\mathsf{SKECD.CDec}$ that was specified in Definition 3.12:

Let $(ct, \mathsf{vk} = (x, \theta)) \leftarrow \mathsf{SKECD.Enc}(\mathsf{skecd.sk}, \mathsf{skfe.sk}_y)$ where $\mathsf{skecd.sk} \leftarrow \mathsf{SKECD.KG}(1^\lambda)$. Now, let u be any arbitrary value such that $u[i] = x[i]$ for all $i : \theta[i] = 0$. Then, the following holds:

$$\Pr\left[\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, u) = \mathsf{skfe.sk}_y\right] \geq 1 - \mathsf{negl}(\lambda)$$

Consequently, if $\widetilde{\mathsf{sk}}$ is such that $\widetilde{\mathsf{sk}}[i] = x[i]$ for all $i : \theta[i] = 0$, where (x, θ) are specified by $\mathsf{vk}_{k^\star}$, then $\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, \widetilde{\mathsf{sk}})$ outputs the value $\mathsf{skfe.sk}_y$ with high probability. Since $\mathsf{CDec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{skecd.sk}, \mathsf{skfe.ct} = \mathsf{SKFE.Enc}(\mathsf{skfe.msk}, x^*)))$ outputs $\mathsf{SKFE.Dec}(\mathsf{SKECD.CDec}(\mathsf{skecd.sk}, \widetilde{\mathsf{sk}}), \mathsf{skfe.ct})$, we have that it outputs x^* with high probability from the decryption correctness of SKFE. Therefore, it must be the case that there exists some index i for which $\widetilde{\mathsf{sk}} \neq x[i]$. With probability $\frac{1}{\ell_{\mathsf{ct}}}$, this happens to be the guessed value $i^\star$. In this case, $\mathcal{A}$ must output $s_{i^\star}$ on register $\mathsf{S_i}$ such that $f(s_{i^\star}) = t_{i^\star,1-x[i^\star]} = y^*$. This concludes the proof. □

Since we have proved selective single-ciphertext security and key-testability, we can now state the following theorem:

Theorem C.2. *Assuming the existence of a BB84-based SKE-CD scheme and the existence of OWFs, there exists a selective single-ciphertext KLA secure SKFE-CR-SKL scheme satisfying the key-testability property.*