A Unified Approach to Quantum Key Leasing with a Classical Lessor

Fuyuki Kitagawa[1,2], Jiahui Liu[3], Shota Yamada[4], and Takashi Yamakawa[1,2]

[1]NTT Social Informatics Laboratories, Japan
{fuyuki.kitagawa,takashi.yamakawa}@ntt.com
[2]NTT Research Center for Theoretical Quantum Information, Japan
[3]Fujitsu Research of America, USA
jiahuiliu.crypto@gmail.com
[4]AIST, Japan
yamada-shota@aist.go.jp

July 21, 2026

Abstract

Secure key leasing allows a cryptographic key to be leased as a quantum state in such a way that the key can later be revoked in a verifiable manner. In this work, we propose a modular framework for constructing secure key leasing with a classical-lessor, where the lessor is entirely classical and, in particular, the quantum secret key can be both leased and revoked using only classical communication. Based on this framework, we obtain classical-lessor secure key leasing schemes for public-key encryption (PKE), pseudorandom function (PRF), and digital signature. We adopt the strong security notion known as security against verification key revealing attacks (VRA security) proposed by Kitagawa et al. (Eurocrypt 2025) into the classical-lessor setting, and we prove that all three of our schemes satisfy this notion under the learning with errors assumption. Our PKE scheme improves upon the previous construction by Chardouvelis et al. (Eurocrypt 2025), and our PRF and digital signature schemes are respectively the first PRF and digital signature with classical-lessor secure key leasing property.

Along the way, we also construct a watermarking scheme and a dual-mode secure function evaluation scheme that satisfy certain useful properties, which may be of independent interest.

Contents

1 Introduction **3**
- 1.1 Our Results . . . 3
- 1.2 Related Works . . . 4
- 1.3 Concurrent Work . . . 5
- 1.4 Technical Overview . . . 6
- 1.5 Discussion . . . 14

2 Preliminaries **15**
- 2.1 Cryptographic Primitives . . . 15
- 2.2 Noisy Trapdoor Claw-Free Generators . . . 18

3 Watermarkable Cryptographic Functionalities **20**
- 3.1 Watermarkable Public Key Encryption . . . 20
- 3.2 Watermarkable Unpredictable Functions . . . 23
- 3.3 Watermarkable Digital Signatures . . . 30

4 Special Dual-Mode Secure Function Evaluation **37**

5 Definitions of Secure Key Leasing with Classical Lessor **39**
- 5.1 Public Key Encryption with Secure Key Leasing . . . 39
- 5.2 Pseudorandom and Unpredictable Functions with Secure Key Leasing . . . 42
- 5.3 Digital Signatures with Secure Key Leasing . . . 44

6 PKE-SKL with Classical Lessor **45**
- 6.1 Construction with Interactive Key Generation . . . 45
- 6.2 Security Proof . . . 50
- 6.3 Construction with Non-Interactive Key Generation . . . 55

7 PRF-SKL with Classical Lessor **56**
- 7.1 Construction . . . 56
- 7.2 Security Proof . . . 59

8 DS-SKL with Classical Lessor **64**
- 8.1 Construction . . . 64
- 8.2 Security Proof . . . 67

A Special Dual-Mode SFE from LWE **74**
- A.1 Definition of Special Dual-mode OT . . . 74
- A.2 Dual-mode OT from LWE . . . 75
- A.3 From OT to SFE via Garbled Circuit . . . 79

1 Introduction

Delegation and Revocation of Cryptographic Functionalities Delegation of computation is a central theme in cryptography and the practice of cloud computing. Beyond efficiency and correctness, many real-world scenarios demand revocability: a client who temporarily outsources a capability must be able to withdraw it cleanly, without incurring operational hazards or reconfiguring public parameters. Consider a manager who leases signing capability to a delegate while on vacation; upon return, she must revoke the delegatee's signing power. The classical solution to this problem requires not only updating the secret signing key, but also updating and re-announcing public verification parameters. This procedure can be costly and can create vulnerability during propagation. This motivates us to look for a cryptographic mechanism that can revoke functionality without updating public parameters or incurring propagation delays

A formalization of the above protocol is called secure key leasing (also known as key-revocable cryptography), first put forward by [AKN+23, APV23]: a lessor issues a quantum secret key that enables some functionality (e.g., decryption, PRF evaluation, or signing) to a lessee (server), and later executes a deletion protocol: the lessee (server) must return the key or a proof of deleting the key; this deletion protocol is supposed to provably strip the lessee of the capability of computing the functionality. Naturally, secure key leasing (SKL) also captures temporary licensing for general programs apart from delegation of computing: a software lessor leases quantum software whose functionality can later be invalidated upon successful return and verification, for instance, when the lessee's subscription to the software has expired. Furthermore, a secure leasing scheme for a cryptographic function can potentially lead to a secure leasing scheme for a software entity of which this cryptographic function is a component.

However, this goal is classically impossible as the lessee/server can always make a copy of the key or software. As shown in a sequence of previous works, leveraging the unclonability of quantum information will help us construct protocols that satisfy all the above requirements.

Importance of a Classical Client/Lessor While we need to utilize the property of quantum information to realize the secure key leasing protocol, minimizing the cost and quantum resources used is also indispensable. Especially in the near-term quantum computing era and practical deployments, it is crucial to minimize the required quantum computing resources on the client's side. For instance, in real-world scenarios of cloud computing, the client will ideally use much less computing resources than the server uses when interacting with the server, otherwise the task of delegation is not very meaningful. When designing a protocol for cloud quantum computing or software leasing, the lessor (client) ideally *remains entirely classical and communicates over classical channels* throughout the protocol, while the lessee (server) performs the quantum work locally on its end.

Towards a Modular Approach A sequence of works has realized secure key leasing (SKL) for several functionalities—PKE, FHE, PRFs, and digital signatures—with the most recent results based on standard cryptographic assumptions [CGJL25, AHH24, KMY25, KNP25]. However, these constructions are largely ad hoc and tailored to each functionality rather than arising from a common, modular blueprint. Moreover, the only known scheme with a fully classical lessor (client) in [CGJL25] applies to PKE and FHE; their underlying approach seems relatively tailored to encryption schemes and is not known to generalize to other primitives. [KMY25] provides a relatively unified approach to construct SKL protocols for several cryptographic functionalities, but the lessor still needs to be fully quantum and there does not seem to be a straightforward way to dequantize the lessor.

Motivated by these limitations, we ask the following question:

Can we develop a modular SKL framework that supports a classical lessor and spans a broad range of widely used cryptographic functionalities?

1.1 Our Results

Main Result: Classical-Lessor SKL Framework We first state our main result, which answers the above question in a positive way.

We propose a simple framework for constructing secure key leasing schemes with a fully classical lessor. Based on our framework, we obtain the following schemes for "upgrading" some very generic cryptographic primitives to their corresponding classical-lessor SKL schemes, using only the post-quantum security of standard lattice assumptions.

- PKE-SKL with classical lessor based on (sub-exponential) LWE. The underlying PKE scheme is, however, *not* limited to any specific LWE-based construction. More concretely, we can "upgrade" *any post-quantum IND-CPA secure classical PKE scheme* to a PKE-SKL with classical lessor, using LWE.

  While we still need the LWE assumption, we use the other two primitives in our construction, the noisy trapdoor claw-free families (NTCFs) and secure function evaluation (SFE) also *in a black-box way* (see Theorems 6.2, 7.2, 8.2.). They have potentially non-LWE-based constructions[1]. Therefore, it is meaningful to modify the highly LWE-structured PKE-SKL protocol in [CGJL25, PWYZ24] into a more modular protocol.

  Moreover, it is unlikely that we can obtain PKE-SKL with a classical lessor from any PKE alone without reliance on any other structures, otherwise we will obtain a classically verifiable quantum advantage from any PKE ([KMY25]).

- PRF-SKL with classical lessor from LWE. The underlying PRF scheme used can be any post-quantum watermarkable PRF, which we upgrade to a classical lessor PRF-SKL with LWE.

- DS-SKL with classical lessor based on LWE. For digital signatures, our construction is less black-box. We lift a digital signature to a classical-lessor DS-SKL through the same framework as we apply to the PKE and PRF above. But we need a special property of the underlying digital signature scheme, which we can obtain from LWE.

Finally, all of our schemes remain secure even if a deletion verification key used by the deletion verification algorithm is leaked after the adversary submits a valid certificate of deletion. This security is called security against verification key revealing attack (VRA security), as proposed in [KMY25].

For a comparison with existing results on the aforementioned properties of SKL, we present Tables 1,2,3.

Non-interactive key generation We further improve our classical-lessor SKL scheme to acquire a non-interactive quantum key generation algorithm (whereas the vanilla protocol requires a constant round interaction) through a clean technique. This is a non-trivial property for SKL with a fully classical lessor. While this property is easy to achieve for a quantum lessor SKL–simply letting the lessor prepare the quantum key and send it to the lessee, a classical lessor can only send classical information to the lessee and may need interactive messages from the lessee to prevent its malicious behavior during key generation.[2]

Side results Along the way of constructing our classical-lessor SKL framework, we construct a special dual-mode Secure Function Evaluation (SFE) scheme and a watermarking scheme with parallel extraction. They satisfy many useful properties, which we believe can be of independent interest.

1.2 Related Works

More on SKL The work of [AKN+23] realized the first construction of PKE-SKL from any PKE scheme. They also realized unbounded (resp. bounded) collusion-resistant PKFE-SKL by combining PKE-SKL with unbounded (resp. bounded) collusion-resistant classical PKFE. Independently, [APV23] proposed constructions of PKE-SKL, FHE-SKL, and PRF-SKL under the LWE assumption, though their security proofs relied on an additional unproven conjecture. This reliance on unproven conjecture was later eliminated by [AHH24]. Moreover, [BGK+24] presented a publicly verifiable PKFE-SKL scheme, where the verification key can be made public, based on indistinguishability obfuscation (iO).

[1] For instance, NTCFs have a candidate construction from group-actions ([AMR22]); the dual-mode SFE could potentially be constructed from code-based assumptions [DVDGMQN08] and LPN variants [DDN14].

[2] [CGJL25, PWYZ24] also have non-interactive key generation algorithms for their PKE-SKL with classical lessor, but their non-interaction relies on the LWE-structure of the scheme. Our approach of removing interaction is more black-box. See the technical overview for more discussions.

Table 1: Comparison of PKE-SKL

	Classical-Lessor	VRA security	Assumption
[AKN+23]			PKE
[APV23, AHH24]			LWE
[CGJL25, PWYZ24]	✓		LWE (poly-modulus LWE in [PWYZ24])
[KMY25]		✓	PKE
Ours	✓	✓	LWE

Table 2: Comparison of PRF-SKL

	Classical-Lessor	VRA security	Assumption
[APV23, AHH24]			LWE
[KMY25]		✓	OWF
Ours	✓	✓	LWE

[CGJL25] constructed PKE-SKL and FHE-SKL with a classical-lessor under the LWE assumption. Their schemes assumed sub-exponential hardness of the LWE problem with sub-exponential modulus, which was subsequently improved to polynomial hardness of the LWE problem with polynomial modulus by [PWYZ24].

[MPY24] developed two DS-SKL schemes: one based on the LWE assumption and another on one-way group actions (OWGA) [JQSY19]. A limitation of their constructions is that the signing key evolves by each signature generation, leading to signing keys and signatures whose sizes grow linearly in the total number of signatures.

Finally, [KNP25] proposed collusion-resistant PKE-SKL and ABE-SKL schemes that support leasing an a-priori unbounded number of quantum secret keys, under the LWE assumption.

Secure software leasing. Secure software leasing (SSL) was introduced by [AL21] as a method to encode classical programs into quantum states that can be leased in a secure manner. The idea of SKL is directly inspired by SSL and can be regarded as its specialization for cryptographic functionalities. Much of the existing research on SSL [AL21, KNY21, BJL+21] has focused on a weaker security model, where pirated programs are restricted to employ the *honest* evaluation procedure. In contrast, [BGK+24] proposed an SSL construction for all differing-input circuit pairs[3], under a stronger model that allows pirated programs to use arbitrary evaluation strategies similarly to SKL. Their scheme, however, relies on iO.

Encryption with certified deletion An encryption scheme with certified deletion [BI20] allows a quantum ciphertext to be deleted in a verifiable way. This notion is closely connected to SKL, as one may view SKL as providing certified deletion of cryptographic keys rather than ciphertexts. In their initial work, [BI20] presented an unconditionally secure one-time secret-key encryption scheme with certified deletion. Subsequent works [HMNY21, Por23, BK23, HKM+24, BGK+24] have explored certified deletion for richer forms of encryption, including public-key encryption, attribute-based encryption, functional encryption, fully homomorphic encryption, and witness encryption. Also, [KNY23, BKM+23] extended these results and obtained publicly verifiable certified deletion schemes under minimal assumptions.

1.3 Concurrent Work

Concurrent work by Takeuchi and Xu [TX25] obtains a result similar to ours. In particular, they also construct PKE-SKL, PRF-SKL, and DS-SKL with classical lessors from the LWE assumption. However, their security proof relies on an

[3]Informally, a pair of circuits (C_0, C_1) is said to be differing input if it is computationally difficult to find an input y such that $C_0(y) \neq C_1(y)$.

Table 3: Comparison of DS-SKL

	Classical-Lessor	VRA security	Signing key size	Signature size	Assumption
[MPY24]			$n \cdot \mathrm{poly}(\lambda)$	$n \cdot \mathrm{poly}(\lambda)$	LWE
[MPY24]			$n \cdot \mathrm{poly}(\lambda)$	$n \cdot \mathrm{poly}(\lambda)$	OWGA
[KMY25]		✓	$\mathrm{poly}(\lambda)$	$\mathrm{poly}(\lambda)$	SIS
Ours	✓	✓	$\mathrm{poly}(\lambda)$	$\mathrm{poly}(\lambda)$	LWE

n: the number of issued signatures, λ: the security parameter.

additional conjecture concerning parallel repetition of a certified-deletion property of the magic square game. Moreover, their key generation procedure requires polynomially many rounds, whereas ours is non-interactive for PKE-SKL and requires only a constant number of rounds for PRF-SKL and DS-SKL.

1.4 Technical Overview

Security notion for secure key leasing The security of the secure key leasing scheme roughly guarantees that if the adversary who has performed the key generation algorithm with the challenger to obtain a quantum secret key sk, later outputs a deletion certificate that is accepted by the deletion verification algorithm $\mathsf{DelVrfy}$, the adversary can no longer perform the task that requires sk.

Like [KMY25], our schemes remain secure even if the secret deletion verification key is given to the adversary after the adversary outputs a deletion certificate. Such security definition is called security against verification key revealing attacks (VRA security), defined in [KMY25]. We refer the readers to [KMY25] for more detailed discussions on the definitions.

We can define both an unpredictability style security definition and an indistinguishability style one for PKE-SKL and PRF-SKL, though it is inherently unpredictability style for DS-SKL. The former can be upgraded into the latter by using the standard (quantum) Goldreich-Levin technique as shown in previous works [AKN+23, KMY25]. Thus, in this work, we primarily focus on constructing secure key leasing primitives satisfying unpredictability style security notions.

The unpredictability style security experiment for X-SKL is abstracted by using a predicate P that determines the adversary's goal. It is described as follows.

1. The challenger and the adversary $\mathcal{A}$ generate sk by executing the (possibly interactive) key generation algorithm $\mathit{IntKeyGen}$. If $\mathcal{A}$ follows the protocol, it gets sk, but it may deviate from the honest protocol. (If X is PKE or DS, $\mathcal{A}$ is also given the corresponding public key output by $\mathit{IntKeyGen}$.)

2. $\mathcal{A}$ outputs a classical string cert. If $\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \bot$, where τ is the transcript of the interaction and dvk is the deletion verification key, the experiment ends. Otherwise, the challenger generates a challenge chall and corresponding auxiliary information aux, and sends dvk and chall to $\mathcal{A}$. Note that aux is used by the predicate P to determine whether $\mathcal{A}$ wins the experiment or not.

3. $\mathcal{A}$ outputs ans.

$\mathcal{A}$ wins the experiment if both $\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \top$ and $P(\mathsf{aux}, \mathsf{ans}) = 1$ holds simultaneously. For example, if X is PKE, chall is a ciphertext of a uniformly random message m, aux is m, and ans is the guess m' for m by $\mathcal{A}$. Also, $P(\mathsf{aux} = m, \mathsf{ans} = m')$ outputs 1 if and only if $m' = m$. It guarantees that the winning probability of any QPT $\mathcal{A}$ is negligible in the security parameter.

For the next few paragraphs in the technical overview, we first focus on secure key leasing for decryption keys for a PKE scheme and discuss how we generalize to other cryptographic functionalities later.

BB84 states based quantum key leasing We start by basing our initial ideas on the secure key leasing framework in [KMY25], which is built on the properties of BB84 states [BB20]. BB84 states are states with the following format: for

strings $x \in \{0,1\}^n$ and $\theta \in \{0,1\}^n$, we let a BB84 state $\left|x^\theta\right\rangle := H^{\theta[1]}\left|x[1]\right\rangle \otimes \cdots \otimes H^{\theta[n]}\left|x[n]\right\rangle$, where H is the Hadamard operator, and $x[i]$ and $\theta[i]$ are the i-th bits of x and θ, respectively. To put in a very informal manner, BB84 states are unclonable and have been used in many quantum cryptographic protocols. For example, [KMY25] deployed the following uncertainty-principle-like property of BB84 states (called certified deletion property in [BSS21]): if an adversary $\mathcal{A}$ without the knowledge of x, θ can correctly output all the value $x[i]$ encoded in Hadamard basis (i.e. $x[i]$ for all i where $\theta[i] = 1$), it is impossible for $\mathcal{A}$ to obtain the value of $x[i]$ for the positions i encoded in the computational basis (i.e. $x[i]$ for all i where $\theta[i] = 0$). [4]

However, as we can see, BB84 states alone are simply random strings encoded in random bases. To formulate a secure key leasing scheme for decryption keys of a PKE, one needs to carefully embed a decryption key into the structure of BB84 states so that we can obtain an arbitrary-polynomially many-time reusable quantum decryption key and meanwhile maintain the unclonable property. To achieve this, [KMY25] generates $2n$ i.i.d classical keys for a classical PKE and entangles them respectively with each qubit in the BB84 states $\left|x^\theta\right\rangle$ in the following manner: Generates $2n$ classical secret keys $(\mathsf{sk}_{i,b})_{i\in[n],b\in\{0,1\}}$ and $(x,\theta) \leftarrow \{0,1\}^n \times \{0,1\}^n$. It then applies the map M_i that acts as $|b\rangle\,|c\rangle \to |b\rangle\,|c \oplus \mathsf{sk}_{i,c}\rangle$ to the i-th qubit of the BB84 state $\left|x^\theta\right\rangle$ and $|0\cdots0\rangle$, and obtains a quantum state $s\mathcal{K}_i$ for every $i \in [n]$. Namely, we have

$$s\mathcal{K}_i := \begin{cases} \left|x[i]\right\rangle \left|\mathsf{sk}_{i,x[i]}\right\rangle & (\text{if } \theta[i] = 0) \\ \frac{1}{\sqrt{2}}\left(|0\rangle\,|\mathsf{sk}_{i,0}\rangle + (-1)^{x[i]}\,|1\rangle\,|\mathsf{sk}_{i,1}\rangle\right) & (\text{if } \theta[i] = 1). \end{cases} \tag{1}$$

The resulting secret key is set to $s\mathcal{K} := (s\mathcal{K}_1, \ldots, s\mathcal{K}_n)$. The corresponding deletion verification key is the basis information, and the value of x and the classical secret keys at the indices encoded in Hadamard basis: $\mathsf{dvk} := (\theta, (x[i])_{i\in[n]:\theta[i]=1}, (\mathsf{sk}_{i,0}, \mathsf{sk}_{i,1})_{i\in[n]:\theta[i]=1})$.

To encrypt a message, anyone with access to the public key $\mathsf{pk} = (\mathsf{pke.pk}_{i,b})_{i\in[n],b\in\{0,1\}}$ and a message $m = m_1\|m_2\|\ldots\|m_n$ (padding the message w.l.o.g.), where $\mathsf{pke.pk}_{i,b}$ is the public key corresponding to $\mathsf{sk}_{i,b}$, can generate $\mathsf{pke.ct}_{i,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,b}, m_i)$ for $i \in [n]$ and $b \in \{0,1\}$ and output $\mathsf{ct} = (\mathsf{pke.ct}_{i,b})_{i\in[n],b\in\{0,1\}}$.

To decrypt with the quantum key $s\mathcal{K} := (s\mathcal{K}_1, \ldots, s\mathcal{K}_n)$ on a ciphertext $\mathsf{ct} = \{\mathsf{pke.ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}$: for each $i \in [n]$, decrypt $(\mathsf{pke.ct}_{i,0}, \mathsf{pke.ct}_{i,1})$ coherently with $s\mathcal{K}_i$[5] to get m_i and output $m = m_1\|m_2\|\ldots\|m_n$.

We will specify more details on deletion, revocation and verification in the following paragraphs after we get a full picture of our construction.

Dequantizing the lessor entirely The next major question is how we can dequantize the preparation of the quantum key as well as the revocation of the key for the lessor, making the lessee do all the quantum work. In particular, the step of preparing the key is challenging to dequantize–the lessee has to prepare a "computationally unclonable" quantum state from only classical information and without knowing its description.

Naturally, we look for inspiration from methods of dequantizing the client in the context of delegating quantum computation and state preparation. One natural approach is to run a remote state preparation (RSP) protocol that allows a classical client to instruct a quantum server to prepare a quantum state whose description is known to the client but not to the server.[6] However, existing RSP protocols [GV19, Zha22, GMP23, Zha25] suffer from the drawback that they achieve only inverse-polynomial security. Consequently, directly using these protocols to replace quantum communication in existing SKL schemes yields a scheme whose security bound is fixed in advance to be inverse-polynomial. It remains unclear how to amplify this to the standard notion of negligible security. In addition, existing RSP protocols are highly costly in both communication and rounds, typically requiring polynomially many rounds. This contradicts the goal of designing a lightweight protocol. The cost of generic-purpose RSP protocols stems from their very strong security requirement: more concretely, a rigidity statement asserting that the server must exactly perform certain quantum operations up to isometry. However, such strong guarantees are not necessary for our purposes. We simply need the server (lessee) to prepare a classically functional quantum state and later delete the function.

[4] As we will discuss later, we will not directly use the property of BB84 states as [KMY25] does, so we will not go into more details here.

[5] In more detail, we execute the circuit that has $(\mathsf{pke.ct}_{i,0}, \mathsf{pke.ct}_{i,1})$ hardwired, takes (b, sk) as input, and outputs $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{pke.ct}_{i,b})$ in superposition and measures the outcome written onto an additional register.

[6] More precisely, what we need is an RSP protocol with verification, in which the quantum client can verify that the quantum server has indeed prepared the desired state up to isometry.

Another possible approach is delegating the computation via a quantum fully homomorphic encryption (QFHE) scheme. By sending QFHE's ciphertext of $x, \theta, \{\mathsf{sk}_{i,0}, \mathsf{sk}_{i,1}\}_{i\in[n]}$, the lessor can let the lessee prepare the above quantum key by homomorphically evaluating on the ciphertext. However, in our case, the obtained key on the server (lessee)'s end will be encrypted and unusable for the decryption functionality. On the other hand, revealing the QFHE decryption information will result in insecurity.

The seminal works [BCM+18, Mah18] introduced a protocol where a classical client can verify the quantumness and even any BQP computation of a quantum server. In particular, during the protocol of [BCM+18, Mah18], after receiving only classical information of a function f from the client, the server has to create a (computationally) unclonable state on its own end, with the following structure:

$$\frac{1}{\sqrt{2}}(|0, a_0\rangle + |1, a_1\rangle), f(a_0) = f(a_1) = y$$

Such a function family f is called a *(noisy) trapdoor claw-free family* [7]. After preparing the above state (which we will refer to from now on as the *claw state*), the server has to "commit" to this state by sending back the classical image y.

In the next step of the protocol, the client will request the server to measure this state randomly in either the computational or Hadamard basis and send back the result. The client, using a trapdoor for the corresponding f (which it generates together with f) can verify if the server has done the requested measurement.

Informally, the trapdoor claw-free function has the following properties:

1. It is computationally hard to find two pre-images a_0, a_1 such that $f(a_0) = f(a_1) = y$;
2. Once the server has measured the state in the Hadamard basis, and obtains a string $(e, d) \in \{0,1\}^{n+1}$ such that $e = d \cdot (a_0 \oplus a_1) \bmod 2$ that it sends back to the client for verification, it will (computationally) be deprived of the preimage information a_0 and a_1 encoded in the computational basis.
3. It can also be extended to have a dual-mode property: in the two-to-one mode, there are two pre-images a_0, a_1 for each image y (as shown above); in the injective mode, there is only one preimage a for each image y. In other words, the claw state prepared by the server in this mode will be $|b, a_b\rangle$, where $f(a_b) = y$ for some random $b \in \{0,1\}$. The two modes are computationally indistinguishable.

The second property is formalized as *adaptive hardcore bit property* in [BCM+18, Mah18]. We can observe that this adaptive hardcore bit property aligns with the property of the BB84 states described above: if one measures the keys in the Hadamard basis, one loses all the information, i.e. the classical description for the secret keys, in the computational basis.

To make this intuition work, we will look for an approach where we can let the server prepare the secret key sk in the above BB84 format (1). The lessor will pick uniformly random $\theta \leftarrow \{0,1\}^n$; it then generates n NTCF functions (with corresponding trapdoors) $(f_1, \cdots, f_n)$ such that f_i is in injective mode for positions $\theta[i] = 0$ and f_i is in two-to-one mode for positions $\theta[i] = 1$; then it sends $(f_1, \cdots, f_n)$ to the lessee. After some key generation procedure that helps the lessee go from preparing claw states to preparing quantum keys (which we will go into later), the lessee obtains the sk where: for indices i with $\theta[i] = 0$, the quantum key $sk_i := |b_i\rangle |\mathsf{sk}_{i,b_i}\rangle$ is prepared by the injective mode f_i; for indices i, $\theta[i] = 1$, the quantum key $sk_i := \frac{1}{\sqrt{2}}(|0\rangle |\mathsf{sk}_{i,0}\rangle + |1\rangle |\mathsf{sk}_{i,1}\rangle)$ is prepared by the two-to-one mode f_i.

To delete the key, the lessor asks the lessee to measure all the keys $sk := (sk_1, \cdots, sk_n)$ in the Hadamard basis and send back the measurement outcomes. Using the verification key which contains the NTCF trapdoors for the indices encoded in Hadamard basis, the lessor can verify the deletion (measurement) outcomes sent by the lessee[8].

Preparing the quantum key with secure function evaluation The next step is to allow the lessee to go from preparing the claw state to preparing the actual quantum key sk. That is, we start from generating a state of the form

[7] The pre-images a_0, a_1 are conventionally denoted as x_0, x_1. We do the same in the actual construction and proof. Here, we use a to distinguish from the string x in the previously mentioned BB84 states.

[8] As an informed reader may notice, the classical-lessor PKE-SKL in [CGJL25] is also based on the NTCF approach in [BCM+21, Mah18]. In fact, they directly use the claw state as the quantum secret key. But we will soon discuss in the upcoming technical overview that using exactly the claw state may not be enough for a modular SKL scheme and may be difficult to apply on more generic cryptographic primitives.

$|0\rangle |a_0\rangle + |1\rangle |a_1\rangle$ by using an NTCF, and convert it into a state of the form $|0\rangle |\mathsf{sk}_0\rangle + |1\rangle |\mathsf{sk}_1\rangle$ for secret keys sk_0 and sk_1 [9]. Our idea is to achieve this by secure function evaluation (SFE). Roughly, SFE is a two-round protocol between a sender and receiver that works as follows:

- The receiver takes a message m as input and sends a ciphertext ct to the sender while keeping a state information st.
- The sender takes a circuit C as input, receives ct, computes ct' and sends ct' to the receiver.
- The receiver recovers $C(m)$ from st and ct'.

In particular, we consider SFE in the CRS model. The security roughly requires that the sender learns nothing about m, and the receiver learns nothing about C beyond $C(m)$. We will specify here some of the concrete security properties we require from the SFE. Notably, we will show that we can obtain SFE with all the properties we need from LWE. In particular, we require the following *dual-mode* property. [10]

- **Mode indistinguishability.** There are two modes for generation of CRS, *hiding mode* and *extraction mode*. In both modes, an associated trapdoor is generated along with CRS. CRSs generated in different modes are computationally indistinguishable.
- **Statistical security against malicious senders in the hiding mode**: In the hiding mode, ct statistically hides m.
- **State recoverability in the hiding mode:** In the hiding mode, ct and m uniquely determine the corresponding st. Moreover, the trapdoor enables us to efficiently recover st from ct and m.
- **Extractability in the extractable mode**: In the extractable mode, we can extract m from (possibly maliciously generated) ct using the trapdoor, so that ct' can be simulated only using $C(m)$ (and not using C). Note that this automatically implies (computational) security against malicious receivers in the extractable mode.

In addition to the above, we need two additional properties: one is the "efficient superposition generation property" and the other is the "decomposability of the state". We explain those additional properties when we need them.[11]

Using SFE, consider the following sub-protocol to generate the key. For simplicity, we consider the "stand-alone" generation of a single key sk_i as part of the final sk, where $\theta[i] = 1$ and $x[i] = 0$.

1. The lessor generates CRS of SFE in the hiding mode and a two-to-one function f from NTCF and sends (CRS, f) to the lessee.
2. The lessee prepares the claw state $|0\rangle |a_0\rangle + |1\rangle |a_1\rangle$ along with $y = f(a_0) = f(a_1)$ by the standard usage of NTCF and sends y to the lessor.
3. The lessee runs the receiver's algorithm of SFE in superposition, where the second register of $|0\rangle |a_0\rangle + |1\rangle |a_1\rangle$ is treated as the message, and measures the resulting ct. Since ct statistically hides the message (which is either a_0 or a_1), this measurement almost never collapses the two branches. Moreover, since ct and the message uniquely determines the corresponding state, the resulting state (un-normalized for convenience) looks like:

   $$|0\rangle |a_0\rangle |\mathsf{st}_0\rangle + |1\rangle |a_1\rangle |\mathsf{st}_1\rangle$$

   where st_0 and st_1 are state information corresponding to (a_0, ct) and (a_1, ct), respectively. [12] Here, we are relying on an additional property of our SFE "efficient superposition generation property", which makes sure that we can generate the above state without remaining entanglement with the randomness used by the receiver.

   The lessee sends the measured ciphertext ct to the lessor.

[9] In the technical overview, for the sake of presentation simplicity, we will sometimes only focus on the $|+\rangle$-format of the claw state and secret key (i.e. $|0\rangle |a_0\rangle + |1\rangle |a_1\rangle$ and correspondingly $|0\rangle |\mathsf{sk}_0\rangle + |1\rangle |\mathsf{sk}_1\rangle$) . But it is easy to see that the following discussions apply to the $|0\rangle, |1\rangle$-versions of the state.

[10] State recoverability in the hiding mode is not a standard requirement for dual-mode SFE, but the LWE-based construction satisfies it.

[11] The former appears in the description of the sub-protocol below; the latter appears in the idea for the security proof.

[12] While this might be unclear from the notation, st_b depends on x_b (and not only on b.) We write st_b instead of st_{x_b} for simplicity.

4. In the next step, the lessor can define a circuit C that maps a_b to some secret key sk_b for $b \in \{0,1\}$ respectively. Then the lessor runs C on ct and sends the result ct' to the lessee and the lessee runs the output derivation algorithm coherently, writing the outcome on an additional register. By the correctness of the SFE, the lessee will finally obtain $|0\rangle |a_0\rangle |\mathsf{st}_0\rangle |\mathsf{sk}_0\rangle + |1\rangle |a_1\rangle |\mathsf{st}_1\rangle |\mathsf{sk}_1\rangle$.

   However, in the actual construction, the circuit C will be slightly more sophisticated than described here. We will elaborate in the next subsection when we discuss the security proof.

While this is not exactly the state of the form $|0\rangle |\mathsf{sk}_0\rangle + |1\rangle |\mathsf{sk}_1\rangle$ as we originally discussed, it is already good enough for decryption since both branches contain the information of a secret key, so that during decryption, the lessee can use the secret key in the last register to decrypt coherently. Moreover, if we define the deletion algorithm as just a Hadamard basis measurement of the above state, then we can still define a deletion verification algorithm, where we verify that the certificate (e,d,c) satisfies: $e = (d\|c) \cdot (a_0\|\mathsf{st}_0\|\mathsf{sk}_0 \oplus a_1\|\mathsf{st}_1\|\mathsf{sk}_1)$. This can also be verified with the trapdoor of the NTCF, the trapdoor of the SFE and the circuit C. We will make some further minor changes to the deletion and verification algorithms, but for the purpose of intuition, we leave them as they are now.

Watermarking: extraction of the pre-images For the security proof, we would like to make the following argument. Recall the security states that the lessee, after successfully passing the revocation verification, will not be able to distinguish the encryptions of two messages (an IND-CPA style security game). Suppose there exists a quantum adversary that passes both the revocation verification and the IND-CPA game afterwards, then we should be able to break the security of the NTCF used, namely, the adaptive hardcore bit property. We can argue in a relatively straightforward manner that the adversary provides the reduction with the measurement outcomes in the Hadamard basis, if it passes the revocation verification. The trickier part is to argue that by passing the IND-CPA game, we can extract the preimage a_0 or a_1 from this quantum decryptor. Finally, by gathering both the measurements in the Hadamard basis and the preimages a_0 or a_1, the reduction can break the adaptive hardcore bit property of the NTCF.

The previous work on fully classical lessor SKL [CGJL25] also uses an NTCF based construction and need to extract the preimages in order to break the adaptive hardcore bit security. They design an algorithm that carefully extracts the preimage information from a possibly destructive quantum decryptor, by using quantum extraction techniques as well as LWE properties. However, this approach is likely limited to analyzing the LWE based PKE scheme in [CGJL25], and does not generalize to any PKE and other cryptographic functionalities.

In this work we provide a simple approach through *watermarking*. This approach is not only is cleaner, but also works for any PKE scheme and is generalizable to other cryptographic functionalities such as PRF and signatures presented in this work. A PKE with watermarking (often called watermarkable PKE) has syntax similar to standard PKE except that the KeyGen algorithm generates a marking key along with the encryption/decryption keys and we have an additional "marking" algorithm Mark. Mark will embed a watermark into an input cryptographic function; the watermarking security roughly states that trying to remove the watermark will destroy the functionality and thereby any functional program output by an adversary who receives a marked program will contain the original mark with high probability. We don't need additional assumptions for a watermarkable PKE scheme: any PKE scheme can be upgraded to have the watermarking security requirements we need.

Next, we go back to modify step 4 of the construction when the lessor receives the ciphertext ct of SFE in step 3 from the lessee. It does the following instead:

- The lessor defines a circuit C that takes a as input and outputs a "watermarked" secret key $\mathsf{sk}(a)$ in which a "mark" a is embedded. The precise requirement for the watermarking will become clear later. Then runs the sender's algorithm of SFE on C and ct to generate ct' and sends it to the lessee.

- The lessee runs the output derivation algorithm of SFE in superposition to get

$$\begin{aligned} &|0\rangle |a_0\rangle |\mathsf{st}_0\rangle |C(a_0)\rangle + |1\rangle |a_1\rangle |\mathsf{st}_1\rangle |C(a_1)\rangle \\ =\ &|0\rangle |a_0\rangle |\mathsf{st}_0\rangle |\mathsf{sk}(a_0)\rangle + |1\rangle |a_1\rangle |\mathsf{st}_1\rangle |\mathsf{sk}(a_1)\rangle . \end{aligned} \tag{2}$$

Now going back to the security proof: the watermarking security requires that if an adversary given $\mathsf{sk}(a)$ (where the mark a is chosen by the adversary) generates a (possibly quantum) "decryptor" that can correctly decrypt a ciphertext of a

uniformly random message with probability ϵ, then we can extract a from the decryptor with probability $\epsilon - \mathsf{negl}(\lambda)$.[13] This even holds if the decryptor is a QPT algorithm that has an auxiliary quantum state, as in our scenario. Therefore, we can use a successful quantum decryptor to extract the marked pre-image a_0 or a_1.

Moreover, we show that a similar extractability holds even in the parallel setting where many instances of the scheme under independently generated keys. That is, if the decryptor succeeds in decryption simultaneously for all instances with probability ϵ, then we should succeed in simultaneously extracting the marks with probability $\epsilon - \mathsf{negl}(\lambda)$. Recall that in our final SKL protocol, the secret key sk contains n independently generated keys sk_i. We will extract the watermarked preimages a_{i,b_i} from all of them using the parallel extraction property.

Cut-and-choose adaptive hardcore bit To make the final security reduction go through, we need to perform a reduction where the reduction plays as the adversary in a security game regarding the NTCF and hopes to leverage the PKE-SKL adversary to break the security of NTCF, namely the adaptive hardcore bit property previously mentioned.

One subtlety is that not all our quantum keys $sk_i, i \in [n]$ are "claw states" that contain a superposition of two claws, but some are classical states that only contain one pre-image, where the adaptive hardcore bit security is not well defined. On the other hand, let us recall that the BB84 states have a certified deletion property shown in [BSS21]): if an adversary $\mathcal{A}$ without the knowledge of x, θ can correctly output all the value $x[i]$ encoded in Hadamard basis (i.e. $x[i]$ for all i where $\theta[i] = 1$), it is impossible for $\mathcal{A}$ to obtain the value of $x[i]$ for the positions i encoded in the computational basis. We therefore would like to find a property of the NTCF/claw states similar to the certified deletion property of BB84 states used in [KMY25], and thus we may be able to replace the latter with the former in the security proof.

Here, we use a property called *cut-and-choose adaptive hardcore bit* property for dual-mode NTCF introduced by Hiroka et al. [HMNY21]. They prove that *any* dual-mode NTCF that satisfies the adaptive hardcore bit property also satisfies the cut-and-choose adaptive hardcore bit property. Our final construction is designed in such a way that its security can be reduced to the cut-and-choose adaptive hardcore bit property.

We consider the following experiment between an adversary $\mathcal{A}$ and challenger:

1. The challenger chooses a uniform subset $S \subseteq [2n]$ such that $|S| = n$.
2. For each $i \in [2n]$, the challenger generates a function f_i along with its trapdoor, where:
   - If $i \in S$, f_i is in the injective mode,
   - If $i \notin S$, f_i is in the two-to-one mode.

   The challenger sends $\{f_i\}_{i\in[2n]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends $\{y_i, d_i, e_i\}_{i\in[2n]}$ to the challenger.
4. For each $i \notin S$, the challenger recovers the preimages $(a_{i,0}, a_{i,1})$, and checks if $e_i = d_i \cdot (a_{i,0} \oplus a_{i,1})$ and $d_i \neq \mathbf{0}$ hold. If they do not hold for some $i \notin S$, the challenger immediately aborts and the experiment returns 0.
5. The challenger sends S to $\mathcal{A}$.
6. $\mathcal{A}$ sends $\{b_i, a_i\}_{i\in S}$ to the challenger.
7. The challenger checks if $f_i(b_i, a_i) = y_i$ holds for all $i \in S$. If this holds for all $i \in S$, the experiment returns 1. Otherwise, it returns 0.

The cut-and-choose adaptive hardcore bit property claims that for any QPT adversary $\mathcal{A}$, the probability that the above experiment returns 1 is negligible (for sufficiently large n).

[13] This security guarantee is weaker than standard watermarking security, but is sufficient for our purpose.

Putting things together for the security analysis Now our goal is to reduce the PKE-SKL security to the cut-and-choose adaptive hardcore bit property, in a manner similar to that of [KMY25] with the certified deletion property of BB84 states. To do so, we mainly have to prove the following two steps:

1. If one knows the trapdoors of the SFE (but not that of NTCF), any valid deletion certificate $\mathsf{cert} = (e_i, d_i, c_i)_{i\in[2n]}$ can be used to compute $(\hat{e}_i, \hat{d}_i)_{i\notin S}$ such that $\hat{e}_i = \hat{d}_i \cdot (a_{i,0} \oplus a_{i,1})$ and $\hat{d}_i \neq \mathbf{0}$.
2. If a "decryptor" (the state after passing the deletion verification) correctly decrypts the special ciphertext with probability ϵ, we can extract $\{a_i\}_{i\in S}$ with probability at least $\epsilon - \mathsf{negl}(\lambda)$.

Recall that in the PKE-SKL security game, the adversary has to hand in a valid deletion certificate and then succeeds in decrypting. Therefore, the goal is to use such an adversary to obtain both of the above items; then we can use them to break the cut-and-choose adaptive hardcore bit property of the NTCF.

Now we argue how to show each item above.

As we have discussed in the watermarking paragraph, the second item straightforwardly follows from the security of watermarkable PKE.

However, the first item is less obvious. First, recall that for all $i \notin S$ (indices for 2-to-1 f_i), the (unnormalized) quantum keys will look like

$$\mathsf{sk}_i = |0\rangle |a_{i,0}\rangle |\mathsf{st}_{i,0}\rangle |\mathsf{sk}(a_{i,0})\rangle + |1\rangle |a_{i,1}\rangle |\mathsf{st}_{i,1}\rangle |\mathsf{sk}(a_{i,1})\rangle$$

We make some minor modifications on how we delete a key to make the analysis simpler: in the deletion algorithm, the honest lessee first uncomputes the register containing the secret keys $|\mathsf{sk}(a_{i,b})\rangle, b \in \{0,1\}$ using the circuit C from SFE which generates the key. Afterwards, it measures all registers in the Hadamard basis. Now we know that when a deletion certificate $\mathsf{cert} = (e_i, d_i, c_i)_{i\in[2n]}$ received from the adversary is valid, we have the following for all $i \notin S$:

$$e_i = d_i \cdot (a_{i,0} \oplus a_{i,1}) \oplus c_i \cdot (\mathsf{st}_{i,0} \oplus \mathsf{st}_{i,1}).$$

Here, we need to use another property of the SFE, named "decomposability of the state", which we have mentioned but did not go into details. Elaborated below, this property allows us to obtain item 1 in a relatively clean way.

Recall that the receiver in the SFE protocol takes a message a and a common reference string CRS, then outputs a ciphertext ct to the sender while also keeping a (classical) secret state information st. This "decomposability of the state" property is with respect to the state st of the SFE: intuitively, it says that st corresponding to input message a and output ct (sometimes we say the state st corresponds to (a, ct) for simplicity) is of the following form

$$\mathsf{st} = \mathsf{st}_{1,a[1]} \| \mathsf{st}_{2,a[2]} \| \dots \| \mathsf{st}_{m,a[m]}$$

where m is the bit-length of a, $a[j]$ is the j-th bit of a, and $(\mathsf{st}_{j,b})_{j\in[m],b\in\{0,1\}}$ is a family of strings that only depend on ct and CRS (and not on a). Using this, for each $b \in \{0,1\}$ we can write $\mathsf{st}_{i,b}$ as

$$\mathsf{st}_{i,b} = \mathsf{st}_{i,1,a_{i,b}[1]} \| \mathsf{st}_{i,2,a_{i,b}[2]} \| \dots \| \mathsf{st}_{i,m,a_{i,b}[m]}.$$

For each $i \in [2n]$, $b \in \{0,1\}$, and $j \in [m]$, let $\mathsf{st}_{i,b}[j]$ be the j-th block of $\mathsf{st}_{i,b}$, i.e.,

$$\mathsf{st}_{i,b}[j] = \mathsf{st}_{i,j,a_{i,b}[j]}.$$

We divide the string c_i in the deletion certificate $\mathsf{cert} = \{(e_i, c_i, d_i)\}_{i\in[2n]}$ into m blocks:

$$c_i = c_{i,1} \| c_{i,2} \| \dots \| c_{i,m}.$$

For $i \in [2n]$ and $j \in [m]$, we have

$$\begin{aligned}
& c_{i,j} \cdot (\mathsf{st}_{i,0}[j] \oplus \mathsf{st}_{i,1}[j]) \\
&= c_{i,j} \cdot (\mathsf{st}_{i,j,a_{i,0}[j]} \oplus \mathsf{st}_{i,j,a_{i,1}[j]}) \\
&= \begin{cases} 0 & \text{if } a_{i,0}[j] = a_{i,1}[j] \\ c_{i,j} \cdot (\mathsf{st}_{i,j,0} \oplus \mathsf{st}_{i,j,1}) & \text{if } a_{i,0}[j] \neq a_{i,1}[j] \end{cases} \\
&= (c_{i,j} \cdot (\mathsf{st}_{i,j,0} \oplus \mathsf{st}_{i,j,1})) \cdot (a_{i,0}[j] \oplus a_{i,1}[j]).
\end{aligned}$$

By observing the above calculation, we define a string d_i^* as

$$d_i^*[j] = c_{i,j} \cdot (\mathsf{st}_{i,j,0} \oplus \mathsf{st}_{i,j,1})$$

for $j \in [m]$, then we have

$$c_{i,j} \cdot (\mathsf{st}_{i,0}[j] \oplus \mathsf{st}_{i,1}[j]) = d_i^*[j] \cdot (a_{i,0}[j] \oplus a_{i,1}[j]).$$

This implies

$$c_i \cdot (\mathsf{st}_{i,0} \oplus \mathsf{st}_{i,1}) = d_i^* \cdot (a_{i,0} \oplus a_{i,1}).$$

Substituting this relation into $e_i = d_i \cdot (a_{i,0} \oplus a_{i,1}) \oplus c_i \cdot (\mathsf{st}_{i,0} \oplus \mathsf{st}_{i,1})$, we have

$$e_i = (d_i \oplus d_i^*) \cdot (a_{i,0} \oplus a_{i,1}).$$

We can then observe that the reduction works as long as $d_i \neq d_i^*$, which happens with overwhelming probability. We therefore obtain a string $(e_i, d_i \oplus d_i^*)$ to help us break the cut-and-choose adaptive hardcore bit property, along with the computational basis information $\{a_{i,b}\}_{i \in S}$ we extract using the watermarking security from the successful decryptor.

Summary so far: classical-lessor SKL from "parallel extractable" watermarking To summarize, we obtain a compiler that transforms a parallel-extractable watermarkable PKE scheme into a classical-lessor PKE-SKL scheme using NTCF and our special SFE, both of which can be based on the LWE assumption. In this compiler, we rely on the implicit fact that the functionality of PKE is preserved under parallel composition (i.e., the encryption and decryption of the resulting classical-lessor PKE-SKL scheme respectively correspond to $2n$ parallel executions of the encryption and decryption of the underlying watermarkable PKE). More generally, this compiler can be applied not only to watermarkable PKE but also to any cryptographic primitive equipped with parallel-extractable watermarking, provided that the functionality of the primitive is preserved under parallel composition.

Classical-lessor SKL for PRF and digital signature Using the above compiler, we obtain classical-lessor SKL for unpredictable functions (UPFs) and digital signature, where the former can then be transformed into a classical-lessor PRF-SKL via the quantum Goldreich-Levin technique, as discussed earlier. Following the previous works [APV23, AHH24, MPY24, KMY25], we consider the security notion for UPF-SKL and DS-SKL where the challenge input/message is chosen uniformly at random and after the adversary outputs the deletion certificate and is given the challenge, it is not given oracle access to the leased functionality (e.g., signing oracle in the DS-SKL case). Under this "non-adaptive" setting, the functionality of UPFs and digital signature is preserved under parallel composition. Thus, relying on our compiler, to achieve classical-lessor UPF-SKL and DS-SKL, the main task is to construct a parallel-extractable watermarkable UPF and digital signature.

UPFs: Building on the technique of [KMY25], we achieve parallel extractable watermarkable UPFs using two-key equivocal PRFs (TEPRFs) [HJO+16], which can be based on OWFs. By combining it with our compiler, we obtain the first classical-lessor PRF-SKL based on the LWE assumption.

Digital signature: To apply our compiler, we require a parallel extractable watermarkable signature scheme with an additional property known as coherent signability, introduced by [KMY25] in the context of constrained signatures. Coherent signability is defined for digital signatures in which a signing key (e.g., a constrained or watermarked key) can be generated from a master secret key. Intuitively, it ensures that for any message m and two signing keys sk_0 and sk_1, one can generate a valid classical signature for m using the superposed signing key $\alpha |0\rangle |\mathsf{sk}_0\rangle |\psi_0\rangle + \beta |1\rangle |\mathsf{sk}_1\rangle |\psi_1\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$, almost without disturbing the superposed state, provided that both sk_0 and sk_1 can individually generate valid signatures for m using the standard (classical) signing algorithm. This property is necessary when our compiler is applied because, in a 2-to-1 mode position, a signature must be generated using such a superposed signing key (see Equation (2)).

[KMY25] showed how to construct a constrained signature scheme with coherent signability under the SIS assumption. We extend this by showing that a parallel-extractable watermarkable signature scheme with coherent signability can be obtained by combining any constrained signature scheme with coherent signability and TEPRFs. The construction builds on that of parallel-extractable watermarkable UPFs. By further applying our compiler, we obtain the first classical-lessor DS-SKL based on the LWE assumption.

Removing interaction during key generation for PKE-SKL Finally, note that our constructions above require a constant-round interaction between the lessor and the lessee for key generation. This is not ideal for practice. We improve this aspect by making the key generation algorithm non-interactive in the following way: instead of letting the lessee prepare the full key in the key generation procedure, it only prepares the claw state and executes the first SFE receiver operation. We delay the generation for the watermarkable PKEs keys to the *encryption* algorithm Enc. The ciphertext produced by Enc will contain some auxiliary information about the watermarkable PKE so that the lessee, once receiving the ciphertext, can use the auxiliary information to prepare the full key (the same one as in the interactive construction), by running the second SFE receiver operation on its previous half-baked quantum key. The resulting key will be able to decrypt the corresponding ciphertext.

Construction of the special SFE In general, SFE can be constructed from oblivious transfer (OT) via Yao's well-known garbled-circuits protocol [Yao86], which in turn can be based on OWFs. The same applies in our setting: the task of realizing the required SFE in our compiler reduces to constructing an OT scheme that satisfies all the properties introduced above. (Since OT is a special case of SFE, the properties for SFE can be naturally translated into OT setting.) We show that the OT scheme of [PVW08], based on the LWE assumption, can be naturally extended to achieve these required properties. This gives us the special SFE needed for our compiler under the LWE assumption.

1.5 Discussion

One might expect that the constructions of PKE-SKL with a classical lessor in [CGJL25, PWYZ24] can be directly generalized to other primitives, such as PRF-SKL and DS-SKL. Here, we explain why such a generalization may be difficult.

We begin by briefly recalling the basic idea of [CGJL25, PWYZ24]. Using an NTCF f, one generates a quantum state of the form

$$\frac{1}{\sqrt{2}}(|0,a_0\rangle + |1,a_1\rangle), f(a_0) = f(a_1) = y.$$

Then they observe that, for a specific LWE-based NTCF, each of a_0 and a_1 can be viewed as a decryption key of a certain PKE scheme. Thus, to encrypt a message m, one generates ciphertexts ct_0 and ct_1 under the corresponding public keys. Given ct_0 and ct_1, the state $\frac{1}{\sqrt{2}}(|0,a_0\rangle + |1,a_1\rangle)$ can then be used to decrypt to m. The crucial property of PKE here is that decrypting ct_0 with a_0 and decrypting ct_1 with a_1 yield the same message m, by correctness of the PKE scheme. Therefore, decryption can be performed coherently in superposition without collapsing the state $\frac{1}{\sqrt{2}}(|0,a_0\rangle + |1,a_1\rangle)$.

However, this situation does not extend to general primitives such as PRFs or signature schemes. Since the issue is essentially the same for both, let us focus on signatures. Suppose that a_0 and a_1 are interpreted as secret keys of some classical signature scheme. One might then hope to regard $\frac{1}{\sqrt{2}}(|0,a_0\rangle + |1,a_1\rangle)$ as a quantum signing key for a DS-SKL scheme. However, this does not work in general, because signing the same message using a_0 and a_1 may produce different signatures. In that case, the output reveals the branch, and measuring it collapses the state to one side, making the state unusable for generating the deletion certificate later.

To circumvent this issue in the quantum communication setting, [KMY25] relies on a special signature scheme that has special pairs $(\mathsf{sk}_0, \mathsf{sk}_1)$ such that $\frac{1}{\sqrt{2}}(|0,\mathsf{sk}_0\rangle + |1,\mathsf{sk}_1\rangle)$ can be used to generate a signature without collapsing the state, for almost all messages. In addition, their security proof essentially relies on another special property: a signature on a hidden special message reveals which of sk_0 or sk_1 was used. In our work, we abstract a signature scheme with these properties as a watermarkable signature scheme.

This makes the difficulty of directly applying the techniques of [CGJL25, PWYZ24] to signatures clear. The main obstacle is that it is unclear how to construct a watermarkable signature scheme in which the preimages (a_0, a_1) of the LWE-based NTCF themselves serve as the required special signing-key pair. This is why we need an additional mechanism based on SFE: it allows us to further entangle auxiliary strings in each branch and thereby embed the signing keys of the watermarkable signature scheme into the superposition.

On the other hand, it is plausible that the constructions of [CGJL25, PWYZ24] can be directly adapted to achieve VRA security using the idea, developed in this paper, of reducing security to the cut-and-choose adaptive hardcore-bit

property of NTCFs. More precisely, one could modify the constructions of [CGJL25, PWYZ24] so that the NTCF is used in injective mode on half of the coordinates and in two-to-one mode on the remaining coordinates. If this idea works for the construction of [PWYZ24], it would yield a PKE-SKL scheme with a classical lessor from the *polynomial*-modulus LWE assumption, which would improve on our assumptions. In this paper, we do not pursue this direction, since our goal is to provide a general framework applicable to various primitives, rather than to optimize the construction of PKE-SKL.

2 Preliminaries

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

Let $[\ell]$ denote the set of integers $\{1, \cdots, \ell\}$, λ denote a security parameter, and $y := z$ denote that y is set, defined, or substituted by z. For a finite set X and a distribution D, $x \leftarrow X$ denotes selecting an element from X uniformly at random, $x \leftarrow D$ denotes sampling an element x according to D. Let $y \leftarrow \mathsf{A}(x)$ and $y \leftarrow \mathcal{A}(\chi)$ denote assigning to y the output of a probabilistic or deterministic algorithm A and a quantum algorithm $\mathcal{A}$ on an input x and χ, respectively. When we explicitly show that A uses randomness r, we write $y \leftarrow \mathsf{A}(x;r)$. For a classical algorithm A, we write $y \in \mathrm{Supp}(\mathsf{A}(x))$ or $y \in \mathsf{A}(x)$ to mean that y is in the range of $\mathsf{A}(x)$, i.e., $\Pr[\mathsf{A}(x) = y] > 0$. PPT and QPT algorithms stand for probabilistic polynomial-time algorithms and polynomial-time quantum algorithms, respectively. Let negl denote a negligible function and poly denote a positive polynomial. For a string $x \in \{0,1\}^n$, $x[i]$ is its i-th bit. For strings $x, y \in \{0,1\}^n$, $\langle x, y\rangle$ denotes $\bigoplus_{i\in[n]} x[i]\cdot y[i]$. For quantum states ρ and σ, $\|\rho - \sigma\|_{tr}$ means their trace distance. When we consider security, we consider non-uniform QPT adversary by default.

2.1 Cryptographic Primitives

Public-key encryption.

Definition 2.1 (PKE). *A PKE scheme* PKE *is a tuple of three algorithms* $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. *Below, let* $\mathcal{M}$ *be the message space of* PKE.

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

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

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

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

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

Definition 2.2 (IND-CPA Security). *We say that a PKE scheme* PKE *with the message space* $\mathcal{M}$ *is IND-CPA secure if it satisfies the following requirement, formalized from the experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and the challenger:*

1. *The challenger runs* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(1^\lambda)$ *and sends* ek *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends* $(m_0^*, m_1^*) \in \mathcal{M}^2$ *to the challenger.*
3. *The challenger generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m^*_{\mathsf{coin}})$ *and sends* ct^* *to* $\mathcal{A}$.

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

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

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

Two-key equivocal PRF. We rely on a primitive called two-key equivocal PRF (TEPRF) introduced in [HJO+16]. The following definition is taken verbatim from [KMY25].

Definition 2.3 (Two-Key Equivocal PRF). *A two-key equivocal PRF (TEPRF) with input length ℓ (and output length 1)*[14] *is a tuple of two algorithms* $(\mathsf{KG},\mathsf{Eval})$.

$\mathsf{KG}(1^\lambda,s^*)\to(\mathsf{key}_0,\mathsf{key}_1)$**:** *The key generation algorithm is a PPT algorithm that takes as input the security parameter 1^λ and a string $s^*\in\{0,1\}^\ell$, and outputs two keys* key_0 *and* key_1.

$\mathsf{Eval}(\mathsf{key},s)\to b$**:** *The evaluation algorithm is a deterministic classical polynomial-time algorithm that takes as input a key* key *and an input $s\in\{0,1\}^\ell$, and outputs a bit $b\in\{0,1\}$.*

Equality: *For all $\lambda\in\mathbb{N}$, $s^*\in\{0,1\}^\ell$, $(\mathsf{key}_0,\mathsf{key}_1)\leftarrow\mathsf{KG}(1^\lambda,s^*)$, $s\in\{0,1\}^\ell\setminus\{s^*\}$,*

$$\mathsf{Eval}(\mathsf{key}_0,s)=\mathsf{Eval}(\mathsf{key}_1,s).$$

Different values on target: *For all $\lambda\in\mathbb{N}$, $s^*\in\{0,1\}^\ell$, $(\mathsf{key}_0,\mathsf{key}_1)\leftarrow\mathsf{KG}(1^\lambda,s^*)$,*

$$\mathsf{Eval}(\mathsf{key}_0,s^*)\neq\mathsf{Eval}(\mathsf{key}_1,s^*).$$

Differing point hiding. *For any (stateful) QPT adversary $\mathcal{A}$,*

$$\left|\Pr\left[\mathcal{A}(\mathsf{key}_b)=1:\begin{array}{l}(s_0^*,s_1^*,b)\leftarrow\mathcal{A}(1^\lambda)\\(\mathsf{key}_0,\mathsf{key}_1)\leftarrow\mathsf{KG}(1^\lambda,s_0^*)\end{array}\right]-\Pr\left[\mathcal{A}(\mathsf{key}_b)=1:\begin{array}{l}(s_0^*,s_1^*,b)\leftarrow\mathcal{A}(1^\lambda)\\(\mathsf{key}_0,\mathsf{key}_1)\leftarrow\mathsf{KG}(1^\lambda,s_1^*)\end{array}\right]\right|\le\mathsf{negl}(\lambda).$$

Theorem 2.4 ([HJO+16]). *Assuming the existence of OWFs, there exist TEPRFs with input length ℓ for any polynomial $\ell=\ell(\lambda)$.*

We also need the following property, which follows from the differing point hiding property defined above for $\ell=\omega(\log\lambda)$.

Hard-to-find differing-point. For any $b\in\{0,1\}$ and for any QPT adversary $\mathcal{A}$, we have

$$\Pr\left[\mathcal{A}(\mathsf{key}_b)=s^*:\begin{array}{l}s^*\leftarrow\{0,1\}^\ell\\(\mathsf{key}_0,\mathsf{key}_1)\leftarrow\mathsf{KG}(1^\lambda,s^*)\end{array}\right]\le\mathsf{negl}(\lambda).$$

Coherently-signable constrained signatures. The following definition is taken from [KMY25].

Definition 2.5 (Constrained signatures). *A constrained signature (CS) scheme with the message space $\mathcal{M}$ and constraint class $\mathcal{F}=\{f:\mathcal{M}\to\{0,1\}\}$ is a tuple of four algorithms* $(\mathsf{Setup},\mathsf{Constrain},\mathsf{Sign},\mathsf{Vrfy})$.

$\mathsf{Setup}(1^\lambda)\to(\mathsf{vk},\mathsf{msk})$**:** *The setup algorithm is a PPT algorithm that takes as input the security parameter 1^λ, and outputs a master signing key* msk *and a verification key* vk.

$\mathsf{Constrain}(\mathsf{msk},f)\to\mathsf{sigk}_f$**:** *The constrain algorithm is a PPT algorithm that takes as input the master signing key* msk *and a constraint $f\in\mathcal{F}$, and outputs a constrained signing key* sigk_f.

[14] Though we can also define it for larger output length, we assume the output length to be 1 by default similarly to [HJO+16].

$\mathsf{Sign}(\mathsf{sigk}_f, m) \to \sigma$**:** *The signing algorithm is a PPT algorithm that takes as input a constrained key* sigk_f *and an message* $m \in \mathcal{M}$*, and outputs a signature* σ*.*

$\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) \to \top/\bot$**:** *The verification algorithm is a deterministic classical polynomial-time algorithm that takes as input a verification key* vk*, message* $m \in \mathcal{M}$*, and signature* σ*, and outputs* $\top$ *or* $\bot$*.*

Correctness: *For any* $m \in \mathcal{M}$ *and* $f \in \mathcal{F}$ *such that* $f(m) = 1$*, we have*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = \top : \begin{array}{l}(\mathsf{vk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sigk}_f \leftarrow \mathsf{Constrain}(\mathsf{msk}, f) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}_f, m)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 2.6 (Function Privacy). *For any* $m \in \mathcal{M}$*,* $f_0, f_1 \in \mathcal{F}$ *such that* $f_0(m) = f_1(m) = 1$*,* $(\mathsf{vk}, \mathsf{msk}) \in \mathsf{Setup}(1^\lambda)$*,* $\mathsf{sigk}_{f_0} \in \mathsf{Constrain}(\mathsf{msk}, f_0)$*, and* $\mathsf{sigk}_{f_1} \in \mathsf{Constrain}(\mathsf{msk}, f_1)$*, we require the following statistical indistinguishability:*

$$\sigma_0 \approx_s \sigma_1, \quad \textit{where} \quad \sigma_0 \leftarrow \mathsf{Sign}(\mathsf{sigk}_{f_0}, m), \quad \sigma_1 \leftarrow \mathsf{Sign}(\mathsf{sigk}_{f_1}, m).$$

Definition 2.7 (Selective single-key security). *We say that a CS scheme satisfies selective single-key security if for any stateful QPT* $\mathcal{A}$*, we have*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = \top \wedge f(m) = 0 : \begin{array}{l}f \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{vk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sigk}_f \leftarrow \mathsf{Constrain}(\mathsf{msk}, f) \\ (m, \sigma) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{sigk}_f)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

We introduce an additional property which we call coherent signability. Roughly, this ensures that a superposition of two constrained signing keys can be used to generate a signature almost without collapsing the state.

Definition 2.8 (Coherent-signability). *We say that a CS scheme is coherently-signable if for any polynomial* $L = L(\lambda)$*, there is a QPT algorithm* $\mathcal{QS}ign$ *that takes a quantum state* $|\psi\rangle$ *and a classical message* $m \in \mathcal{M}$ *and outputs a quantum state* $|\psi'\rangle$ *and a signature* σ*, satisfying the following:*

1. *For any family* $\{f_z \in \mathcal{F}\}_{z\in\{0,1\}^L}$*, for any* $z \in \{0,1\}^L$*,* $(\mathsf{vk}, \mathsf{msk}) \in \mathsf{Setup}(1^\lambda)$*,* $\mathsf{sigk}_{f_z} \in \mathsf{Constrain}(\mathsf{msk}, f_z)$*, and* $m \in \mathcal{M}$*, the output distribution of* $\mathcal{QS}ign\left(|z\rangle \left|\mathsf{sigk}_{f_z}\right\rangle, m\right)$ *is identical to that of* $\mathsf{Sign}(\mathsf{sigk}_{f_z}, m)$*.*
2. *For any family* $\{f_z \in \mathcal{F}\}_{z\in\{0,1\}^L}$*, any amplitudes* $\{\alpha_z\}_{z\in\{0,1\}^L}$ *such that* $\sum_{z\in\{0,1\}^L} |\alpha_z|^2 = 1$*, any* $(\mathsf{vk}, \mathsf{msk}) \in \mathsf{Setup}(1^\lambda)$*, any* $\mathsf{sigk}_{f_z} \in \mathsf{Constrain}(\mathsf{msk}, f_z)$*, and any message* $m \in \mathcal{M}$ *such that* $f_z(m) = 1$ *for every* z *with* $\alpha_z \neq 1$*, let* $|\psi\rangle$ *be the state* $|\psi\rangle = \sum_{z\in\{0,1\}^L} \alpha_z |z\rangle \left|\mathsf{sigk}_{f_z}\right\rangle$*. Suppose that we run* $(|\psi'\rangle, \sigma) \leftarrow \mathcal{QS}ign(|\psi\rangle, m)$*. Then we have*

$$\| |\psi\rangle\langle\psi| - |\psi'\rangle\langle\psi'| \|_{tr} = \mathsf{negl}(\lambda).$$

Lemma 2.9 (Adapted from [KMY25]). *If the SIS assumption holds for a subexponential modulus-to-target-norm-bound ratio, then there exists coherently-signable constrained signatures with selective security, function privacy, and coherent-signability that supports any constrained class representable by circuits of bounded arbitrary polynomial depth.*

We note that [KMY25] does not explicitly prove function privacy of their construction. However, this property follows almost immediately from the way their scheme is obtained: they apply the general conversion from homomorphic signatures to constrained signatures due to [Tsa17] to the homomorphic signature scheme of [GVW15]. Since their homomorphic signature already enjoys the context-hiding property, this yields the function private constrained signatures. More concretely, the construction satisfies function privacy because the signature distribution is negligibly close to one that depends only on the verification key and the message—denoted as $D_{\mathbb{Z}_q^m \cap \Lambda^\perp_{\mathbf{A}|\mathbf{BH}+\mathbf{C}}}$ in their notation (see Section 8.2 of [KMY25]).

2.2 Noisy Trapdoor Claw-Free Generators

Here, we introduce noisy trapdoor claw-free (NTCF) generators, which is an abstraction of NTCF functions [BCM+18, Mah18] and its associated algorithms. NTCF generators with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$ consist of the following algorithms:

$\mathsf{FuncGen}(1^\lambda, \mathsf{mode}) \to (\mathsf{pp}, \mathsf{td})$**:** The function generation algorithm is a PPT algorithm that takes a security parameter 1^λ and $\mathsf{mode} \in \{1,2\}$ and outputs a public parameter pp and a corresponding trapdoor td, where $\mathsf{mode} = 1$ indicates that the system is in "injective mode", and $\mathsf{mode} = 2$ indicates "two-to-one mode".

$\mathit{StateGen}(\mathsf{pp}) \to (y, |\psi\rangle)$ **:** The state generation algorithm is a QPT algorithm that takes a public parameter pp and outputs a classical string y and a quantum state $|\psi\rangle$.

$\mathsf{Check}(\mathsf{pp}, x, y) \to \top$ **or** $\bot$ **:** The verification algorithm is a deterministic algorithm that takes a public parameter pp and classical strings $x \in \mathcal{X}$ and y and outputs $\top$ indicating accept or $\bot$ indicating reject.

$\mathsf{Invert}(\mathsf{pp}, \mathsf{td}, y) \to x$ **or** (x_0, x_1) **or** $\bot$: The inversion algorithm is a PPT algorithm that takes as input a public parameter pp, a trapdoor td, and a classical string y and outputs a single classical string $x \in \mathcal{X}$, two classical strings $(x_0, x_1) \in \mathcal{X} \times \mathcal{X}$, or $\bot$ indicating that the inversion failed.

We highlight two differences from the original formulation of NTCF functions in [BCM+18, Mah18]. (i) In their definition, an input x is mapped to a distribution over outputs y via a function description (in our case, pp). In contrast, our formulation disregards the definition of the distribution and retains only a verification algorithm Check, which checks whether a pair (x, y) is consistent, meaning that x maps to y in the noiseless case, or that y belongs to the support associated with x in the noisy case. (ii) The original works define injective and two-to-one modes as separate families. However, we unify them into a single family, using the parameter mode to specify the operational mode.

Definition 2.10 (NTCF Generator). *Let* $(\mathsf{FuncGen}, \mathit{StateGen}, \mathsf{Check}, \mathsf{Invert})$ *be a set of algorithms. It is a NTCF generator if it satisfies the following requirements. In the following, let* $\mathcal{Y}_{\mathsf{pp}}$ *be*

$$\mathcal{Y}_{\mathsf{pp}} = \{y : \exists x \in \mathcal{X} \text{ s.t. } \mathsf{Check}(\mathsf{pp}, x, y) = \top\}.$$

- **Correctness of function generation and inversion in injective mode:** *For all* $(\mathsf{pp}, \mathsf{td}) \in \mathsf{FuncGen}(1^\lambda, 1)$ *and* $y \in \mathcal{Y}_{\mathsf{pp}}$*, there exists a unique* $x \in \mathcal{X}$ *such that* $\mathsf{Check}(\mathsf{pp}, x, y) = \top$*. Furthermore,* $\mathsf{Invert}(\mathsf{pp}, \mathsf{td}, y)$ *outputs* x*.*
- **Correctness of function generation and inversion in 2-to-1 mode:** *For all* $(\mathsf{pp}, \mathsf{td}) \in \mathsf{FuncGen}(1^\lambda, 2)$ *and* $y \in \mathcal{Y}_{\mathsf{pp}}$*, there exist distinct* $x_0, x_1 \in \mathcal{X}$ *such that* $\mathsf{Check}(\mathsf{pp}, x_0, y) = \mathsf{Check}(\mathsf{pp}, x_1, y) = \top$*, and no other* $x \notin \{x_0, x_1\}$ *satisfies* $\mathsf{Check}(\mathsf{pp}, x, y) = \top$*. Furthermore,* $\mathsf{Invert}(\mathsf{pp}, \mathsf{td}, y)$ *outputs* (x_0, x_1)*.*
- **Correctness of state generation:** *For all* pp *generated by* $\mathsf{FuncGen}(1^\lambda, \mathsf{mode})$*, we have*

  $$\Pr[y \in \mathcal{Y}_{\mathsf{pp}} : \mathit{StateGen}(\mathsf{pp}) \to (y, |\psi\rangle)] = 1 - \mathsf{negl}(\lambda).$$

  Furthermore, with overwhelming probability over the randomness of $\mathit{StateGen}(\mathsf{pp}) \to (y, |\psi\rangle)$*, we have*

  $$\| \, |\psi\rangle\langle\psi| - |\psi'\rangle\langle\psi'| \, \|_{tr} \leq \mathsf{negl}(\lambda)$$

  where

  $$|\psi'\rangle \stackrel{def}{=} \begin{cases} |x\rangle & \text{if } y \in \mathcal{Y}_{\mathsf{pp}} \text{ and } \mathsf{mode} = 1, \text{ where } x = \mathsf{Invert}(\mathsf{pp}, \mathsf{td}, y) \\ \frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle) & \text{if } y \in \mathcal{Y}_{\mathsf{pp}} \text{ and } \mathsf{mode} = 2, \text{ where } (x_0, x_1) = \mathsf{Invert}(\mathsf{pp}, \mathsf{td}, y) \\ \bot & \text{Otherwise.} \end{cases}$$

- **Indistinguishability of the modes** *For any QPT algorithm* $\mathcal{A}$*, we have*

  $$\left|\Pr\left[\mathcal{A}(\mathsf{pp}) = 1 : (\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{FuncGen}(1^\lambda, 1)\right] - \Pr\left[\mathcal{A}(\mathsf{pp}) = 1 : (\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{FuncGen}(1^\lambda, 2)\right]\right| \leq \mathsf{negl}(\lambda).$$

- **Adaptive hardcore bit security** *For all* $(\mathsf{pp},\mathsf{td}) \in \mathsf{FuncGen}(1^\lambda,2)$ *and* $w = \mathrm{poly}(\lambda)$ *such that* $\mathcal{X} \subseteq \{0,1\}^w$*, the following holds.*

  1. *There exists an efficient deterministic algorithm* $\mathsf{GoodSet}$ *that takes as input* td*,* y*, and* $d \in \{0,1\}^w$ *and outputs* $\top$ *or* $\bot$*. We require that for all* $(\mathsf{pp},\mathsf{td}) \in \mathsf{FuncGen}(1^\lambda,2)$*,* $y \in \mathcal{Y}_{\mathsf{pp}}$*,*

$$\Pr_{d\leftarrow\{0,1\}^w}[\mathsf{GoodSet}(\mathsf{td},y,d) = \top] \geq 1 - \mathsf{negl}(\lambda).$$

     In the case of $y \notin \mathcal{Y}_{\mathsf{pp}}$*,* $\mathsf{GoodSet}(\mathsf{td},y,d)$ *outputs* $\bot$ *for any* $d \in \{0,1\}^w$*.*

  2. *Let*

$$H_{\mathsf{pp},0} := \{(x_b,y,d) \mid b \in \{0,1\}, (x_0,x_1) = \mathsf{Invert}(\mathsf{pp},\mathsf{td},y), \mathsf{GoodSet}(\mathsf{td},y,d) = \top,\ \langle d,(x_0 \oplus x_1)\rangle = 0\},$$
$$H_{\mathsf{pp},1} := \{(x_b,y,d) \mid b \in \{0,1\}, (x_0,x_1) = \mathsf{Invert}(\mathsf{pp},\mathsf{td},y), \mathsf{GoodSet}(\mathsf{td},y,d) = \top,\ \langle d,(x_0 \oplus x_1)\rangle = 1\},$$

     then for any QPT $\mathcal{A}$*,*[15] *it holds that*

$$\left|\Pr_{(\mathsf{pp},\mathsf{td})\leftarrow\mathsf{FuncGen}(1^\lambda,2)}[\mathcal{A}(\mathsf{pp}) \in H_{\mathsf{pp},0}] - \Pr_{(\mathsf{pp},\mathsf{td})\leftarrow\mathsf{FuncGen}(1^\lambda,2)}[\mathcal{A}(\mathsf{pp}) \in H_{\mathsf{pp},1}]\right| \leq \mathsf{negl}(\lambda).$$

Lemma 2.11 ([Mah18]). *If the LWE assumption holds against QPT adversaries, there exists an NTCF generator.*

The following lemma is a variant of [HMNY21]. In our variant, the challenger provides $\mathcal{A}$ with the trapdoors corresponding to the set $\overline{S}$, whereas in [HMNY21], these trapdoors are not given. Therefore, the following statement is slightly stronger, but can be proven by the same proof.

Lemma 2.12 (Cut-and-Choose Adaptive Hardcore Property, adapted from Lemma 5.6 in [HMNY21]). *Let* $\mathsf{NTCF} = (\mathsf{FuncGen},\mathsf{StateGen},\mathsf{Check},\mathsf{Invert})$ *be an NTCF generator that satisfies the adaptive hardcore bit security. Then it satisfies what we call the* cut-and-choose adaptive hardcore property *defined below. For a QPT adversary* $\mathcal{A}$ *and a positive integer* n*, we consider the following experiment* $\mathsf{Exp}^{\mathsf{cut\text{-}and\text{-}choose}}_{\mathsf{NTCF},\mathcal{A}}(\lambda,n)$*.*

1. *The challenger chooses a uniform subset* $S \subseteq [2n]$ *such that* $|S| = n$*. In the following,* $\overline{S}$ *denotes* $[2n]\backslash S$*.*
2. *The challenger generates* $(\mathsf{pp}_i,\mathsf{td}_i) \leftarrow \mathsf{FuncGen}(1^\lambda,1)$ *for all* $i \in S$ *and* $(\mathsf{pp}_i,\mathsf{td}_i) \leftarrow \mathsf{FuncGen}(1^\lambda,2)$ *for all* $i \in \overline{S}$ *and sends* $\{\mathsf{pp}_i\}_{i\in[2n]}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *sends* $\{y_i,d_i\}_{i\in[2n]}$ *to the challenger.*
4. *The challenger computes* $(x_{i,0},x_{i,1}) \leftarrow \mathsf{Invert}(\mathsf{pp}_i,\mathsf{td}_i,y_i)$ *for all* $i \in \overline{S}$ *and checks if* $\mathsf{GoodSet}(\mathsf{td}_i,y_i,d_i) = \top$ *and* $d_i \cdot (x_{i,0} \oplus x_{i,1}) = 0$ *hold for all* $i \in \overline{S}$*. If they do not hold for some* $i \in \overline{S}$*, the challenger immediately aborts and the experiment returns* 0*.*
5. *The challenger sends* S *and* $\{\mathsf{td}_i\}_{i\in\overline{S}}$ *to* $\mathcal{A}$*.*
6. $\mathcal{A}$ *sends* $\{x_i\}_{i\in S}$ *to the challenger.*
7. *The challenger checks if* $\mathsf{Check}(\mathsf{pp}_i,x_i,y_i) = 1$ *holds for all* $i \in S$*. If this holds for all* $i \in S$*, the experiment returns* 1*. Otherwise, it returns* 0*.*

Then for any n *such that* $n \leq \mathrm{poly}(\lambda)$ *and* $n = \omega(\log\lambda)$*, it holds that*

$$\mathsf{Adv}^{\mathsf{cut\text{-}and\text{-}choose}}_{\mathsf{NTCF},\mathcal{A}}(\lambda,n) := \Pr\left[\mathsf{Exp}^{\mathsf{cut\text{-}and\text{-}choose}}_{\mathsf{NTCF},\mathcal{A}}(\lambda,n) = 1\right] \leq \mathsf{negl}(\lambda).$$

[15] Note that the original definition by [Mah18] applies a bijective map J to an element of $\mathcal{X}$ when taking the hardcore bit. We can remove this indirection caused by this map by directly defining their $J(\mathcal{X})$ as our $\mathcal{X}$. This slightly simplifies the notation.

3 Watermarkable Cryptographic Functionalities

Here, we introduce and construct several watermarkable cryptographic primitives. Specifically, we build watermarkable public-key encryption from plain public-key encryption, watermarkable unpredictable functions from one-way functions, and watermarkable digital signatures from SIS. Our constructions are inspired by the ideas implicit in [KMY25]; for the latter two primitives, however, we add new twists to handle evaluation and signing oracles in the security proofs.

3.1 Watermarkable Public Key Encryption

Syntax A watermarkable public key encryption scheme $\mathsf{WPKE} = (\mathsf{KG}, \mathsf{Mark}, \mathsf{Enc}, \mathsf{Dec})$ with the message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ and the mark space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$ consists of the following algorithms. In the following, we omit the subscript for $\mathcal{X}_\lambda$ and $\mathcal{M}_\lambda$ to denote $\mathcal{X}$ and $\mathcal{M}$ when it is clear from the context.

$\mathsf{KG}(1^\lambda) \to (\mathsf{ek}, \mathsf{msk})$**:** The key generation algorithm is a PPT algorithm that takes a security parameter 1^λ, and outputs a public key ek and a master secret key msk.

$\mathsf{Mark}(\mathsf{msk}, x) \to \mathsf{dk}(x)$**:** The marking algorithm is a deterministic classical polynomial time algorithm that takes a master secret key msk and a mark $x \in \mathcal{X}$, and outputs a marked secret key $\mathsf{dk}(x)$.

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

$\mathsf{Dec}(\mathsf{dk}(x), \mathsf{ct}) \to m$**:** The decryption algorithm is a deterministic classical polynomial-time algorithm that takes a marked decryption key $\mathsf{dk}(x)$ or a master secret key msk and a ciphertext ct, and outputs a value $\tilde{m}$.

Correctness: For every $m \in \mathcal{M}$ and $x \in \mathcal{X}$, we have

$$\Pr\left[\begin{array}{l|l} \mathsf{Dec}(\mathsf{dk}(x), \mathsf{ct}) = m & (\mathsf{ek}, \mathsf{msk}) \leftarrow \mathsf{KG}(1^\lambda) \\ \wedge & \mathsf{dk}(x) \leftarrow \mathsf{Mark}(\mathsf{msk}, x) \\ \mathsf{Dec}(\mathsf{msk}, \mathsf{ct}) = m & \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, m) \end{array}\right] = 1.$$

One-wayness: For all QPT adversary $\mathcal{A}$, we need

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

Parallel mark extractability: Here, we introduce the notion of "parallel mark extractability". To do so, we first define the following parallel one-way inversion experiment involving an adversary $\mathcal{A}$ and a challenger, parameterized by an arbitrary polynomial $n = \mathrm{poly}(\lambda)$, denoted as $\mathsf{Exp}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(1^\lambda)$.

1. The challenger generates $(\mathsf{ek}_i, \mathsf{msk}_i) \leftarrow \mathsf{KG}(1^\lambda)$ for $i \in [n]$ and sends $\{\mathsf{ek}_i\}_{i\in[n]}$ to $\mathcal{A}$.
2. Given $\{\mathsf{ek}_i\}_{i\in[n]}$, the adversary $\mathcal{A}$ chooses arbitrary $x_1, ..., x_n \in \mathcal{X}$ and sends them to the challenger.
3. The challenger computes $\mathsf{dk}_i(x_i) \leftarrow \mathsf{Mark}(\mathsf{msk}_i, x_i)$ for $i \in [n]$ and sends $\{\mathsf{dk}_i(x_i)\}_{i\in[n]}$ to $\mathcal{A}$.
4. For $i \in [n]$, the challenger generates $m_i \leftarrow \mathcal{M}$ and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i)$ and sends $\{\mathsf{ct}_i\}_{i\in[n]}$ to $\mathcal{A}$.
5. Given $\{\mathsf{ct}_i\}_{i\in[n]}$, $\mathcal{A}$ outputs $\{m'_i\}_{i\in[n]}$.
6. The experiment outputs 1 (indicating that $\mathcal{A}$ wins) if $m'_i = m_i$ for all $i \in [n]$ and 0 otherwise.

We define the advantage of a QPT adversary $\mathcal{A}$ to be

$$\mathsf{Adv}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(1^\lambda) = 1\right].$$

We also define parallel mark extraction experiment defined by a QPT extractor $\mathcal{Ext}$, an adversary $\mathcal{A}$, and a challenger, parameterized by an arbitrary polynomial $n = \mathrm{poly}(\lambda)$, denoted as $\mathsf{Exp}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{A},\mathcal{Ext},n}(1^\lambda)$. The parallel extraction experiment is the same as parallel inversion experiment except for the last 3 steps. Namely, we replace Step 4, 5, and 6 with the following:

4' Take the adversary $\mathcal{A}$ right before Step 4 of $\mathsf{Exp}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(1^\lambda)$. Since $\mathcal{A}$ is a quantum non-uniform QPT adversary, we can parse $\mathcal{A}$ as a pair of a unitary circuit $U_\mathcal{A}$ and a quantum state $\rho_\mathcal{A}$, where to run $\mathcal{A}$ on a set of ciphertexts $\{\mathsf{ct}_i\}_{i\in[n]}$, we apply $U_\mathcal{A}$ to $\{\mathsf{ct}_i\}_{i\in[n]}$ and $\rho_\mathcal{A}$ and then measure the corresponding registers to obtain the output $\{m'_i\}_{i\in[n]}$.

5' We run $\mathcal{Ext}(\{\mathsf{ek}_i\}_{i\in[n]}, (U_\mathcal{A}, \rho_\mathcal{A})) \to \{x'_i\}_{i\in[n]}$.

6' The experiment outputs 1 (indicating that the adversary wins) if $x'_i = x_i$ for all $i \in [n]$ and 0 otherwise.

We define the advantage of a QPT adversary $\mathcal{A}$ with respect to $\mathcal{Ext}$ to be

$$\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{A},\mathcal{Ext},n}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{A},\mathcal{Ext},n}(1^\lambda) = 1\right].$$

We say that WPKE satisfies parallel mark extractability if there exists a QPT algorithm $\mathcal{Ext}$ such that for all $n = \mathrm{poly}(\lambda)$, for all QPT algorithm $\mathcal{A}$, the following holds:

$$\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{A},\mathcal{Ext},n}(\lambda) \geq \mathsf{Adv}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(\lambda) - \mathsf{negl}(\lambda).$$

Construction The above defined watermarkable PKE can be constructed from any IND-CPA secure PKE scheme $\mathsf{PKE}.(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ for single-bit messages. Let $\ell = \omega(\lambda)$ be the bit-length of marks. Then we construct a watermarkable PKE scheme for message length ℓ as follows.

$\mathsf{KG}(1^\lambda)$**:** On input the security parameter 1^λ, the key generation algorithm proceeds as follows.

- Generate $(\mathsf{ek}_{j,b}, \mathsf{dk}_{j,b}) \leftarrow \mathsf{PKE.KG}(1^\lambda)$ for $j \in [\ell]$ and $b \in \{0,1\}$.
- Output $\mathsf{ek} = \{\mathsf{ek}_{j,b}\}_{j\in[\ell],b\in\{0,1\}}$ and $\mathsf{msk} = \{\mathsf{dk}_{j,b}\}_{j\in[\ell],b\in\{0,1\}}$.

$\mathsf{Mark}(\mathsf{msk}, x)$**:** On input $\mathsf{msk} = \{\mathsf{dk}_{j,b}\}_{j\in[\ell],b\in\{0,1\}}$ and $x \in \{0,1\}^\ell$, output $\mathsf{dk}(x) = (x, \{\mathsf{dk}_{j,x[j]}\}_{j\in[\ell]})$ where $x[j]$ is the j-th bit of x.

$\mathsf{Enc}(\mathsf{ek}, m)$**:** On input $\mathsf{ek} = \{\mathsf{ek}_{j,b}\}_{j\in[\ell],b\in\{0,1\}}$ and $m = m_1\|m_2\|\dots\|m_\ell$, the encryption algorithm proceeds as follows.

- Generate $\mathsf{ct}_{j,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_{j,b}, m_j)$ for $j \in [\ell]$ and $b \in \{0,1\}$.
- Output $\mathsf{ct} = \{\mathsf{ct}_{j,b}\}_{j\in[\ell],b\in\{0,1\}}$.

$\mathsf{Dec}(\mathsf{dk}(x), \mathsf{ct}) \to \tilde{m}$**:** On input a marked key $\mathsf{dk}(x) = (x, \{\mathsf{dk}'_j\}_{j\in[\ell]})$ or a master secret key $\mathsf{msk} = \{\mathsf{dk}_{j,b}\}_{j\in[\ell],b\in\{0,1\}}$ and $\mathsf{ct} = \{\mathsf{ct}_{j,b}\}_{j\in[\ell],b\in\{0,1\}}$, the decryption algorithm proceeds as follows.

- If the first input is a master secret key, set $\mathsf{dk}'_j = \mathsf{dk}_{j,0}$ and $x[j] = 0$ for all $j \in [\ell]$.
- Run $m'_j \leftarrow \mathsf{PKE.Dec}(\mathsf{dk}'_j, \mathsf{ct}_{j,x[j]})$ for $j \in [\ell]$.
- Output $m' = m'_1\|m'_2\|\dots\|m'_\ell$

Correctness and one-wayness of the above scheme immediately follows from correctness and IND-CPA security of PKE, respectively. Below, we prove parallel mark extractability.

Security proof

Theorem 3.1. *The above construction satisfies parallel mark extractability if* PKE *satisfies IND-CPA security.*

Proof. To prove the theorem, we first define the extraction algorithm $\mathcal{Ext}$ as follows.

$\mathcal{Ext}(\{\mathsf{ek}_i\}_{i\in[n]}, (U_{\mathcal{A}}, \rho_{\mathcal{A}})) \to \{x'_i\}_{i\in[n]}$**:** On input $\{\mathsf{ek}_i\}_{i\in[n]}$ and the description of the adversary $\mathcal{A} = (U_{\mathcal{A}}, \rho_{\mathcal{A}})$, it proceeds as follows.

- Parse the encryption key as $\mathsf{ek}_i = \{\mathsf{ek}_{i,j,b}\}_{j\in[\ell],b\in\{0,1\}}$ for each $i \in [n]$.
- Choose $m_{i,j,0} \leftarrow \{0,1\}$ and set $m_{i,j,1} = 1 - m_{i,j,0}$ for $i \in [n]$ and $j \in [\ell]$.
- Generate $\mathsf{ct}_{i,j,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_{i,j,b}, m_{i,j,b})$ for $i \in [n]$, $j \in [\ell]$, and $b \in \{0,1\}$.
- Set $\mathsf{ct}_i = \{\mathsf{ct}_{i,j,b}\}_{j\in[\ell],b\in\{0,1\}}$ for $i \in [n]$.
- Run the adversary $\mathcal{A}$ on input $\{\mathsf{ct}_i\}_{i\in[n]}$ (i.e., run the unitary $U_{\mathcal{A}}$ on input $(\{\mathsf{ct}_i\}_{i\in[n]}, \rho_{\mathcal{A}})$ and measure the output register) to obtain $\{m'_i\}_{i\in[n]}$
- Parse $m'_i = m'_i[1]\| \cdots \|m'_i[\ell]$ for $i \in [n]$.
- Let $x'_{i,j}$ be the unique bit such that $m'_i[j] = m_{i,j,x'_{i,j}}$ for $i \in [n]$ and $j \in [\ell]$.
- Let $x'_i = x'_{i,1}\| \ldots \|x'_{i,\ell}$ for $i \in [n]$.
- Output $\{x'_i\}_{i\in[n]}$.

We then prove the theorem by a sequence of hybrids. For the sake of the contradiction, let us consider an adversary $\mathcal{A}$ against the parallel one-way inversion experiment $\mathsf{Exp}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(1^\lambda)$. In the following, the event that $\mathcal{A}$ wins in $\mathsf{Hyb}_{\mathsf{xx}}$ is denoted by E_{xx}.

Hyb_0**:** This is the parallel predictability game between the challenger and an adversary $\mathcal{A}$. Namely, the game proceeds as follows.

1. The challenger runs $\mathsf{KG}(1^\lambda)$ to obtain ek_i for $i \in [n]$ as follows.
   - Generate $(\mathsf{ek}_{i,j,b}, \mathsf{dk}_{i,j,b}) \leftarrow \mathsf{PKE.KG}(1^\lambda)$ for $j \in [\ell]$, and $b \in \{0,1\}$.
   - Set $\mathsf{ek}_i = \{\mathsf{ek}_{i,j,b}\}_{j\in[\ell],b\in\{0,1\}}$.

   It then sends $\{\mathsf{ek}_i\}_{i\in[n]}$ to $\mathcal{A}$.
2. Given $\{\mathsf{ek}_i\}_{i\in[n]}$, the adversary $\mathcal{A}$ sends arbitrarily chosen $x_1, ..., x_n \in \{0,1\}^\ell$ to the challenger.
3. The challenger sets $\mathsf{dk}_i(x_i) = (x_i, \{\mathsf{dk}_{i,j,x_i[j]}\}_{j\in[\ell]})$ for $i \in [n]$ and sends $\{\mathsf{dk}_i(x_i)\}_{i\in[n]}$ to $\mathcal{A}$.
4. For $i \in [n]$, the challenger generates $m_i \leftarrow \{0,1\}^\ell$ and $\mathsf{ct}_{i,j,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_{i,j,b}, m_i[j])$ for $j \in [\ell]$ and $b \in \{0,1\}$, sets $\mathsf{ct}_i = \{\mathsf{ct}_{i,j,b}\}_{j\in[\ell],b\in\{0,1\}}$ and sends $\{\mathsf{ct}_i\}_{i\in[n]}$ to $\mathcal{A}$.
5. Receiving $\{\mathsf{ct}_i\}_{i\in[n]}$ from the challenger, $\mathcal{A}$ outputs $\{m'_i\}_{i\in[n]}$.
6. The challenger sets the output of the game to be 1 (indicating that $\mathcal{A}$ wins) if $m'_i = m_i$ for all $i \in [n]$ and 0 otherwise.

By definition, we have $\Pr[\mathsf{E}_0 = 1] = \mathsf{Adv}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(\lambda)$.

Hyb_1**:** In this hybrid, we change the way of generating $\mathsf{ct}_{i,j,1-x_i[j]}$ for $i \in [n]$ and $j \in [\ell]$. Specifically, it is generated as

$$\mathsf{ct}_{i,j,1-x_i[j]} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_{i,j,1-x_i[j]}, 1 - m_i[j]).$$

Note that $\mathsf{ct}_{i,j,x_i[j]}$ is generated as $\mathsf{ct}_{i,j,x_i[j]} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_{i,j,x_i[j]}, m_i[j])$ similarly to the previous hybrid.

Since $\mathsf{dk}_{i,j,1-x_i[j]}$ is not given to the adversary for any $i \in [n]$ and $j \in [\ell]$, by the IND-CPA security of PKE, we have $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\lambda)$. This can be shown using a hybrid argument over all positions of i and j. The reduction algorithm embeds its challenge instance into $\mathsf{ek}_{i,j,1-x_i[j]}$ by randomly guessing $x_i[j]$, where guess is correct with probability of $1/2$.

Hyb_2**:** In this hybrid, we change the way of generating $\mathsf{ct}_{i,j,b}$ for $i \in [n]$, $j \in [\ell]$, and $b \in \{0,1\}$, and the winning condition. In Step 4, the challenger chooses $m_{i,j,0} \leftarrow \{0,1\}$ and sets $m_{i,j,1} = 1 - m_{i,j,0}$ for $i \in [n]$ and $j \in [\ell]$. Then the challenger generates

$$\mathsf{ct}_{i,j,b} \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}_{i,j,b}, m_{i,j,b})$$

for $i \in [n]$, $j \in [\ell]$, and $b \in \{0,1\}$. In Step 6, the challenger sets the output of the game to be 1 if $m'_i[j] = m_{i,j,x_i[j]}$ for all $i \in [n]$ and $j \in [\ell]$ and 0 otherwise.

This is just a conceptual change, and thus we have $\Pr[\mathsf{E}_1 = 1] = \Pr[\mathsf{E}_2 = 1]$. It is also easy to see that we have $\Pr[\mathsf{E}_2 = 1] = \mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{A},\mathcal{Ext},n}(\lambda)$.

Combining the above, we obtain $\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{A},\mathcal{Ext},n}(\lambda) \geq \mathsf{Adv}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{A},n}(\lambda) - \mathsf{negl}(\lambda)$ as desired. □

3.2 Watermarkable Unpredictable Functions

Here, we define and construct watermarkable unpredictable functions (UPF). Its definition is very similar to watermarkable PKE above; if an adversary is given $\mathsf{key}(x)$ and generates a (possibly quantum) "predictor" that can guess the output on a uniformly random input with probability ϵ, then we can extract x from the predictor with probability $\epsilon - \mathsf{negl}(\lambda)$, and moreover the extraction should work even in the parallel setting. Based on the idea of [KMY25], we can construct such a watermarkable UPF from two-key equivocal PRFs [HJO+16], which in turn can be built from OWFs. To deal with evaluation queries in the security proof, which were not considered in [KMY25], we employ a simple trick using random bits.

Syntax A watermarkable unpredictable functions $\mathsf{WUPF} = (\mathsf{KG}, \mathsf{Mark}, \mathsf{Eval})$ with the input space $\{0,1\}^{\ell(\lambda)}$ and the mark space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$ consists of the following algorithms. In the following, we omit the security parameter λ for $\mathcal{X}_\lambda$ and $\ell(\lambda)$ to denote $\mathcal{X}$ and ℓ when it is clear from the context.

$\mathsf{KG}(1^\lambda) \to (\mathsf{msk}, \mathsf{xk})$**:** The key generation algorithm is a PPT algorithm that takes a security parameter 1^λ, and outputs a master secret key msk and an extraction key xk.

$\mathsf{Mark}(\mathsf{msk}, x) \to \mathsf{key}(x)$**:** The marking algorithm is a deterministic polynomial time algorithm that takes a master secret key msk and a mark $x \in \mathcal{X}$, and outputs a marked key $\mathsf{key}(x)$.

$\mathsf{Eval}(\mathsf{key}(x), s) \to t$**:** The evaluation algorithm is a PPT algorithm that takes a master secret key msk or marked key $\mathsf{key}(x)$ and an input $s \in \{0,1\}^\ell$, and outputs t.

Correctness: For every $s \in \{0,1\}^\ell$, we require that

$$\Pr\left[\begin{array}{l|l} \mathsf{Eval}(\mathsf{key}(x), s) = \mathsf{Eval}(\mathsf{msk}, s) & (\mathsf{msk}, \mathsf{xk}) \leftarrow \mathsf{KG}(1^\lambda) \\ \text{holds for all } x \in \mathcal{X} & \mathsf{key}(x) = \mathsf{Mark}(\mathsf{msk}, x) \text{ for all } x \in \mathcal{X} \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

One can consider a relaxed correctness requirement, where the equation $\mathsf{Eval}(\mathsf{key}(x), s) = \mathsf{Eval}(\mathsf{msk}, s)$ only needs to hold for a fixed x, as opposed to all x simultaneously as above. Nonetheless, our construction fulfills the stronger requirement.

Correctness against adversarially chosen mark and input: We also consider a strengthened version of the correctness. Here, we require the correctness to hold, even if the adversary chooses the mark and the input to the function adversarially. To define the notion, we define the correctness experiment involving an adversary $\mathcal{A}$, denoted as $\mathsf{Expt}^{\mathsf{corre}}_{\mathsf{WUPF},\mathcal{A}}(1^\lambda)$.

1. The challenger runs $\mathsf{KG}(1^\lambda) \to (\mathsf{msk}, \mathsf{xk})$.
2. The adversary $\mathcal{A}$ is given input 1^λ and starts to run. Throughout the game, it has an oracle access to an oracle $\mathsf{Eval}(\mathsf{msk}, \cdot)$, which takes as input $\hat{s} \in \{0,1\}^\ell$ and returns $\mathsf{Eval}(\mathsf{msk}, \hat{s})$.
3. At some point, $\mathcal{A}$ chooses arbitrary $x \in \mathcal{X}$.

4. Then, the challenger runs $\mathsf{Mark}(\mathsf{msk}, x) \to \mathsf{key}(x)$ and returns $\mathsf{key}(x)$ to $\mathcal{A}$.
5. The adversary continues to have access to the oracle. Finally, it outputs an input s. The output of the experiment is then set to be 1 if $\mathsf{Eval}(\mathsf{msk}, s) \neq \mathsf{Eval}(\mathsf{key}(x), s)$.

We say that WUPF has correctness against adversarially chosen input if for all QPT adversary $\mathcal{A}$, we have

$$\mathsf{Expt}^{\mathsf{corre}}_{\mathsf{WUPF},\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda).$$

Parallel mark extractability: Here, we introduce the notion of "parallel mark extractability", which is similar to that for watermarkable PKE. We first define the following parallel prediction experiment involving an adversary $\mathcal{A}$ and a challenger, parameterized by an arbitrary polynomial $n = \mathrm{poly}(\lambda)$, denoted as $\mathsf{Exp}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{A},n}(1^\lambda)$.

1. The challenger runs $\mathsf{KG}(1^\lambda) \to (\mathsf{msk}_i, \mathsf{xk}_i)$ for $i \in [n]$.
2. The adversary $\mathcal{A}$ is given input 1^λ and 1^n and starts to run. The adversary has an oracle access to the function $\mathsf{Eval}(\mathsf{msk}_i, \cdot)$ for each $i \in [n]$, which on input $\hat{s} \in \{0,1\}^\ell$ returns $\mathsf{Eval}(\mathsf{msk}_i, \hat{s})$. The adversary can make a query on any input at any timing arbitrarily many times, until $\mathcal{A}$ receives the challenge at Step 7.
3. At some point, $\mathcal{A}$ chooses arbitrary $x_1, ..., x_n \in \mathcal{X}$ and sends them to the challenger.
4. For $i \in [n]$, the challenger runs $\mathsf{key}_i(x_i) \leftarrow \mathsf{Mark}(\mathsf{msk}_i, x_i)$ and sends $\{\mathsf{key}_i(x_i)\}_{i\in[n]}$ to the adversary.
5. At some point, $\mathcal{A}$ declares that it is ready for the challenge.
6. The challenger picks uniformly random $s_i \leftarrow \{0,1\}^\ell$ for $i \in [n]$ and sends $\{s_i\}_{i\in[n]}$ to $\mathcal{A}$.
7. $\mathcal{A}$ receives $\{s_i\}_{i\in[n]}$ from the challenger and loses access to the evaluation oracles. It then outputs $\{t'_i\}_{i\in[n]}$.
8. The challenger sets the output of the experiment to be 1 (indicating that $\mathcal{A}$ wins) if $t'_i = \mathsf{Eval}(\mathsf{msk}_i, s_i)$ for all $i \in [n]$ and otherwise outputs 0.

We define the advantage of a QPT adversary $\mathcal{A}$ to be

$$\mathsf{Adv}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{A},n}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{A},n}(1^\lambda) = 1\right].$$

We also define parallel mark extraction experiment defined by a QPT extractor $\mathcal{E}\mathcal{X}\mathcal{t}$, an adversary $\mathcal{A}$, and a challenger, parameterized by an arbitrary polynomial $n = \mathrm{poly}(\lambda)$, denoted as $\mathsf{Exp}^{\mathsf{par\text{-}ext}}_{\mathsf{WUPF},\mathcal{A},\mathcal{E}\mathcal{X}\mathcal{t},n}(1^\lambda)$. The parallel extraction experiment is the same as parallel prediction experiment except for the last 3 steps. Namely, we replace Step 6, 7, and 8 with the following:

6' Take the adversary $\mathcal{A}$ right before Step 6 of $\mathsf{Exp}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{A},n}(1^\lambda)$. Since $\mathcal{A}$ is a quantum non-uniform QPT adversary, we can parse $\mathcal{A}$ as a pair of a unitary circuit $U_\mathcal{A}$ and a quantum state $\rho_\mathcal{A}$, where to run $\mathcal{A}$ on the inputs $\{s_i\}_{i\in[n]}$, we apply $U_\mathcal{A}$ to $\{s_i\}_{i\in[n]}$ and $\rho_\mathcal{A}$ and then measure the corresponding registers to obtain the output $\{t'_i\}_{i\in[n]}$.

7' We run $\mathcal{E}\mathcal{X}\mathcal{t}(\{\mathsf{xk}_i\}_{i\in[n]}, (U_\mathcal{A}, \rho_\mathcal{A})) \to \{x'_i\}_{i\in[n]}$.

8' The experiment outputs 1 (indicating that the adversary wins) if $x'_i = x_i$ for all $i \in [n]$ and 0 otherwise.

We define the advantage of a QPT adversary $\mathcal{A}$ with respect to $\mathcal{E}\mathcal{X}\mathcal{t}$ to be

$$\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WUPF},\mathcal{A},\mathcal{E}\mathcal{X}\mathcal{t},n}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{par\text{-}ext}}_{\mathsf{WUPF},\mathcal{A},\mathcal{E}\mathcal{X}\mathcal{t},n}(1^\lambda) = 1\right].$$

We say that WUPF satisfies parallel mark extractability if there exists a QPT algorithm $\mathcal{E}\mathcal{X}\mathcal{t}$ such that for all $n = \mathrm{poly}(\lambda)$, for all QPT algorithm $\mathcal{A}$, the following holds:

$$\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WUPF},\mathcal{A},\mathcal{E}\mathcal{X}\mathcal{t},n}(\lambda) \geq \mathsf{Adv}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{A},n}(\lambda) - \mathsf{negl}(\lambda).$$

Remark 3.2 (Public extraction vs. secret extraction). In our definition, unlike in watermarkable PKE in Section 3.1, the extraction key xk must be kept hidden from the adversary. This is often referred to as "secret extraction" in the watermarking literature and suffices for our purpose. While secret extraction would suffice also for PKE, we chose to formalize it with the public extraction since our construction satisfies it.

Remark 3.3 (Security as a plain UPF). We note that the above definition of a watermarkable UPF does not necessarily imply the standard notion of unpredictability as a plain UPF. Nevertheless, unpredictability can be added by a simple conversion: run a plain UPF in parallel and append its output alongside the original one. The marked key of the resulting scheme consists of that of the original one along with the secret key of the plain UPF.

We also introduce the notion of equivocality. This property will be used for the construction of watermarkable digital signatures using watermarkable UPF.

Equivocality: We define the equivocality experiment involving an adversary $\mathcal{A}$, an equivocation simulation algorithm Sim, and a challenger, denoted as $\mathsf{Expt}^{\mathsf{equiv}}_{\mathsf{WUPF},\mathcal{A}}(1^\lambda, \mathsf{coin})$.

1. The challenger runs $\mathsf{KG}(1^\lambda) \to (\mathsf{msk}, \mathsf{xk})$ if $\mathsf{coin} = 0$ and $\mathsf{Sim}(1^\lambda) \to \mathsf{key}^*$ if $\mathsf{coin} = 1$.
2. The adversary $\mathcal{A}$ is given input 1^λ and starts to run. Throughout the game, it has an oracle access to an oracle $\mathcal{O}(\cdot)$, which takes as input $s \in \{0,1\}^\ell$ and works as follows:
$$\mathcal{O}(s) = \begin{cases} \mathsf{Eval}(\mathsf{msk}, s) & \text{if } \mathsf{coin} = 0 \\ \mathsf{Eval}(\mathsf{key}^*, s), & \text{if } \mathsf{coin} = 1 \end{cases}$$
3. At some point, $\mathcal{A}$ chooses arbitrary $x \in \mathcal{X}$.
4. If $\mathsf{coin} = 0$, the challenger runs $\mathsf{Mark}(\mathsf{msk}, x) \to \mathsf{key}(x)$ and returns $\mathsf{key}(x)$ to $\mathcal{A}$. If $\mathsf{coin} = 1$, the challenger returns key^* to $\mathcal{A}$.
5. The adversary continues to have access to the oracle. Finally, it outputs its guess coin' for coin. The output of the experiment is then set to be coin'.

We say that WUPF has equivocality if there exists a classical PPT algorithm Sim such that for all QPT adversary $\mathcal{A}$, we have
$$\mathsf{Adv}^{\mathsf{equiv}}_{\mathsf{WUPF},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Expt}^{\mathsf{equiv}}_{\mathsf{WUPF},\mathcal{A}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{equiv}}_{\mathsf{WUPF},\mathcal{A}}(1^\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Remark 3.4. We can see that the equivocality implies the correctness against adversarially chosen mark and input, since if the adversary is able to find x and s such that $\mathsf{Eval}(\mathsf{msk}, s) \neq \mathsf{Eval}(\mathsf{key}(x), s)$ with non-negligible probability, these can be used for distinguishing the case of $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$ in the equivocality experiment. On the other hand, we do not know whether the correctness against adversarially chosen mark and input implies the correctness, since the latter requires the equation to hold for all $x \in \mathcal{X}$ simultaneously, which is somewhat strong condition.

Construction The above defined watermarkable UPF can be constructed from any two-key equivocal PRF $\mathsf{TEPRF} = \mathsf{TEPRF}.(\mathsf{KG}, \mathsf{Eval})$, which in turn can be constructed from any OWF. (See Definition 2.3 for the definition of two-key equivocal PRF.) Let ℓ be the bit-length of the inputs for TEPRF. We assume $\ell = \omega(\log \lambda)$. The bit-length of the marks for the construction can be set to an arbitrary polynomial $w = w(\lambda)$. The bit-length of the input of the construction then becomes ℓw.

$\mathsf{KG}(1^\lambda) \to (\mathsf{msk}, \mathsf{xk})$**:** On input the security parameter 1^λ, the key generation algorithm proceeds as follows.

- Sample $s_1^*, s_2^*, ..., s_w^* \leftarrow \{0,1\}^\ell$ and $c \leftarrow \{0,1\}^w$.
- Run $\mathsf{TEPRF.KG}(1^\lambda, s_j^*) \to (\mathsf{TEPRF.key}_{j,0}, \mathsf{TEPRF.key}_{j,1})$ for $j \in [w]$.
- Output $\mathsf{msk} = (c, \{\mathsf{TEPRF.key}_{j,b}\}_{j\in[w],b\in\{0,1\}})$ and $\mathsf{xk} = (c, \{s_j^*\}_{j\in[w]}, \{\mathsf{TEPRF.key}_{j,b}\}_{j\in[w],b\in\{0,1\}})$.

$\mathsf{Mark}(\mathsf{msk}, x) \to \mathsf{key}(x)$**:** Upon input the master secret key $\mathsf{msk} = (c, \{\mathsf{TEPRF.key}_{j,b}\}_{j\in[w],b\in\{0,1\}})$ and a mark $x \in \{0,1\}^w$, it outputs $\mathsf{key}(x) = \{\mathsf{TEPRF.key}_{j,x[j]\oplus c[j]}\}_{j\in[w]}$ where $x[j]$ and $c[j]$ are the j-th bit of x and c respectively.

$\mathsf{Eval}(\mathsf{key}(x), s) \to t$**:** On input the marked key $\mathsf{key}(x)$ or the master secret key msk, the evaluation algorithm proceeds as follows.

- If the first input is a master secret key, parse it as $\mathsf{msk} = (c, \{\mathsf{TEPRF.key}_{j,b}\}_{j\in[w],b\in\{0,1\}})$. Then, set $\mathsf{TEPRF.key}'_j = \mathsf{TEPRF.key}_{j,0}$ for all $j \in [w]$.
- If the first input is a marked key $\mathsf{key}(x)$, parse it as $\mathsf{key}(x) = \{\mathsf{TEPRF.key}'_j\}_{j\in[w]}$.
- Parse the input as $s = s_1\|s_2\|\dots\|s_w$.
- For $j \in [w]$, compute $t_j \leftarrow \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}'_j, s_j)$, where each block is of length ℓ. Then, output $t = t_1\|t_2\|\dots\|t_w$.

Correctness For an input $s = s_1\|\cdots\|s_w$, unless $s_i = s_i^*$ for some $i \in [w]$, it holds that $\mathsf{Eval}(\mathsf{key}(x), s) = \mathsf{Eval}(\mathsf{msk}, s)$ for all x. By the union bound, the event $s_i = s_i^*$ for some i occurs with probability at most $w2^{-\ell}$, which is negligible if $\ell = \omega(\log\lambda)$.

Security Proof The following theorem asserts the parallel mark extractability of the above construction.

Theorem 3.5. *The above construction satisfies parallel mark extractability if $\ell(\lambda) = \omega(\log\lambda)$ and* TEPRF *satisfies the different values on target input property and the differing point hiding property.*

Before presenting the proof, we briefly outline the intuition. Our construction builds on [KMY25], with the main difference that we introduce "equivocating bits" c to simulate evaluation oracles. The core idea for simulating the evaluation oracle is to evaluate the function using the marked key instead of the master secret key, where the latter is not available to the reduction. This approach seems to work, since the adversary cannot issue evaluation queries on points that would distinguish the simulated oracle from the real one by the differing point hiding property of the underlying TEPRF. A difficulty arises, however, because the marks are determined by the adversary through its evaluation queries. If we follow the naive strategy, this makes it impossible to simulate the evaluation oracle before the marks are fixed. To overcome this, we employ the "equivocating bits" c. They allow us to use $\{\mathsf{TEPRF.key}_{i,j,c_i^*[j]}\}_{j\in[w]}$ for random c^* as if it were a marked key, even before the actual mark x is chosen. Once x is fixed, we can equivocate by interpreting c^* as $c \oplus x$.

Proof. To prove the theorem, we first define the extraction algorithm $\mathcal{E}\mathcal{X}\mathcal{t}$ as follows.

$\mathcal{E}\mathcal{X}\mathcal{t}(\{\mathsf{xk}_i\}_{i\in[n]}, (U_{\mathcal{A}}, \rho_{\mathcal{A}})) \to \{x'_i\}_{i\in[n]}$**:** Given the extraction keys $\{\mathsf{xk}_i\}_{i\in[n]}$ and the description of the adversary $\mathcal{A} = (U_{\mathcal{A}}, \rho_{\mathcal{A}})$, it proceeds as follows.

- Parse the extraction key as $\mathsf{xk}_i = (c_i, s_i^* = \{s_{i,j}^*\}_{j\in[w]}, \{\mathsf{TEPRF.key}_{i,j,b}\}_{j\in[w],b\in\{0,1\}})$ for each $i \in [n]$, where $c_i \in \{0,1\}^w$, $s_{i,j}^* \in \{0,1\}^\ell$.
- Run the adversary $\mathcal{A}$ on input $(s_1^*, \dots, s_n^*)$ (i.e., run the unitary $U_{\mathcal{A}}$ on input $((s_1^*, \dots, s_n^*), \rho_{\mathcal{A}})$ and measure the output register) to obtain $t'_1, \dots, t'_n$.
- Define $x'_i = x'_i[1]\|\cdots\|x'_i[w]$ as

$$x'_i[j] = d_i[j] \oplus c_i[j], \quad \text{where} \quad d_i[j] = \begin{cases} 0 & \text{if } t'_{i,j} = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{i,j,0}, s_{i,j}^*) \\ 1 & \text{otherwise .} \end{cases}$$

- Output $\{x'_i\}_{i\in[n]}$.

We then prove the theorem by a sequence of hybrids. For the sake of the contradiction, let us consider an adversary $\mathcal{A}$ against the parallel prediction game $\mathsf{Exp}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{A},n}(1^\lambda)$. In the following, the event that $\mathcal{A}$ wins in $\mathsf{Hyb}_{\mathsf{xx}}$ is denoted by E_{xx}.

Hyb_0**:** This is the parallel prediction game between the challenger and an adversary $\mathcal{A}$. Namely, the game proceeds as follows.

1. The challenger runs $\mathsf{KG}(1^\lambda)$ to obtain $(\mathsf{msk}_i, \mathsf{xk}_i)$ for $i \in [n]$ as follows.
   - Sample $s^*_{i,1}, s^*_{i,2}, ..., s^*_{i,w} \leftarrow \{0,1\}^\ell$ and $c_i \leftarrow \{0,1\}^w$.
   - Run $\mathsf{TEPRF.KG}(1^\lambda, s^*_{i,j}) \to (\mathsf{TEPRF.key}_{i,j,0}, \mathsf{TEPRF.key}_{i,j,1})$ for $j \in [w]$.
   - Set $\mathsf{msk}_i = (c_i, \{\mathsf{TEPRF.key}_{i,j,b}\}_{j\in[w],b\in\{0,1\}})$ and $\mathsf{xk}_i = (c_i, \{s^*_{i,j}\}_{j\in[w]}, \{\mathsf{TEPRF.key}_{i,j,b}\}_{j\in[w],b\in\{0,1\}})$.
2. The adversary $\mathcal{A}$ is given input 1^λ and 1^n and starts to run. Throughout the game, the adversary has an oracle access to the function $\mathsf{Eval}(\mathsf{msk}_i, \cdot)$, until it receives the challenge inputs. Recall that given an input $\hat{s} = \hat{s}_1 \| \hat{s}_2 \| \dots \| \hat{s}_w$, $\mathsf{Eval}(\mathsf{msk}_i, \hat{s})$ computes
$$\hat{t}_j = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{i,j,0}, \hat{s}_j) \tag{3}$$
for $j \in [w]$ and returns $\hat{t} = \hat{t}_1 \| \hat{t}_2 \| \dots \| \hat{t}_w$.
3. At some point, $\mathcal{A}$ chooses arbitrary $x_1, ..., x_n \in \mathcal{X}$ and sends them to the challenger.
4. For $i \in [n]$, the challenger sets $\mathsf{key}_i = \{\mathsf{TEPRF.key}_{i,j,x_i[j]\oplus c_i[j]}\}_{j\in[w]}$ and sends $\{\mathsf{key}_i\}_{i\in[n]}$ to the adversary.
5. At some point, the adversary declares that it is ready for the challenge.
6. The challenger picks uniformly random $s_i = s_{i,1} \| \cdots \| s_{i,w} \leftarrow \{0,1\}^{\ell w}$ for $i \in [n]$ and sends $\{s_i\}_{i\in[n]}$ to the adversary.
7. Receiving $\{s_i\}_{i\in[n]}$ from the challenger, $\mathcal{A}$ outputs $\{t'_i\}_{i\in[n]}$. Note that after receiving $\{s_i\}_{i\in[n]}$ from the challenger, $\mathcal{A}$ loses access to the oracles.
8. The challenger sets the output of the game to be 1 (indicating that the adversary wins) if $t'_i = t_i$ for all $i \in [n]$ and otherwise outputs 0, where we define $t_i = \mathsf{Eval}(\mathsf{msk}_i, s_i)$. By the definition of $\mathsf{Eval}(\mathsf{msk}_i, s_i)$,
$$t_i = t_{i,1} \| t_{i,2} \| \dots \| t_{i,w} \quad \text{where} \quad t_{i,j} = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{i,j,0}, s_{i,j}).$$

Hyb_1**:** In this hybrid, we change the game so that the challenger does not choose c_i at Step 1 any more. Therefore, msk_i and xk_i are undefined. Instead, $c^*_i \leftarrow \{0,1\}^w$ is chosen in Step 1 and key_i is defined as $\mathsf{key}_i = \{\mathsf{TEPRF.key}_{i,j,c^*_i[j]}\}_{j\in[w]}$ in Step 4. We can see that Hyb_1 is equivalent to Hyb_0 if c^*_i is replaced with $c^*_i = c_i \oplus x_i$. This is a conceptual change and thus we have $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_0]$.

Hyb_2**:** In this hybrid, we change the winning condition of the adversary. Namely, in Step 8, the challenger outputs 1 if $t'_i = t_i$ for all $i \in [n]$, where we now define t_i as $t_i = \mathsf{Eval}(\mathsf{key}_i, s_i)$. By the definition of $\mathsf{Eval}(\mathsf{key}_i, s_i)$, we have
$$t_i = t_{i,1} \| t_{i,2} \| \dots \| t_{i,w} \quad \text{where} \quad t_{i,j} = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{i,j,c^*_i[j]}, s_{i,j}).$$
By the equality of TEPRF, the value of $t_{i,j}$ is different from the one in the previous hybrid only when $c^*_i[j] = 1$ and $s_{i,j} = s^*_{i,j}$. However, the latter occurs only with negligible probability since $s_{i,j}$ is chosen uniformly at random, independently from anything else. We therefore have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda)$.

Hyb_3**:** In this hybrid, we change the way the evaluation oracles answer the queries. Namely, in this hybrid, the i-th evaluation oracle on input $\hat{s} = \hat{s}_1 \| \hat{s}_2 \| \dots \| \hat{s}_w$ computes $\hat{t}_j$ for $j \in [w]$ as
$$\hat{t}_j = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{i,j,c^*_i[j]}, \hat{s}_j)$$
instead of as Equation (3), and returns $\hat{t} = \hat{t}_1 \| \hat{t}_2 \| \dots \| \hat{t}_w$ to $\mathcal{A}$.

We claim that this hybrid is indistinguishable from the previous hybrid. To see this, let us observe that Hyb_2 and Hyb_3 differ only when $\mathcal{A}$ makes an evaluation query for $\hat{s}$ with $\hat{s}_j = s^*_{i,j}$ to the i-th evaluation oracle for some $i \in [n]$ and $j \in [w]$. However, we can show that such an event occurs only with negligible probability by a straightforward reduction to the hard-to-find differing-point property of TEPRF, by noting that we only need $\mathsf{TEPRF.key}_{i,j,c^*_i[j]}$ for each i, j for simulating Hyb_3 (i.e., $\mathsf{TEPRF.key}_{i,j,1-c^*_i[j]}$ is not needed). We therefore have $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\lambda)$.

Hyb_4**:** In this hybrid, we set the challenge inputs to be $s^*_i = s^*_{i,1} \| s^*_{i,2} \| \cdots \| s^*_{i,w}$ for $i \in [n]$ at Step 6. Accordingly, to check the winning condition in Step 8, t_i is set as $t_i = \mathsf{Eval}(\mathsf{key}_i, s^*_i)$ for $i \in [n]$. Note that $s_{i,1}, s_{i,2}, ..., s_{i,w}$ are no longer necessary to run the game.

We claim that this hybrid is indistinguishable from the previous hybrid. This can be shown by a reduction to the differing-point-hiding property of TEPRF, where we replace the challenge input for each position of i, j one by one. The reduction starts by choosing random $(s^0_{i,j}, s^1_{i,j}, c^*_i[j])$ and submitting this to its challenger and is given $\mathsf{TEPRF.key}_{i,j,c^*_i[j]}$, where the differing point is either $s^0_{i,j}$ or $s^1_{i,j}$. The reduction uses $s^1_{i,j}$ as a challenge input. In the case $\mathsf{TEPRF.key}_{i,j,c^*_i[j]}$ is generated using $s^0_{i,j}$, the challenge input for the position is simulated as Hyb_3 (i.e., the differing point and challenge input are independent for the position), while Hyb_4 (i.e., the differing point and challenge input are the same) otherwise. The rest of the reduction is straightforward, since Hyb_3 and Hyb_4 can be run only using $\mathsf{TEPRF.key}_{i,j,c^*_i[j]}$ for each i, j. We therefore have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| = \mathsf{negl}(\lambda)$.

Hyb_5**:** In this hybrid, we switch the way the evaluation oracles are answered to the original one. Namely, on input $\hat{s}$, the i-th evaluation oracle computes $\hat{t}_j$ as Equation (3) again.

Similarly to the case of Hyb_2 and Hyb_3, we can show that this hybrid is indistinguishable from the previous hybrid by the hard-to-find differing-point property of TEPRF. Here again, we have to bound the probability that $\mathcal{A}$ makes an evaluation query for $\hat{s}$ with $\hat{s}_j = s^*_{i,j}$ for some $j \in [w]$. One additional subtlety that arises here is that the reduction algorithm does not know $s^*_{i,j}$ and thus it cannot simulate the challenge input for the adversary. However, this is not a problem because it is sufficient for the reduction to simulate the game right before the challenge input is given to $\mathcal{A}$, since the evaluation queries are made only before that. We therefore have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| = \mathsf{negl}(\lambda)$.

Hyb_6**:** In this hybrid, we undo the changes introduced in Hyb_1. Namely, the challenger chooses c_i for $i \in [n]$ in Step 1 again. The challenger then sets $\mathsf{key}_i = \{\mathsf{TEPRF.key}_{i,j,x_i[j] \oplus c_i[j]}\}_{j \in [w]}$ for $i \in [n]$ in Step 4 and uses it for checking the winning condition in Step 8. We can easily see that this is just a conceptual change and thus we have $\Pr[\mathsf{E}_6] = \Pr[\mathsf{E}_5]$.

Hyb_7**:** In this hybrid, we further change the winning condition in Step 8 again. Namely, from the output $\{t'_i = t'_{i,1} \| \cdots \| t'_{i,w}\}_{i \in [n]}$ of $\mathcal{A}$, it computes $\{x'_i = x'_i[1] \| \cdots \| x'_i[w]\}_{i \in [n]}$ as follows:

$$x'_i[j] = d_i[j] \oplus c_i[j], \quad \text{where} \quad d_i[j] = \begin{cases} 0 & \text{if } t'_{i,j} = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{i,j,0}, s^*_{i,j}) \\ 1 & \text{otherwise .} \end{cases}$$

Then, the challenger sets the output of the game to be 1 if $x'_i = x_i$ for all $i \in [n]$.

We claim that this change does not reduce the winning probability of $\mathcal{A}$. To see this, we recall that when the adversary wins the game in the previous hybrid, $t'_{i,j} = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{i,j,x_i[j] \oplus c_i[j]}, s^*_{i,j})$ holds for all i and j. In such a case, we have $x'_i = x_i$ for all i, since we have $d_i[j] = x_i[j] \oplus c_i[j]$ by the different values on target property of TEPRF. We therefore have $\Pr[\mathsf{E}_7] \geq \Pr[\mathsf{E}_6]$.

From the above discussion, we can see that $\Pr[\mathsf{E}_7] \geq \Pr[\mathsf{E}_0] - \mathsf{negl}(\lambda)$ holds. Furthermore, one can see by inspection that Hyb_7 corresponds to the parallel extraction experiment $\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WUPF},\mathcal{A},\mathcal{EXT},n}(\lambda)$ for $\mathcal{A}$ with respect to $\mathcal{EXT}$ we defined. The theorem therefore follows. □

We also prove the equivocality of the construction.

Theorem 3.6. *The above construction satisfies equivocality if* TEPRF *satisfies hard-to-find differing-point property.*

Proof. To prove the theorem, we first define the equivocation simulator Sim as follows.

$\mathsf{Sim}(1^\lambda) \to \mathsf{key}^*$**:** Given the security parameter 1^λ, it proceeds as follows.

- Sample $s_1^*, s_2^*, \ldots, s_w^* \leftarrow \{0,1\}^\ell$ and $c^* \leftarrow \{0,1\}^w$.
- Run $\mathsf{TEPRF.KG}(1^\lambda, s_j^*) \to (\mathsf{TEPRF.key}_{j,0}, \mathsf{TEPRF.key}_{j,1})$ for $j \in [w]$.
- Set $\mathsf{key}^* = \{\mathsf{TEPRF.key}_{j,c^*[j]}\}_{j\in[w]}$.

We then prove the theorem by a sequence of hybrids. For the sake of the contradiction, let us consider an adversary $\mathcal{A}$ against the equivocality game $\mathsf{Expt}^{\mathsf{equiv}}_{\mathsf{WUPF},\mathcal{A}}(1^\lambda)$. In the following, the event that $\mathcal{A}$ outputs 1 in $\mathsf{Hyb}_{\mathsf{xx}}$ is denoted by E_{xx}.

Hyb_0**:** This is the equivocality game between the challenger and an adversary $\mathcal{A}$ with $\mathsf{coin} = 0$. Namely, the game proceeds as follows.

1. The challenger runs $\mathsf{KG}(1^\lambda)$ to obtain $(\mathsf{msk}, \mathsf{xk})$ as follows.
   - Sample $s_1^*, s_2^*, \ldots, s_w^* \leftarrow \{0,1\}^\ell$ and $c \leftarrow \{0,1\}^w$.
   - Run $\mathsf{TEPRF.KG}(1^\lambda, s_j^*) \to (\mathsf{TEPRF.key}_{j,0}, \mathsf{TEPRF.key}_{j,1})$ for $j \in [w]$.
   - Set $\mathsf{msk} = (c, \{\mathsf{TEPRF.key}_{j,b}\}_{j\in[w],b\in\{0,1\}})$ and $\mathsf{xk} = (c, \{s_j^*\}_{j\in[w]})$.
2. The adversary $\mathcal{A}$ is given input 1^λ and starts to run. Throughout the game, the adversary has an oracle access to the function $\mathsf{Eval}(\mathsf{msk}, \cdot)$. Recall that given an input $\hat{s} = \hat{s}_1 \| \hat{s}_2 \| \ldots \| \hat{s}_w$, $\mathsf{Eval}(\mathsf{msk}, \hat{s})$ computes
$$\hat{t}_j = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{j,0}, \hat{s}_j)$$
for $j \in [w]$ and returns $\hat{t} = \hat{t}_1 \| \hat{t}_2 \| \ldots \| \hat{t}_w$.
3. At some point, $\mathcal{A}$ chooses arbitrary $x \in \mathcal{X}$ and sends it to the challenger.
4. The challenger sets $\mathsf{key} = \{\mathsf{TEPRF.key}_{j,x[j]\oplus c[j]}\}_{j\in[w]}$ and sends key to the adversary.
5. The adversary continues to have access to the oracle. Finally, it outputs its guess coin' for coin. The output of the experiment is then set to be coin'.

Hyb_1**:** In this hybrid, we choose $c^* \leftarrow \{0,1\}^w$ at Step 1 of the game. Furthermore, at Step 4 of the game, the challenger returns $\mathsf{key} = \{\mathsf{TEPRF.key}_{j,c^*[j]}\}_{j\in[w]}$ to the adversary. We can see that this change is conceptual, since no information of c is revealed except for Step 4 of the game in the previous hybrid and thus c effectively works as a one-time pad. We therefore have $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_0]$.

Hyb_2**:** In this hybrid, we change the evaluation oracle so that $\hat{t}_j$ for $j \in [w]$ is defined as
$$\hat{t}_j = \mathsf{TEPRF.Eval}(\mathsf{TEPRF.key}_{j,c^*[j]}, \hat{s}_j).$$
We claim that this change is unnoticed by $\mathcal{A}$ except for a negligible probability. To see this, we first observe that by the equality of TEPRF, the response from the oracle in this hybrid differs from that of the previous game only when $\mathcal{A}$ queries on input $\hat{s}$ such that $\hat{s}_j = s_j^*$ for some $j \in [w]$. However, we can show that such an event occurs only with negligible probability by a straightforward reduction to the hard-to-find differing-point property of TEPRF, by noting that we only need $\mathsf{TEPRF.key}_{j,c^*[j]}$ for each j for simulating Hyb_2 (i.e., $\mathsf{TEPRF.key}_{j,1-c^*[j]}$ is not needed). We therefore have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda)$.

We can observe that Hyb_2 is identical to the equivocality game between the challenger and an adversary $\mathcal{A}$ with $\mathsf{coin} = 1$. Furthermore, from the above discussion, we know $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda)$. Therefore, the theorem follows. □

3.3 Watermarkable Digital Signatures

The definition of watermarkable digital signatures (DS) is very similar to that of watermarkable PKE in Section 3.1; if an adversary is given a marked signing key $\mathsf{sigk}(x)$ and generates a (possibly quantum) "signer" that can sign a uniformly random message with probability ϵ, then we can extract x from the signer with probability $\epsilon - \mathsf{negl}(\lambda)$, and moreover the extraction should work even in the parallel setting. We generalize the idea of [KMY25] and show a generic construction of watermarkable DS from coherently signable constrained signatures and the watermarkable UPF, which in turn can be based on the SIS assumption. Similarly to the case of watermarkable UPF, we must also account for signing queries in the security proof, a scenario not addressed in [KMY25]. This introduces some technicalities in the proof, which can be resolved by exploiting the equivocality property of watermarkable UPF.

Syntax A watermarkable DS scheme $\mathsf{WDS} = (\mathsf{KG}, \mathsf{Mark}, \mathsf{Sign}, \mathsf{Vrfy})$ with the message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ and the mark space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$ consists of the following algorithms. In the following, we omit the security parameter λ for $\mathcal{X}_\lambda$ and $\ell(\lambda)$ to denote $\mathcal{X}$ and ℓ when it is clear from the context.

$\mathsf{KG}(1^\lambda) \to (\mathsf{vk}, \mathsf{msk}, \mathsf{xk})$**:** The key generation algorithm is a PPT algorithm that takes a security parameter 1^λ, and outputs a verification key vk, master secret key msk, and an extraction key xk.

$\mathsf{Mark}(\mathsf{msk}, x) \to \mathsf{sigk}(x)$**:** The marking algorithm is a PPT algorithm that takes a master secret key msk and a mark $x \in \mathcal{X}$, and outputs a marked signing key $\mathsf{sigk}(x)$.

$\mathsf{Sign}(\mathsf{sigk}(x), m) \to \sigma$**:** The signing algorithm is a PPT algorithm that takes a master secret key msk or marked signing key $\mathsf{sigk}(x)$ and a message $m \in \mathcal{M}$, and outputs a signature σ.

$\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) \to \top/\bot$**:** The verification algorithm is a PPT algorithm that takes a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, and outputs $\top$ or $\bot$.

Correctness: For every $x \in \mathcal{X}$ and $m \in \mathcal{M}$, we have

$$\Pr\left[\begin{array}{c|l} \mathsf{Vrfy}(\mathsf{vk}, m, \sigma) = \top & (\mathsf{vk}, \mathsf{msk}, \mathsf{xk}) \leftarrow \mathsf{KG}(1^\lambda) \\ \wedge & \mathsf{sigk}(x) \leftarrow \mathsf{Mark}(\mathsf{msk}, x) \\ \mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = \top & \sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}(x), m) \\ & \sigma' \leftarrow \mathsf{Sign}(\mathsf{msk}, m) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Coherent signability We say that a watermarkable DS scheme is coherently-signable if for any polynomial $L = L(\lambda)$, there is a QPT algorithm $\mathcal{QSign}$ that takes a quantum state $|\psi\rangle$ and a classical message $m \in \mathcal{M}$ and outputs a quantum state $|\psi'\rangle$ and a signature σ, satisfying the following:

1. For any family $\{x_z \in \mathcal{X}\}_{z\in\{0,1\}^L}$, for any $z \in \{0,1\}^L$, $m \in \mathcal{M}$, $(\mathsf{vk}, \mathsf{msk}, \mathsf{xk}) \in \mathsf{Supp}(\mathsf{KG}(1^\lambda))$, and $\mathsf{sigk}(x_z) \in \mathsf{Supp}(\mathsf{Mark}(\mathsf{msk}, x_z))$, the output distribution of $\mathcal{QSign}(|z\rangle\,|\mathsf{sigk}(x_z)\rangle, m)$ is identical to that of $\mathsf{Sign}(\mathsf{sigk}(x_z), m)$.
2. For any families $\{\alpha_z \in \mathbb{C}\}_{z\in\{0,1\}^L}$ such that $\sum_{z\in\{0,1\}^L} |\alpha_z|^2 = 1$ and $\{x_z \in \mathcal{X}\}_{z\in\{0,1\}^L}$ and for any $m \in \mathcal{M}$, the following holds: For overwhelming probability over the choice of $(\mathsf{vk}, \mathsf{msk}, \mathsf{xk}) \leftarrow \mathsf{KG}(1^\lambda)$ and for all $\mathsf{sigk}(x_z) \in \mathsf{Supp}(\mathsf{Mark}(\mathsf{msk}, x_z))$, we have

$$\| |\psi\rangle\langle\psi| - |\psi'\rangle\langle\psi'| \|_{tr} = \mathsf{negl}(\lambda), \quad \text{where} \quad |\psi\rangle = \sum_{z\in\{0,1\}^L} \alpha_z |z\rangle |\mathsf{sigk}(x_z)\rangle, \quad (|\psi'\rangle, \sigma) \leftarrow \mathcal{QSign}(|\psi\rangle, m).$$

Remark 3.7. The coherent-signability is required so that the DS-SKL scheme we construct from the watermarkable DS has reusability of quantum signing key, which states that the quantum signing key can be repeatedly used to sign on multiple messages.

Parallel mark extractability: Here, we introduce the notion of "parallel mark extractability", which is similar to that for watermarkable PKE. We first define the following parallel forgery experiment involving an adversary $\mathcal{A}$ and a challenger, parameterized by an arbitrary polynomial $n = \mathrm{poly}(\lambda)$, denoted as $\mathsf{Exp}^{\mathsf{par\text{-}for}}_{\mathsf{WDS},\mathcal{A},n}(1^\lambda)$.

1. The challenger runs $\mathsf{KG}(1^\lambda) \to (\mathsf{vk}_i, \mathsf{msk}_i, \mathsf{xk}_i)$ for $i \in [n]$ and sends $\{\mathsf{vk}_i\}_{i\in[n]}$ to $\mathcal{A}$.
2. Throughout the game, $\mathcal{A}$ has an oracle access to the signing oracle $\mathsf{Sign}(\mathsf{msk}_i,)$ for each $i \in [n]$, until it receives the challenge messages in Step 7. When queried on input $\hat{m} \in \mathcal{M}$, the oracles $\mathsf{Sign}(\mathsf{msk}_i,)$ runs $\mathsf{Sign}(\mathsf{msk}_i, \hat{m}) \to \hat{\sigma}$ and returns $\hat{\sigma}$ to $\mathcal{A}$.
3. At some point, the adversary sends $x_1, ..., x_n \in \mathcal{X}$ to the challenger.
4. The challenger computes the marked signing key $\mathsf{sigk}_i(x_i) \leftarrow \mathsf{Mark}(\mathsf{msk}_i, x_i)$ for $i \in [n]$ and sends $\{\mathsf{sigk}_i(x_i)\}_{i\in[n]}$ to $\mathcal{A}$.
5. After receiving the marked signing keys, $\mathcal{A}$ continues to make queries to the signing oracles. At some point, it declares that it is ready for the challenge.
6. The challenger picks uniformly random $m_i \leftarrow \mathcal{M}$ for $i \in [n]$ and sends $\{m_i\}_{i\in[n]}$ to $\mathcal{A}$.
7. Receiving $\{m_i\}_{i\in[n]}$ from the challenger, $\mathcal{A}$ outputs $\{\sigma_i\}_{i\in[n]}$. Note that after receiving $\{m_i\}_{i\in[n]}$ from the challenger, $\mathcal{A}$ loses access to the signing oracles.
8. The experiment outputs 1 (indicating that $\mathcal{A}$ wins) if $\mathsf{Vrfy}(\mathsf{vk}_i, m_i, \sigma_i) = \top$ for all $i \in [n]$ and otherwise outputs 0.

We define the advantage of a QPT adversary $\mathcal{A}$ to be

$$\mathsf{Adv}^{\mathsf{par\text{-}for}}_{\mathsf{WDS},\mathcal{A},n}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{par\text{-}for}}_{\mathsf{WDS},\mathcal{A},n}(1^\lambda) = 1\right].$$

We also define parallel mark extraction experiment defined by a QPT extractor $\mathcal{Ext}$, an adversary $\mathcal{A}$, and a challenger, parameterized by an arbitrary polynomial $n = \mathrm{poly}(\lambda)$, denoted as $\mathsf{Exp}^{\mathsf{par\text{-}ext}}_{\mathsf{WDS},\mathcal{A},\mathcal{Ext},n}(1^\lambda)$. The parallel extraction experiment is the same as parallel forgery experiment except for the last 3 steps. Namely, we replace Step 6, 7, and 8 with the following:

6' Take the adversary $\mathcal{A}$ right before Step 6 of $\mathsf{Exp}^{\mathsf{par\text{-}for}}_{\mathsf{WDS},\mathcal{A},n}(1^\lambda)$. Since $\mathcal{A}$ is a quantum non-uniform QPT adversary, we can parse $\mathcal{A}$ as a pair of a unitary circuit $U_\mathcal{A}$ and a quantum state $\rho_\mathcal{A}$, where to run $\mathcal{A}$ on the inputs $\{m_i\}_{i\in[n]}$, we apply $U_\mathcal{A}$ to $\{m_i\}_{i\in[n]}$ and $\rho_\mathcal{A}$ and then measure the corresponding registers to obtain the output $\{\sigma_i\}_{i\in[n]}$.

7' We run $\mathcal{Ext}(\{\mathsf{xk}_i\}_{i\in[n]}, (U_\mathcal{A}, \rho_\mathcal{A})) \to \{x'_i\}_{i\in[n]}$.

8' The experiment outputs 1 (indicating that the adversary wins) if $x'_i = x_i$ for all $i \in [n]$ and 0 otherwise.

We define the advantage of a QPT adversary $\mathcal{A}$ with respect to $\mathcal{Ext}$ to be

$$\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WDS},\mathcal{A},\mathcal{Ext},n}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{par\text{-}ext}}_{\mathsf{WDS},\mathcal{A},\mathcal{Ext},n}(1^\lambda) = 1\right].$$

We say that WDS satisfies parallel mark extractability if there exists a QPT algorithm $\mathcal{Ext}$ such that for all $n = \mathrm{poly}(\lambda)$, for all QPT algorithm $\mathcal{A}$, the following holds:

$$\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WDS},\mathcal{A},\mathcal{Ext},n}(\lambda) \geq \mathsf{Adv}^{\mathsf{par\text{-}for}}_{\mathsf{WDS},\mathcal{A},n}(\lambda) - \mathsf{negl}(\lambda).$$

Remark 3.8. Similarly to the case of UPF, the above definition only supports secret extraction. However, it is sufficient for our purpose.

Remark 3.9 (Security as a plain DS). We note that the above definition of a watermarkable DS does not necessarily imply the standard EUF-CMA security as a plain DS. Nevertheless, the EUF-CMA security can be added by a simple conversion: run a plain DS in parallel and append its signature alongside the original one. The new verification algorithm verifies both the signatures and accepts only when both are valid. The marked key of the resulting scheme consists of that of the original one along with the signing key of the plain DS.

Construction The above defined watermarkable digital signature scheme can be constructed from any watermarkable UPF and coherently-signable constrained signatures. Our construction abstracts the core idea of the watermarkable DS scheme implicit in [KMY25]. Observe first that UPF can already be viewed as a MAC; what remains is simply to augment it with a public verification algorithm. To achieve this, we equip the signer with a constrained signing key that allows signatures to be generated only on valid input–output pairs of the UPF. Consequently, in order to produce a valid signature on a message (treated as an input to the UPF), the signer must also provide a corresponding valid output of the UPF as part of the signature. This structure enables us to reuse the same extractor as in the UPF setting.

Formally, our construction of watermarkable digital signature scheme $\mathsf{WDS} = (\mathsf{KG}, \mathsf{Mark}, \mathsf{Sign}, \mathsf{Vrfy})$ uses watermarkable UPF $\mathsf{WUPF} = \mathsf{WUPF}.(\mathsf{KG}, \mathsf{Mark}, \mathsf{Eval})$ and coherently-signable constrained signatures $\mathsf{CS} = \mathsf{CS}.(\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Sign}, \mathsf{Vrfy})$ (See Definition 2.5 for the definition of coherently-signable constrained signatures.) as building blocks, both of which in turn can be constructed from SIS. As additional requirements, we need WUPF to satisfy equivocality and CS to satisfy function privacy. Our construction WDS supports mark space $\{0,1\}^w$ and the message space $\{0,1\}^\ell$. For the construction, we need the underlying WUPF to support the mark space $\{0,1\}^w$ and the input space $\{0,1\}^\ell$. We denote the bit-length of the output of the WUPF function by v. We then need CS to support the WUPF evaluation circuit defined below, whose input length is $\ell + v$ and the depth is bounded by a fixed polynomial.

$\mathsf{KG}(1^\lambda) \to (\mathsf{vk}, \mathsf{msk}, \mathsf{xk})$**:** On input the security parameter 1^λ, the key generation algorithm proceeds as follows.

- Run $\mathsf{WUPF.KG}(1^\lambda) \to (\mathsf{WUPF.msk}, \mathsf{WUPF.xk})$.
- Run $\mathsf{CS.Setup}(1^\lambda) \to (\mathsf{CS.vk}, \mathsf{CS.msk})$.
- Output $\mathsf{vk} = \mathsf{CS.vk}$, $\mathsf{msk} = (\mathsf{WUPF.msk}, \mathsf{CS.msk})$, and $\mathsf{xk} = \mathsf{WUPF.xk}$.

$\mathsf{Mark}(\mathsf{msk}, x) \to \mathsf{sigk}(x)$**:** Upon input the master secret key $\mathsf{msk} = (\mathsf{WUPF.msk}, \mathsf{CS.msk})$ and a mark $x \in \{0,1\}^w$, it proceeds as follows.

- Run $\mathsf{WUPF.Mark}(\mathsf{WUPF.msk}, x) = \mathsf{WUPF.key}(x)$.
- Construct a circuit $f[\mathsf{WUPF.key}(x)]$ that takes as input a pair $(m,t) \in \{0,1\}^\ell \times \{0,1\}^v$ and outputs 1 if $\mathsf{WUPF.Eval}(\mathsf{WUPF.key}(x), m) = t$ and 0 otherwise.
- Run $\mathsf{CS.Constrain}(\mathsf{CS.msk}, f[\mathsf{WUPF.key}(x)]) \to \mathsf{CS.sigk}(x)$.
- Output $\mathsf{sigk}(x) = (\mathsf{WUPF.key}(x), \mathsf{CS.sigk}(x))$.

$\mathsf{Sign}(\mathsf{sigk}(x), m) \to \sigma$**:** On input the marked key $\mathsf{sigk}(x)$ or the master secret key msk and the message $m \in \{0,1\}^\ell$, the signing algorithm proceeds as follows.

- If the first input is a master secret key, parse it as $\mathsf{msk} = (\mathsf{WUPF.msk}, \mathsf{CS.msk})$. Then, set $\mathsf{WUPF.key} = \mathsf{WUPF.msk}$ and compute $\mathsf{CS.Constrain}(\mathsf{CS.msk}, f_{=1}) \to \mathsf{CS.sigk}$, where $f_{=1}$ is a constant function that always outputs 1.
- If the first input is a marked key $\mathsf{sigk}(x)$, parse it as $\mathsf{sigk}(x) = (\mathsf{WUPF.key}, \mathsf{CS.sigk})$.
- Compute $t = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}, m)$ and $\mathsf{CS.Sign}(\mathsf{CS.sigk}, (m,t)) \to \mathsf{CS}.\sigma$.
- Output $\sigma = (t, \mathsf{CS}.\sigma)$.

$\mathsf{Vrfy}(\mathsf{vk}, m, \sigma)$**:** On input the message m and the signature $\sigma = (t, \mathsf{CS}.\sigma)$, it runs $\mathsf{CS.Vrfy}(\mathsf{CS.vk}, (m,t), \mathsf{CS}.\sigma)$ and outputs whatever it outputs.

Correctness We first observe that the signature $\sigma = (t, \mathsf{CS}.\sigma)$ generated by the master secret key passes the verification with overwhelming probability by the correctness of CS, since $f_{=1}(m,t) = 1$ for any m and t. We then observe that the signature $\sigma = (t, \mathsf{CS}.\sigma)$ generated by the signing key $\mathsf{sigk}(x)$ also passes the verification with overwhelming probability by the correctness of CS, since $f[\mathsf{WUPF.key}(x)](m,t) = 1$ holds when $t = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}(x), m)$.

Coherent Signability To show the coherent signability, we first define the quantum signing algorithm $\mathcal{QSign}$. It takes as input a quantum state $|\psi\rangle$ and a message m. We can assume that $|\psi\rangle$ is of the form $\sum_{z\in\{0,1\}^L} \alpha_z |z\rangle |\mathsf{sigk}(x_z)\rangle = \sum_{z\in\{0,1\}^L} \alpha_z |z\rangle |\mathsf{WUPF.key}(x_z)\rangle |\mathsf{CS.sigk}(x_z)\rangle$.

1. Apply the following isometry to $|\psi\rangle$:

$$|z\rangle |\mathsf{WUPF.key}(x_z)\rangle |\mathsf{CS.sigk}(x_z)\rangle \mapsto |z\rangle |\mathsf{WUPF.key}(x_z)\rangle |\mathsf{CS.sigk}(x_z)\rangle |\mathsf{WUPF.Eval}(\mathsf{WUPF.key}(x_z), m)\rangle .$$

2. Measure the last register and discard it. Let the result of the measurement be t and the resulting state after discarding the register be $|\phi\rangle$.

3. Run $\mathsf{CS}.\mathcal{QSign}(|\phi\rangle, (m,t)) \to (|\psi'\rangle, \mathsf{CS}.\sigma)$ and output $|\psi'\rangle$ and $\sigma = (t, \mathsf{CS}.\sigma)$.

We first show that the above defined algorithm satisfies the first property of coherent signability, which states that applying $\mathcal{QSign}$ to a classical message results in a signature distributed identically to when Sign is applied. This straightforwardly follows from the fact that $\mathsf{CS}.\mathcal{QSign}$ has the same output distribution as CS.Sign when the input is classical.

We then show the second property, which states that the application of $\mathcal{QSign}$ on a superposition of a marked signing key almost does not change the state. To see this, we first observe that after applying the first step, the state becomes

$$\begin{aligned}&\sum_{z\in\{0,1\}^L} \alpha_z |z\rangle |\mathsf{WUPF.key}(x_z)\rangle |\mathsf{CS.sigk}(x_z)\rangle |\mathsf{WUPF.Eval}(\mathsf{WUPF.key}(x_z), m)\rangle \\ &= \sum_{z\in\{0,1\}^L} \alpha_z |z\rangle |\mathsf{WUPF.key}(x_z)\rangle |\mathsf{CS.sigk}(x_z)\rangle |\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}, m)\rangle ,\end{aligned}$$

where the equation holds with overwhelming probability over the randomness of $\mathsf{WUPF.KG}(1^\lambda)$ from the correctness of WUPF.[16] Therefore, we have $|\phi\rangle = |\psi\rangle$ with overwhelming probability. We next see that the application of $\mathsf{CS}.\mathcal{QSign}$ almost does not change the state $|\phi\rangle$. To see this, we consider a function family $\{g_{z'}\}_{z'}$ where for $z' = (z, \mathsf{WUPF.key}(x_z))$, $g_{z'}$ is defined as $g_{z'} = f[\mathsf{WUPF.key}(x_z)]$ and apply the coherent signability of CS. We therefore have that $|\psi'\rangle$ is negligibly close to $|\psi\rangle$ in trace distance as desired.

Security Proof The following theorem asserts the parallel mark extractability of the above construction.

Theorem 3.10. *Assume that* WUPF *satisfies the parallel mark extractability as watermarkable unpredictable function and equivocality and* CS *satisfies selective single-key security and function privacy. Then, the above construction satisfies parallel mark extractability as watermarkable digital signatures.*

Before presenting the proof, we briefly outline the intuition. At a high level, one can extract the mark from a signer, since a signature $\sigma = (t, \mathsf{CS}.\sigma)$ contains the WUPF value t and the extractor for WUPF can be applied to perform the extraction. The only way for the adversary to evade the extraction is to produce a signature $(t, \mathsf{CS}.\sigma)$ such that $\mathsf{CS}.\sigma$ is a valid signature on (m,t), but $t \neq \mathsf{WUPF.Eval}(\mathsf{WUPF.key}(x), m)$. However, generating such a signature constitutes a forgery against CS and is therefore computationally infeasible, since the adversary only obtains a constrained key that permits signing on pairs (m,t) satisfying $t = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}(x), m)$.

Turning this intuition into a formal proof is nontrivial, because the mark is chosen adaptively by the adversary (whereas in [KMY25] it is sampled uniformly at random). In particular, when reducing to selectively single-key forgery, the reduction must commit to a function f at the beginning of the game. In our setting, however, this function necessarily depends on the mark x, since it hardwires the WUPF key for that mark, which is chosen adaptively by the adversary. As a result, the reduction cannot commit to the function at the outset. To overcome this difficulty, we rely on the equivocality of WUPF, which provides a simulated key that is indistinguishable from a marked key even before x is known. This simulated key can be chosen independently of x and then embedded into the function.

[16] Recall that we require somewhat strong evaluation correctness condition for WUPF, which requires that with overwhelming probability over the choice of WUPF.msk and WUPF.xk, the evaluation result by the master secret key equals that of the marked key with mark x *for all* x.

An additional subtlety arises from the presence of a signing oracle, which is not introduced in [KMY25]. For simulating the signing oracle, we use the constrained key of CS, similarly to the case of WUPF. Here, however, the function hardwired into the constrained key differs from the one in the real execution, and thus could be detected by the adversary. The function privacy of CS guarantees this is not the case and such a difference is hidden from the adversary.

Proof. To prove the theorem, we first define the extraction algorithm $\mathcal{Ext}$ as follows.

$\mathcal{Ext}(\{\mathsf{xk}_i\}_{i\in[n]}, (U_{\mathcal{A}}, \rho_{\mathcal{A}})) \to \{x'_i\}_{i\in[n]}$**:** Given the extraction keys $\{\mathsf{xk}_i\}_{i\in[n]}$ and the description of the adversary $\mathcal{A} = (U_{\mathcal{A}}, \rho_{\mathcal{A}})$, it proceeds as follows.

- Parse the extraction key as $\mathsf{xk}_i = \mathsf{WUPF.xk}_i$ for each $i \in [n]$.
- From the adversary $\mathcal{A}$, construct a predictor $\mathcal{P} = (U_{\mathcal{P}}, \rho_{\mathcal{P}})$ for WUPF as follows.

  $\mathcal{P}(m_1, \dots, m_n)$: Given $m_1, \dots, m_n$, it first runs $\mathcal{A}$ on input $m_1, \dots, m_n$ to obtain $\{\sigma_i\}_{i\in[n]}$. It then parses $\sigma_i = (t_i, \mathsf{CS}.\sigma_i)$ for each $i \in [n]$. It finally outputs $(t_1, \dots, t_n)$.
- Run $\mathsf{WUPF}.\mathcal{Ext}(\{\mathsf{WUPF.xk}_i\}_{i\in[n]}, (U_{\mathcal{P}}, \rho_{\mathcal{P}})) \to \{x'_i\}_{i\in[n]}$.
- Output $\{x'_i\}_{i\in[n]}$.

We then prove the theorem by a sequence of hybrids. Let us consider an adversary $\mathcal{A}$ against the parallel forgery game $\mathsf{Exp}^{\mathsf{par\text{-}for}}_{\mathsf{WDS},\mathcal{A},n}(1^\lambda)$. We gradually change the game into the parallel extraction game $\mathsf{Exp}^{\mathsf{par\text{-}ext}}_{\mathsf{WDS},\mathcal{A},\mathcal{Ext},n}(1^\lambda)$ without changing the winning probability of $\mathcal{A}$ more than negligibly. In the following, the event that $\mathcal{A}$ wins in $\mathsf{Hyb}_{\mathsf{xx}}$ is denoted by E_{xx}.

Hyb_0**:** This is the parallel forgery game between the challenger and an adversary $\mathcal{A}$. Namely, the game proceeds as follows.

1. The challenger runs $\mathsf{KG}(1^\lambda)$ to obtain $\mathsf{vk}_i = \mathsf{CS.vk}_i$, $\mathsf{msk}_i = (\mathsf{WUPF.msk}_i, \mathsf{CS.msk}_i)$, and $\mathsf{xk}_i = \mathsf{WUPF.xk}_i$ for $i \in [n]$. $\mathcal{A}$ is given $\{\mathsf{vk}_i\}_{i\in[n]}$.
2. Throughout the game, the adversary has an oracle access to the signing oracle $\mathsf{Sign}(\mathsf{msk}_i,)$ for each $i \in [n]$, until it receives the challenge messages in Step 7. When queried on input $\hat{m}$, the oracles $\mathsf{Sign}(\mathsf{msk}_i,)$ proceeds as follows:
   - Compute $\mathsf{CS.Constrain}(\mathsf{CS.msk}_i, f_{=1}) \to \mathsf{CS.sigk}_i$, where $f_{=1}$ is a constant function that always outputs 1.
   - Compute $\hat{t} = \mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, \hat{m})$.
   - Run $\mathsf{CS.Sign}(\mathsf{CS.sigk}_i, (\hat{m}, \hat{t})) \to \mathsf{CS}.\hat{\sigma}$.
   - Return $\hat{\sigma} = (\hat{t}, \mathsf{CS}.\hat{\sigma})$.
3. At some point, $\mathcal{A}$ chooses arbitrary $x_1, ..., x_n \in \mathcal{X}$ and sends them to the challenger.
4. For $i \in [n]$, the challenger sets the marked signing keys $\mathsf{sigk}_i := (\mathsf{WUPF.key}_i(x_i), \mathsf{CS.sigk}_i(x_i))$ and sends $\{\mathsf{sigk}_i\}_{i\in[n]}$ to the adversary, where

   $\mathsf{WUPF.key}_i(x_i) = \mathsf{WUPF.Mark}(\mathsf{WUPF.msk}_i, x_i)$ and $\mathsf{CS.sigk}_i(x_i) \leftarrow \mathsf{CS.Constrain}(\mathsf{CS.msk}_i, f[\mathsf{WUPF.key}_i(x_i)])$.
5. After receiving the marked signing keys, $\mathcal{A}$ continues to make queries to the signing oracles. At some point, it declares that it is ready for the challenge.
6. The challenger picks uniformly random $m_i \leftarrow \{0,1\}^\ell$ for $i \in [n]$ and sends $\{m_i\}_{i\in[n]}$ to the adversary.
7. Receiving $\{m_i\}_{i\in[n]}$ from the challenger, $\mathcal{A}$ outputs $\{\sigma_i\}_{i\in[n]}$. Note that after receiving $\{m_i\}_{i\in[n]}$ from the challenger, $\mathcal{A}$ loses access to the signing oracles.
8. Given $\{\sigma_i\}_{i\in[n]}$, the challenger parses it as $\sigma_i = (t_i, \mathsf{CS}.\sigma_i)$ for $i \in [n]$ and proceeds as follows.
   - It checks whether $\mathsf{CS.Vrfy}(\mathsf{CS.vk}_i, (m_i, t_i), \mathsf{CS}.\sigma_i) = \top$ holds for all $i \in [n]$. If it holds, then it sets the output of the game to be 1 (indicating that the adversary wins). Otherwise, it outputs 0.

Hyb_1: In this hybrid, we use the simulated key of WUPF to answer signing queries and to simulate the marked signing keys. Concretely, we replace Step 1, Step 2, and Step 4 with the following:

1' The challenger runs $\mathsf{CS.Setup}(1^\lambda) \to (\mathsf{CS.vk}_i, \mathsf{CS.msk}_i)$ and $\mathsf{WUPF.Sim}(1^\lambda) \to \mathsf{WUPF.key}_i^*$ for $i \in [n]$. Then, $\mathcal{A}$ is given $\{\mathsf{vk}_i = \mathsf{CS.vk}_i\}_{i\in[n]}$.

2' Throughout the game, the adversary has an oracle access to the signing oracles, until it receives the challenge messages in Step 7. When queried on input $\hat{m}$, the i-th oracle proceeds as follows:

- Compute $\mathsf{CS.Constrain}(\mathsf{CS.msk}_i, f_{=1}) \to \mathsf{CS.sigk}_i$, where $f_{=1}$ is a constant function that always outputs 1.
- Compute $\hat{t} = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i^*, \hat{m})$.
- Run $\mathsf{CS.Sign}(\mathsf{CS.sigk}_i, (\hat{m}, \hat{t})) \to \mathsf{CS.}\hat{\sigma}$.
- Return $\hat{\sigma} = (\hat{t}, \mathsf{CS.}\hat{\sigma})$.

4' For $i \in [n]$, the challenger sets the marked signing keys $\mathsf{sigk}_i := (\mathsf{WUPF.key}_i^*, \mathsf{CS.sigk}_i^*)$ and sends $\{\mathsf{sigk}_i\}_{i\in[n]}$ to the adversary, where

$$\mathsf{CS.sigk}_i^* \leftarrow \mathsf{CS.Constrain}(\mathsf{CS.msk}_i, f[\mathsf{WUPF.key}_i^*]). \tag{4}$$

By a straightforward reduction to the equivocality of WUPF, we have $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\lambda)$.

Hyb_2: In this hybrid, we change how we handle signing queries. We move the sampling of $\mathsf{CS.sigk}_i^*$ as per Equation (4) to Step 1, which is possible since it is not dependent on x_i any more, and then use it for answering the signing queries. Namely, we replace Step 1', Step 2', and Step 4' with the following:

1" The challenger runs $\mathsf{CS.Setup}(1^\lambda) \to (\mathsf{CS.vk}_i, \mathsf{CS.msk}_i)$, $\mathsf{WUPF.Sim}(1^\lambda) \to \mathsf{WUPF.key}_i^*$, and $\mathsf{CS.sigk}_i^* \leftarrow \mathsf{CS.Constrain}(\mathsf{CS.msk}_i, f[\mathsf{WUPF.key}_i^*])$ for $i \in [n]$. Then, $\mathcal{A}$ is given $\{\mathsf{vk}_i = \mathsf{CS.vk}_i\}_{i\in[n]}$.

2" Throughout the game, the adversary has an oracle access to the signing oracles, until it receives the challenge messages in Step 7. When queried on input $\hat{m}$, the i-th oracle proceeds as follows:

- Compute $\hat{t} = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i^*, \hat{m})$.
- Run $\mathsf{CS.Sign}(\mathsf{CS.sigk}_i^*, (\hat{m}, \hat{t})) \to \mathsf{CS.}\hat{\sigma}$.
- Return $\hat{\sigma} = (\hat{t}, \mathsf{CS.}\hat{\sigma})$.

4" For $i \in [n]$, the challenger sets the marked signing keys $\mathsf{sigk}_i := (\mathsf{WUPF.key}_i^*, \mathsf{CS.sigk}_i^*)$ and sends $\{\mathsf{sigk}_i\}_{i\in[n]}$ to the adversary.

We claim that this hybrid is statistically indistinguishable from the previous one, due to the function privacy of CS. We recall that $\mathsf{CS.sigk}_i^*$ in this hybrid encodes a function $f[\mathsf{WUPF.key}_i^*]$ and thus we have $f[\mathsf{WUPF.key}_i^*](\hat{m}, \hat{t}) = 1 = f_{=1}(\hat{m}, \hat{t})$. Therefore, the distribution of the returned $\mathsf{CS.}\hat{\sigma}$ is statistically close to that of the previous hybrid. We therefore have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda)$.

Hyb_3: In this hybrid, we change the winning condition of $\mathcal{A}$. Namely, we replace Step 8 with the following, where additional check regarding t_i is inserted.

8' Given $\{\sigma_i\}_{i\in[n]}$, the challenger parses it as $\sigma_i = (t_i, \mathsf{CS.}\sigma_i)$ for $i \in [n]$ and proceeds as follows.

- It checks whether $t_i = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i^*, m_i)$ holds for all $i \in [n]$. If not, it outputs 0.
- It then checks whether $\mathsf{CS.Vrfy}(\mathsf{CS.vk}_i, (m_i, t_i), \mathsf{CS.}\sigma_i) = \top$ holds for all $i \in [n]$. If it holds, then it sets the output of the game to be 1 (indicating that the adversary wins). Otherwise, it outputs 0.

We claim that this change does not reduce the winning probability of $\mathcal{A}$ more than negligibly due to the selective single key unforgeability of CS. To see this, we observe that the only case where $\mathcal{A}$ wins the game in the previous hybrid, but does not in the current one occurs when $\mathsf{CS.Vrfy}(\mathsf{CS.vk}_i, (m_i, t_i), \mathsf{CS.}\sigma_i) = \top$ holds for all $i \in [n]$, but there exists $i^* \in [n]$ such that $t_{i^*} \neq \mathsf{WUPF.Eval}(\mathsf{WUPF.key}_{i^*}^*, m_{i^*})$. However, the probability that such an event occurs is negligible due to the selective single-key security of CS. To see this, we consider

a reduction that guesses i^* and embeds the parameter of the CS into this instance, while the other instances as well as the parameters related to WUPF are generated by itself. We observe that the reduction algorithm can choose its target function $f[\mathsf{WUPF.key}^*_{i^*}]$ at the beginning of the game, since $\mathsf{WUPF.key}^*_{i^*}$ can be chosen by $\mathsf{WUPF.Sim}(1^\lambda)$, without depending on any parameter other than the security parameter. Furthermore, by the changes we introduced so far, all the signing queries made by $\mathcal{A}$ can be handled by the single key $\mathsf{CS.sigk}^*_{i^*}$. Finally, since $t_{i^*} \neq \mathsf{WUPF.Eval}(\mathsf{WUPF.key}^*_{i^*}, m_{i^*})$ implies $f[\mathsf{WUPF.key}^*_{i^*}](m_{i^*}, t_{i^*}) = 0$, the reduction breaks the selective single-key security if such an event occurs. We therefore have $\Pr[\mathsf{E}_3] \geq \Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda)$.

Hyb_4: In this hybrid, we switch Step 1", Step 2", and Step 4" back to those in Hyb_1, i.e., Step 1', Step 2', and Step 4'. In particular, we use $\mathsf{CS.sigk}_i$ derived as $\mathsf{CS.Constrain}(\mathsf{CS.msk}_i, f_{=1}) \to \mathsf{CS.sigk}_i$ to sign on $(\hat{m}, \hat{t})$. Similarly to the change from Hyb_1 to Hyb_2, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| = \mathsf{negl}(\lambda)$ by the function privacy of CS.

Hyb_5: In this hybrid, we switch Step 1', Step 2', and Step 4' back to those in Hyb_0, i.e., Step 1, Step 2, and Step 4. Namely, the simulated key $\mathsf{WUPF.key}^*_i$ is not generated any more. Accordingly, we also change Step 8' so that it uses $\mathsf{WUPF.key}_i(x_i)$ instead of $\mathsf{WUPF.key}^*_i$. Concretely, we replace Step 8' with the following.

8" Given $\{\sigma_i\}_{i\in[n]}$, the challenger parses it as $\sigma_i = (t_i, \mathsf{CS}.\sigma_i)$ for $i \in [n]$ and proceeds as follows.

- It checks whether $t_i = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i(x_i), m_i)$ holds for all $i \in [n]$. If not, it outputs 0.
- It then checks whether $\mathsf{CS.Vrfy}(\mathsf{CS.vk}_i, (m_i, t_i), \mathsf{CS}.\sigma_i) = \top$ holds for all $i \in [n]$. If it holds, then it sets the output of the game to be 1 (indicating that the adversary wins). Otherwise, it outputs 0.

Similarly to the change from Hyb_0 to Hyb_1, we have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| = \mathsf{negl}(\lambda)$ by the equivocality of WUPF.

Hyb_6: In this hybrid, we further change the winning condition so that it does not check the validity of CS signatures. Namely, we replace Step 8" with the following:

8"' Given $\{\sigma_i\}_{i\in[n]}$, the challenger parses it as $\sigma_i = (t_i, \mathsf{CS}.\sigma_i)$ for $i \in [n]$ and proceeds as follows.

- It checks whether $t_i = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i(x_i), m_i)$ holds for all $i \in [n]$. If not, it outputs 0.

This change relaxes the winning condition for $\mathcal{A}$ and only increases the chance of it winning. We therefore have $\Pr[\mathsf{E}_6] \geq \Pr[\mathsf{E}_5]$.

Hyb_7: In this hybrid, we further change the winning condition so that the check of the WUPF values are done using $\mathsf{WUPF.msk}$. Namely, we replace Step 8"' with the following:

8* Given $\{\sigma_i\}_{i\in[n]}$, the challenger parses it as $\sigma_i = (t_i, \mathsf{CS}.\sigma_i)$ for $i \in [n]$ and proceeds as follows.

- It checks whether $t_i = \mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, m_i)$ holds for all $i \in [n]$. If not, it outputs 0.

By the correctness against adversarially chosen mark and input of WUPF, which is implied by the equivocality, we have $\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, m_i) = \mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i(x_i), m_i)$ with overwhelming probability for each $i \in [n]$, even if m_i was adversarially chosen by $\mathcal{A}$. We therefore have $|\Pr[\mathsf{E}_7] - \Pr[\mathsf{E}_6]| \leq \mathsf{negl}(\lambda)$.

Hyb_8: In this hybrid, we further change the winning condition. Here, we use $\mathcal{A}$ as a predictor for WUPF and then extract the marks using the corresponding extractor of WUPF. Concretely, we replace Step 6, Step 7, and Step 8* with the following:

6** The challenger parses the adversary $\mathcal{A}$ right before Step 7 of the game as a unitary circuit $U_{\mathcal{A}}$ and a quantum state $\rho_{\mathcal{A}}$. It then constructs a quantum predictor $\mathcal{P}$ for WUPF that takes $\{m_i\}_{i\in[n]}$ as input and outputs $\{t_i\}_{i\in[n]}$ from $\mathcal{A}$ as follows.

$\mathcal{P}(\{m_i\}_{i\in[n]})$**:** It runs $\mathcal{A}$ on input $\{m_i\}_{i\in[n]}$ to obtain $\{\sigma_i\}_{i\in[n]}$. It then parses $\sigma_i = (t_i, \mathsf{CS}.\sigma_i)$ for $i \in [n]$ and outputs $\{t_i\}_{i\in[n]}$.

7** The challenger parses $\mathcal{P}$ as a pair of unitary $U_{\mathcal{P}}$ and a quantum state $\rho_{\mathcal{P}}$. Then, it runs the extraction algorithm as $\mathsf{WUPF}.\mathcal{E}\mathcal{X}\mathsf{t}(\{\mathsf{WUPF.xk}_i\}_{i\in[n]}, (U_{\mathcal{P}}, \rho_{\mathcal{P}})) \to \{x'_i\}_{i\in[n]}$.

8^{**} If $x_i = x'_i$ holds for all $i \in [n]$, the challenger sets the output of the game to be 1. Otherwise, it sets it to be 0.

Let us construct an adversary $\mathcal{B}$ against the parallel prediction experiment of WUPF by running $\mathcal{A}$ internally. $\mathcal{B}$ samples the parameters related to CS by itself and simulates the computation of $\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, \cdot)$ necessary for answering the signing queries from $\mathcal{A}$ by accessing its evaluation oracles. $\mathcal{B}$ then predicts the PRF values by constructing $\mathcal{P}$ from $\mathcal{A}$ as above and running it on the challenge input. It is straightforward to see that the advantage of $\mathcal{B}$ against parallel predictability game of WUPF is the same as $\Pr[\mathsf{E}_7]$. It is also easy to see that the advantage of $\mathcal{B}$ against the parallel mark extraction experiment equals $\Pr[\mathsf{E}_8]$. Therefore, by the parallel mark extractability of WUPF, we have $\Pr[\mathsf{E}_8] \geq \Pr[\mathsf{E}_7] - \mathsf{negl}(\lambda)$.

From the above discussion, we can see that $\Pr[\mathsf{E}_8] \geq \Pr[\mathsf{E}_0] - \mathsf{negl}(\lambda)$ holds. Furthermore, one can see by inspection that Hyb_8 corresponds to the parallel extraction experiment $\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WDS},\mathcal{A},\mathcal{E}\mathcal{X}\mathcal{t},n}(\lambda)$ for $\mathcal{A}$ with respect to $\mathcal{E}\mathcal{X}\mathcal{t}$ we defined. The theorem therefore follows. □

4 Special Dual-Mode Secure Function Evaluation

In this section, we introduce special dual-mode secure function evaluation (SFE), which is used as a building block of our secure key leasing schemes with classical lessors. We show that special dual-mode SFE exists assuming the LWE assumption.

Below, we define special dual-mode secure SFE.

Definition 4.1 (Special Dual-mode SFE). *A special dual-mode SFE scheme* SFE *is a tuple of four algorithms* $\mathsf{SFE} = \mathsf{SFE}.(\mathsf{CRSGen}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$. *Below, let* $\{0,1\}^w$ *be the message space of* SFE.

$\mathsf{CRSGen}(1^\lambda, \mathsf{mode}) \to (\mathsf{crs}, \mathsf{td})$**:** *The CRS generation algorithm is a PPT algorithm that takes a security parameter* 1^λ *and* $\mathsf{mode} \in \{1,2\}$*, and outputs a CRS* crs *and a trapdoor* td. $\mathsf{mode} = 1$ *indicates that the system is in the "extractable mode", while* $\mathsf{mode} = 2$ *indicates "hiding mode".*

$\mathsf{Rec}_1(\mathsf{crs}, x) \to (\mathsf{msg}^{(1)}, \mathsf{st})$**:** *This is a classical PPT algorithm supposed to be run by a receiver to generate the first message of the protocol. It takes as input the CRS* crs *and an input* $x \in \{0,1\}^w$ *and outputs the first message* $\mathsf{msg}^{(1)}$ *and a secret state* st.

$\mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, C) \to \mathsf{msg}^{(2)}$**:** *This is a classical PPT algorithm supposed to be run by a sender to generate the second message of the protocol. It takes as input the CRS* crs*, a message* $\mathsf{msg}^{(1)}$ *from the receiver, and a circuit* C *with input length* w *and outputs the second message* $\mathsf{msg}^{(2)}$.

$\mathsf{Rec}_2(\mathsf{crs}, x, \mathsf{st}, \mathsf{msg}^{(2)}) \to y$**:** *This is a classical PPT algorithm supposed to be run by a receiver to derive the output. It takes as input the CRS* crs*, an input* $x \in \{0,1\}^w$*, a state* st*, and a message* $\mathsf{msg}^{(2)}$ *from the sender, and outputs a string* y.

We require special dual-mode SFE to satisfy the following properties.

Evaluation correctness: *For all* $x \in \{0,1\}^w$, $\mathsf{mode} \in \{1,2\}$*, and a circuit* C *with input length* w*, we have*

$$\Pr\left[\mathsf{Rec}_2(\mathsf{crs}, x, \mathsf{st}, \mathsf{msg}^{(2)}) = C(x) \;\middle|\; \begin{array}{l}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda, \mathsf{mode}) \\ (\mathsf{msg}^{(1)}, \mathsf{st}) \leftarrow \mathsf{Rec}_1(\mathsf{crs}, x) \\ \mathsf{msg}^{(2)} \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, C)\end{array}\right] = 1.$$

Unique state: *For all* $\mathsf{mode} \in \{1,2\}$, $(\mathsf{crs}, \mathsf{td}) \in \mathsf{Supp}(\mathsf{CRSGen}(1^\lambda, \mathsf{mode}))$, $x \in \{0,1\}^w$*, and* $(\mathsf{msg}^{(1)}, \mathsf{st}) \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}, x))$, st *is the unique state that corresponds to* crs*,* x*, and* $\mathsf{msg}^{(1)}$*, that is, there is no* $\mathsf{st}' \neq \mathsf{st}$ *such that* $(\mathsf{msg}^{(1)}, \mathsf{st}') \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}, x))$*. We denote the unique state* st *such that* $(\mathsf{msg}^{(1)}, \mathsf{st}) \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}, x))$ *by* $\mathsf{st}_{x \to \mathsf{msg}^{(1)}}$.[17]

[17] The state also depends on crs, but we omit it from the notation.

Mode indistinguishability: *For any QPT algorithm $\mathcal{A}$, we have*

$$\left|\Pr\left[\mathcal{A}(\mathsf{crs})=1:(\mathsf{crs},\mathsf{td})\leftarrow\mathsf{CRSGen}(1^\lambda,1)\right]-\Pr\left[\mathcal{A}(\mathsf{crs})=1:(\mathsf{crs},\mathsf{td})\leftarrow\mathsf{CRSGen}(1^\lambda,2)\right]\right|\leq\mathsf{negl}(\lambda).$$

Statistical security against malicious senders in the hiding mode: *For all* crs *output by* $\mathsf{CRSGen}(1^\lambda,2)$ *and for all* $x_0,x_1\in\{0,1\}^w$*, we require the following statistical indistinguishability:*

$$\mathsf{msg}_0^{(1)}\approx_s\mathsf{msg}_1^{(1)}\quad\textit{where}\quad(\mathsf{msg}_b^{(1)},\mathsf{st})\leftarrow\mathsf{Rec}_1(\mathsf{crs},x_b)\quad\textit{for}\quad b\in\{0,1\}.$$

State recoverability in the hiding mode: *We require that there exists a classical PPT algorithm* StaRcv *that takes as input the trapdoor* $\mathsf{SFE.td}$ *and the first message of the receiver* $\mathsf{msg}^{(1)}$ *and outputs a tuple of strings* $\{\alpha_j^b\}_{j\in[w],b\in\{0,1\}}$ *or* $\bot$*, where the latter indicates that the state recovery failed. We require that for all* $x=x[1]\|\cdots\|x[w]\in\{0,1\}^w$*, the following holds:*

$$\Pr\left[\mathsf{st}=\alpha_1^{x[1]}\|\alpha_1^{x[2]}\|\cdots\|\alpha_w^{x[w]}\;\middle|\;\begin{array}{l}(\mathsf{crs},\mathsf{td})\leftarrow\mathsf{CRSGen}(1^\lambda,2)\\(\mathsf{msg}^{(1)},\mathsf{st})\leftarrow\mathsf{Rec}_1(\mathsf{crs},x)\\\{\alpha_j^b\}_{j\in[w],b\in\{0,1\}}\leftarrow\mathsf{StaRcv}(\mathsf{td},\mathsf{msg}^{(1)})\end{array}\right]=1.$$

That is, we can recover the corresponding st *from* $\mathsf{msg}^{(1)}$ *and* x *using* td*, and moreover, the state is in a specific form, where each block only depends on the bit of* x *at the corresponding position.*

Extractability in the extraction mode: *For classical algorithms* Ext *and* Sim*, let us consider the following experiment formalized by the experiment* $\mathsf{Exp}^{\mathsf{ext\text{-}sim}}_{\mathsf{SFE},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(1^\lambda,\mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and the challenger:*

1. *The challenger runs* $(\mathsf{crs},\mathsf{td})\leftarrow\mathsf{CRSGen}(1^\lambda,1)$ *and sends* $(\mathsf{crs},\mathsf{td})$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends* $\mathsf{msg}^{(1)}$ *and a circuit* C *with input length* w *to the challenger.*
3. *The challenger runs* $\mathsf{Ext}(\mathsf{td},\mathsf{msg}^{(1)})\to x$ *and it computes*

$$\mathsf{msg}^{(2)}\leftarrow\begin{cases}\mathsf{Send}(\mathsf{crs},\mathsf{msg}^{(1)},C), & \textit{if}\ \mathsf{coin}=0\\ \mathsf{Sim}(\mathsf{crs},\mathsf{msg}^{(1)},1^{|C|},C(x)) & \textit{if}\ \mathsf{coin}=1,\end{cases}$$

   where $|C|$ *denotes the size of* C*.*
   Then, it gives $\mathsf{msg}^{(2)}$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* coin' *for* coin*. The challenger outputs* coin' *as the final output of the experiment.*

We require that there exist PPT algorithms Ext *and* Sim *such that for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ext\text{-}sim}}_{\mathsf{SFE},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(\lambda):=\left|\Pr\left[\mathsf{Exp}^{\mathsf{ext\text{-}sim}}_{\mathsf{SFE},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(1^\lambda,0)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{ext\text{-}sim}}_{\mathsf{SFE},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(1^\lambda,1)=1\right]\right|\leq\mathsf{negl}(\lambda).$$

Efficient state superposition: *We require that there exists a QPT "coherent version"* $\mathcal{R}ec_1$ *of* Rec_1 *that is given a quantum state* $|\psi\rangle=\sum_{x\in\{0,1\}^w}\alpha_x|x\rangle$ *and* crs *and works as follows:*

It first generates a state that is within negligible trace distance of the following state:

$$\sum_{x\in\{0,1\}^w}\alpha_x\sum_{\mathsf{msg}^{(1)}\in\mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs},x))}\sqrt{p_{x\to\mathsf{msg}^{(1)}}}\,|x\rangle\left|\mathsf{st}_{x\to\mathsf{msg}^{(1)}}\right\rangle\left|\mathsf{msg}^{(1)}\right\rangle$$

where $p_{x\to\mathsf{msg}^{(1)}}$ *is the probability that* $\mathsf{Rec}_1(\mathsf{crs},x)$ *outputs* $\mathsf{msg}^{(1)}$*. Then, it measures the last register to obtain* $\mathsf{msg}^{(1)}$ *and outputs the residual quantum state* $|\psi'\rangle$*.*

We remark that implementing the above by simply running the classical algorithm in the superposition does not work, since it may leave the entanglement with the randomness for Rec_1*.*

We observe that a combination of the dual-mode oblivious transfer of [PVW08] and garbled circuits gives special dual-mode SFE satisfying the above definition.

Theorem 4.2. *If the LWE assumption holds, then there exists a special dual-mode SFE scheme.*

See Appendix A for the proof of Theorem 4.2.

5 Definitions of Secure Key Leasing with Classical Lessor

Here, we define secure key-leasing cryptographic primitives with a classical lessor. Specifically, we consider the cases of PKE, PRF, and digital signatures. Our definitions closely follow those of [KMY25], but with two key differences. First, in our setting, quantum keys are generated through classical communication between the lessor and the lessee, rather than being generated directly by a quantum lessor. Second, we strengthen the security notions: in our definitions, the adversary is additionally granted access to an evaluation oracle (for PRFs) or a signing oracle (for signatures), in contrast to [KMY25].

5.1 Public Key Encryption with Secure Key Leasing

In this subsection, we define public key encryption with secure key leasing (PKE-SKL), where the lessor is entirely classical and thus all the communication is classical as well. The lessee operates quantum computations only internally.

Definition 5.1 (PKE-SKL with classical lessor). *A PKE-SKL scheme* PKESKL *with classical lessor is a tuple of seven algorithms* $(\mathsf{Setup}, \mathit{IntKeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathcal{Dec}, \mathcal{Del}, \mathsf{DelVrfy})$*. Below, let* $\mathcal{M}$ *be the message space of* PKESKL*.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{ek}, \mathsf{msk}, \mathsf{dvk})$**:** *The setup algorithm is a classical PPT algorithm that takes as input the security parameter* 1^λ *and outputs an encryption key* ek*, a master secret key* msk*, and a deletion verification key* dvk*.*

$\mathit{IntKeyGen}(\mathsf{ek}, \mathsf{msk}) \to (\tau, \mathit{dk})$ **or** $(\tau, \bot)$**:** *The interactive key generation algorithm is an interactive protocol with classical communication run between a classical PPT lessor on input* msk *and QPT lessee on input* ek*. This either outputs a decryption key* dk *or* $\bot$*, which indicates that the protocol failed because the lessee cheated. After the protocol, the lessee obtains* dk*. We write* τ *to denote the transcript of the execution of* $\mathit{IntKeyGen}$*.*

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

$\mathsf{Dec}(\mathsf{msk}, \mathsf{ct}) \to \tilde{m}$**:** *The lessor's decryption algorithm is a PPT algorithm that takes a master secret key* msk *and a ciphertext* ct*, and outputs a value* $\tilde{m}$*.*

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

$\mathcal{Del}(\mathit{dk}) \to \mathsf{cert}$**:** *The deletion algorithm is a QPT algorithm that takes a decryption key* dk*, and outputs a classical string* cert*.*

$\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) \to \top/\bot$**:** *The deletion verification algorithm is a deterministic classical polynomial-time algorithm that takes a transcript* τ *for the interactive key generation, a deletion verification key* dvk*, and a deletion certificate* cert*, and outputs* $\top$ *or* $\bot$*.*

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

$$\Pr\left[\begin{array}{c|l} \mathcal{Dec}(\mathit{dk}, \mathsf{ct}) = m & (\mathsf{ek}, \mathsf{msk}, \mathsf{dvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \wedge & (\tau, \mathit{dk}) \leftarrow \mathit{IntKeyGen}(\mathsf{ek}, \mathsf{msk}) \\ \mathsf{Dec}(\mathsf{msk}, \mathsf{ct}) = m & \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, m) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Deletion verification correctness: *We have*

$$\Pr\left[\mathsf{DelVrfy}(\tau,\mathsf{dvk},\mathsf{cert})=\top \;\middle|\; \begin{array}{l}(\mathsf{ek},\mathsf{msk},\mathsf{dvk})\leftarrow\mathsf{Setup}(1^\lambda)\\(\tau,\mathit{dk})\leftarrow \mathit{IntKeyGen}(\mathsf{ek},\mathsf{msk})\\ \mathsf{cert}\leftarrow \mathcal{D}\mathit{el}(\mathit{dk})\end{array}\right]=1-\mathsf{negl}(\lambda).$$

Remark 5.2 (Syntax). One could consider the syntax where Setup and $\mathit{IntKeyGen}$ are merged. In such a setting, Setup is run by the lessor and ek is then sent to the lessee as the first message. We chose the above syntax as the main one because it allows the encryption key to be generated even without the interaction with the lessee, which seems to be more aligned with the concept of the secure key leasing.

Remark 5.3 (On the number of rounds in key generation). For a PKE-SKL as per the above syntax, at least two rounds of communication are necessary for $\mathit{IntKeyGen}$. Otherwise, it becomes possible for an adversary to copy the quantum decryption key dk and compromise the security (as defined below). Our construction consists of two rounds and is thus round-optimal in this sense.

Remark 5.4. We can assume without loss of generality that a decryption key of a PKE-SKL scheme is reusable. This means that it can be reused to decrypt arbitrarily many polynomial number of ciphertexts. This is because the decryption result of ct using dk is almost deterministic by the decryption correctness and thus measuring the decryption result does not disturb the input state by the gentle measurement lemma [Win99].

We next introduce the security notions for PKE-SKL with classical lessor. Similarly to [KMY25], we allow the adversary to obtain the verification key after submitting a valid certificate that passes the verification, which is stronger than many existing security definitions of PKE-SKL [AKN+23] (or key-revocable PKE [APV23]).

Definition 5.5 (IND-VRA security). *We say that a PKE-SKL scheme* PKESKL *with classical lessor for the message space $\mathcal{M}$ is IND-VRA secure,*[18] *if it satisfies the following requirement, formalized by the experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}vra}}_{\mathsf{PKESKL},\mathcal{A}}(1^\lambda,\mathsf{coin})$ *between an adversary $\mathcal{A}$ and the challenger:*

1. *The challenger runs* $\mathsf{Setup}(1^\lambda)\to(\mathsf{ek},\mathsf{msk},\mathsf{dvk})$ *and sends* ek *to $\mathcal{A}$.*
2. *Then, the challenger plays the role of the lessor on input* msk *and runs $\mathit{IntKeyGen}$ with the adversary $\mathcal{A}$. If the protocol outputs $\bot$, the challenger outputs* 0 *as the output of the game. Let τ be the transcript of the execution of $\mathit{IntKeyGen}$.*
3. *$\mathcal{A}$ sends* cert *and $(m_0^*,m_1^*)\in\mathcal{M}^2$ to the challenger. If* $\mathsf{DelVrfy}(\tau,\mathsf{dvk},\mathsf{cert})=\bot$*, the challenger outputs* 0 *as the final output of this experiment. Otherwise, the challenger generates* $\mathsf{ct}^*\leftarrow\mathsf{Enc}(\mathsf{ek},m^*_{\mathsf{coin}})$*, and sends* dvk *and* ct^* *to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs a guess* coin' *for* coin*. The challenger outputs* coin' *as the final output of the experiment.*

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

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

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

Definition 5.6 (OW-VRA security). *We say that a PKE-SKL scheme* PKESKL *with classical lessor for the message space $\mathcal{M}$ is OW-VRA secure, if it satisfies the following requirement, formalized by the experiment* $\mathsf{Exp}^{\mathsf{ow\text{-}vra}}_{\mathsf{PKESKL},\mathcal{A}}(1^\lambda)$ *between an adversary $\mathcal{A}$ and the challenger:*

1. *The challenger runs* $\mathsf{Setup}(1^\lambda)\to(\mathsf{ek},\mathsf{msk},\mathsf{dvk})$ *and sends* ek *to $\mathcal{A}$.*
2. *Then, the challenger plays the role of the lessor on input* msk *and runs $\mathit{IntKeyGen}$ with the adversary $\mathcal{A}$. If the protocol outputs $\bot$, the challenger outputs* 0 *as the output of the game. Let τ be the transcript of the execution of $\mathit{IntKeyGen}$.*

[18] "VRA" stands for "**V**erification key **R**evealing **A**ttack"

3. *$\mathcal{A}$ sends* cert *to the challenger. If* $\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \bot$*, the challenger outputs* 0 *as the final output of this experiment. Otherwise, the challenger chooses* $m^* \leftarrow \mathcal{M}$*, generates* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m^*)$*, and sends* dvk *and* ct^* *to* $\mathcal{A}$*.*

4. *$\mathcal{A}$ outputs m'. The challenger outputs* 1 *if* $m' = m^*$ *and otherwise outputs* 0 *as the final output of the experiment.*

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

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

Remark 5.7 (Security as plain PKE). It is straightforward to see that IND-VRA (resp. OW-VRA) security implies IND-CPA (resp. OW-CPA) security as a plain PKE scheme with quantum decryption keys.

By the quantum Goldreich-Levin lemma [AC02, CLLZ21] we have the following lemma. The proof is almost identical to that of [AKN+23, Lemma 3.12], and thus we omit it.

Lemma 5.8. *If there exists a OW-VRA secure PKE-SKL scheme with classical lessor, then there exists an IND-VRA secure PKE-SKL scheme with classical lessor.*

As we discuss in Remark 5.2, we can consider an alternative syntax where Setup is merged into $\mathit{IntKeyGen}$ and ek, msk, dvk, and dk are jointly generated through the interaction between the lessee and lessor. If we consider such an alternative syntax, our main construction in Section 6.1 requires 3 rounds of interaction, which is not optimal. However, as we show in Section 6.3, we can achieve optimal 2 rounds of interaction even in this setting as well by adding a small modification to our main construction. To capture the scheme, we define the modified version of the syntax for PKE-SKL with classical lessor and *non-interactive key generation*.

Definition 5.9 (PKE-SKL with classical lessor and non-interactive key generation). *We define PKE-SKL with classical lessor with non-interactive key generation by replacing* Setup *and* $\mathit{IntKeyGen}$ *in PKE-SKL as per Definition 5.1 with the following 2 algorithms:*

$\mathsf{Setup}_{\mathsf{NI}}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk}, \mathsf{dvk})$**:** *The lessor runs the setup algorithm on input* 1^λ *and outputs the public parameter* pp*, master secret key* msk*, and the deletion verification key* dvk*.*

$\mathcal{KG}_{\mathsf{NI}}(\mathsf{pp}) \to (\mathsf{ek}, \mathit{dk})$**:** *The key generation algorithm is a QPT algorithm that takes as input a public parameter* pp *and outputs an encryption key* ek *and a decryption key* dk*. We assume that* pp *is included in* ek*.*

In addition, $\mathsf{DelVrfy}$ *takes* ek *as input instead of the transcript* τ *of the interactive key generation.*

One can define decryption correctness and deletion verification correctness similarly to Definition 5.1 by replacing Setup *and* $\mathit{IntKeyGen}$ *with the above two algorithms. The IND-VRA and OW-VRA security are defined similarly to Definition 5.5 and 5.6, where the adversary receives* pp *from the challenger at the outset of the game and then sends* ek *to the challenger. The rest of the games are the same.*

In the above modified syntax, one can consider $\mathsf{Setup}_{\mathsf{NI}}$ combined with $\mathcal{KG}_{\mathsf{NI}}(\mathsf{pp})$ as an interactive key generation protocol with 2 rounds of communication. We say the key generation of the scheme is non-interactive, since the lessor does not have to do anything after running $\mathsf{Setup}_{\mathsf{NI}}(1^\lambda)$ and publicizing pp. The drawback of PKE-SKL as per Definition 5.9 is that it requires interaction between the lessor and lessee to generate the encryption key ek, unlike Definition 5.1.

An analogue of Lemma 5.8 in the non-interactive key generation setting holds. We therefore can focus on the construction of OW-VRA secure scheme. We omit the proof for the same reason as Lemma 5.8.

Lemma 5.10. *If there exists a OW-VRA secure PKE-SKL scheme with classical lessor and non-interactive key generation, then there exists an IND-VRA secure PKE-SKL scheme with classical lessor and non-interactive key generation.*

5.2 Pseudorandom and Unpredictable Functions with Secure Key Leasing

In this subsection, we define pseudorandom functions with secure key leasing (PRF-SKL) with classical lessor. Similarly to the case of PKE-SKL, everything is classical except that the internal computation by the lessee is quantum.

Definition 5.11 (PRF-SKL with classical lessor). *A PRF-SKL scheme* PRFSKL *with classical lessor for the domain* D_{prf} *and the range* R_{prf} *is a tuple of six algorithms* $(\mathsf{Setup}, \mathit{IntKeyGen}, \mathsf{Eval}, \mathcal{LE}\mathit{val}, \mathcal{D}\mathit{el}, \mathsf{DelVrfy})$.

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

$\mathit{IntKeyGen}(\mathsf{pp}, \mathsf{msk}) \to (\tau, sk)$ **or** $(\tau, \bot)$**:** *The interactive key generation algorithm is an interactive protocol with classical communication run between a classical PPT lessor who takes as input* msk *and QPT lessee on input* pp*. This either outputs a secret key* sk *or* $\bot$*, which indicates that the protocol failed because the lessee cheats. After the protocol, the lessee obtains* sk*. We write* τ *to denote the transcript of the execution of* $\mathit{IntKeyGen}$.

$\mathsf{Eval}(\mathsf{msk}, s) \to t$**:** *The evaluation algorithm is a deterministic classical polynomial-time algorithm that takes a master secret key* msk *and an input* $s \in D_{\mathsf{prf}}$*, and outputs a value* t.

$\mathcal{LE}\mathit{val}(sk, s) \to t$**:** *The leased evaluation algorithm is a QPT algorithm that takes a secret key* sk *and an input* $s \in D_{\mathsf{prf}}$*, and outputs a value* t.

$\mathcal{D}\mathit{el}(sk) \to \mathsf{cert}$**:** *The deletion algorithm is a QPT algorithm that takes a secret key* sk*, and outputs a classical string* cert.

$\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) \to \top/\bot$**:** *The deletion verification algorithm is a deterministic classical polynomial-time algorithm that takes a transcript* τ *of the interactive key generation, a deletion verification key* dvk*, and a deletion certificate* cert*, and outputs* $\top$ *or* $\bot$.

Evaluation correctness: *For every* $s \in D_{\mathsf{prf}}$*, we have*

$$\Pr\left[\mathcal{LE}\mathit{val}(sk, s) = \mathsf{Eval}(\mathsf{msk}, s) \;\middle|\; \begin{array}{l}(\mathsf{pp}, \mathsf{msk}, \mathsf{dvk}) \leftarrow \mathsf{Setup}(1^\lambda)\\ (\tau, sk) \leftarrow \mathit{IntKeyGen}(\mathsf{pp}, \mathsf{msk})\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Deletion verification correctness: *We have*

$$\Pr\left[\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \top \;\middle|\; \begin{array}{l}(\mathsf{pp}, \mathsf{msk}, \mathsf{dvk}) \leftarrow \mathsf{Setup}(1^\lambda)\\ (\tau, sk) \leftarrow \mathit{IntKeyGen}(\mathsf{pp}, \mathsf{msk})\\ \mathsf{cert} \leftarrow \mathcal{D}\mathit{el}(sk)\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Remark 5.12 (Syntax). Similarly to the case of PKE-SKL, one could consider the syntax where Setup and $\mathit{IntKeyGen}$ are merged. However, we chose the above syntax because it allows the PRF key to be generated without the interaction with the lessee.

Remark 5.13 (On the number of rounds in key generation). Similarly to the case of PKE-SKL, at least two rounds of communication are necessary for $\mathit{IntKeyGen}$. Otherwise, it cannot be secure as an adversary can generate multiple copies of the quantum secret key. Our construction only requires 2 rounds and thus is round-optimal in this sense.

Remark 5.14 (Reusability). We can assume without loss of generality that a key of a PRF-SKL scheme is reusable, i.e., it can be reused to evaluate on (polynomially) many inputs. This is because the output of the leased evaluation algorithm is almost unique by evaluation correctness, and thus such an operation can be done without almost disturbing the input state by the gentle measurement lemma [Win99].

Definition 5.15 (PR-VRA security). *We say that a PRF-SKL scheme* PRFSKL *with classical lessor for the domain* D_{prf} *and the range* R_{prf} *is PR-VRA secure, if it satisfies the following requirement, formalized by the experiment* $\mathsf{Exp}^{\mathsf{pr\text{-}vra}}_{\mathsf{PRFSKL},\mathcal{A}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and the challenger:*

1. *The challenger runs* $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk}, \mathsf{dvk})$ *and sends* pp *to* $\mathcal{A}$.
2. *Given* pp *from the challenger,* $\mathcal{A}$ *starts to run. Throughout the game,* $\mathcal{A}$ *has an oracle access to the evaluation oracle* $\mathsf{Eval}(\mathsf{msk}, \cdot)$*, which takes as input* $s \in D_{\mathsf{prf}}$ *and returns* $\mathsf{Eval}(\mathsf{msk}, s)$*, until it receives the challenge input at Step 5 below.* $\mathcal{A}$ *can access the oracle arbitrarily many times at any time.*
3. *The challenger plays the role of the lessor and runs* $\mathit{IntKeyGen}(\mathsf{pp}, \mathsf{msk})$ *with* $\mathcal{A}$*. If the output is* $\bot$*, the challenger outputs* 0 *as the output of the game. Let* τ *be the transcript of the execution of* $\mathit{IntKeyGen}$*.*
4. $\mathcal{A}$ *continues to have an oracle access to the evaluation oracle. At some point,* $\mathcal{A}$ *sends* cert *to the challenger.*
5. *If* $\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \bot$*, the challenger outputs* 0 *as the final output of this experiment. Otherwise, the challenger generates* $s^* \leftarrow D_{\mathsf{prf}}$*,* $t_0^* \leftarrow \mathsf{Eval}(\mathsf{msk}, s^*)$*, and* $t_1^* \leftarrow R_{\mathsf{prf}}$*, and sends* $(\mathsf{dvk}, s^*, t^*_{\mathsf{coin}})$ *to* $\mathcal{A}$*. After receiving the challenge input* s^**,* $\mathcal{A}$ *loses its access to the evaluation oracle.*
6. $\mathcal{A}$ *outputs a guess* coin' *for* coin*. The challenger outputs* coin' *as the final output of the experiment.*

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

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

Remark 5.16. While PR-VRA security defined as above *does* imply security as a weak PRF, it does not imply the security as a standard PRF. However, as observed in [KMY25], we can add standard PRF security by a very simple conversion: run a standard PRF scheme in parallel and take the XOR of the outputs. Thus, we focus on PR-VRA secure construction.

As an intermediate goal towards constructing PRF-SKL, we introduce a primitive which we call unpredictable functions with secure key leasing (UPF-SKL).[19]

Definition 5.17 (UPF-SKL). *A UPF-SKL scheme* UPFSKL *with classical lessor has the same syntax as PRF-SKL with classical lessor and satisfies the following security, which we call UP-VRA security, formalized by the experiment* $\mathsf{Exp}^{\mathsf{up\text{-}vra}}_{\mathsf{UPFSKL},\mathcal{A}}(1^\lambda)$ *between an adversary* $\mathcal{A}$ *and the challenger:*

1. *The challenger runs* $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk}, \mathsf{dvk})$ *and sends* pp *to* $\mathcal{A}$.
2. *Given* pp *from the challenger,* $\mathcal{A}$ *starts to run. Throughout the game,* $\mathcal{A}$ *has an oracle access to the evaluation oracle* $\mathsf{Eval}(\mathsf{msk}, \cdot)$*, which takes as input* $s \in D_{\mathsf{prf}}$ *and returns* $\mathsf{Eval}(\mathsf{msk}, s)$*, until it receives the challenge input at Step 5 below.* $\mathcal{A}$ *can access the oracle arbitrarily many times at any time.*
3. *The challenger plays the role of the lessor and runs* $\mathit{IntKeyGen}(\mathsf{pp}, \mathsf{msk})$ *with* $\mathcal{A}$*. If the output is* $\bot$*, the challenger outputs* 0 *as the output of the game. Let* τ *be the transcript of the execution of* $\mathit{IntKeyGen}$*.*
4. $\mathcal{A}$ *continues to have an oracle access to the evaluation oracle. At some point,* $\mathcal{A}$ *sends* cert *to the challenger.*
5. *If* $\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \bot$*, the challenger outputs* 0 *as the final output of this experiment. Otherwise, the challenger generates* $s^* \leftarrow D_{\mathsf{prf}}$*, and sends* (dvk, s^*) *to* $\mathcal{A}$*. After receiving the challenge input* s^**,* $\mathcal{A}$ *loses its access to the evaluation oracle.*
6. $\mathcal{A}$ *outputs a guess* t' *for the output on* s^**. The challenger outputs* 1 *if* $t' = \mathsf{Eval}(\mathsf{msk}, s^*)$ *and otherwise outputs* 0*.*

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

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

By the quantum Goldreich-Levin lemma [AC02, CLLZ21] we have the following theorem. The proof is almost identical to that of [KMY25, Lemma 4.14], and thus we omit it.

Lemma 5.18. *If there exists a UP-VRA secure UPF-SKL scheme with classical lessor, then there exists a PR-VRA secure PRF-SKL scheme with classical lessor.*

[19] Though UPF-SKL and PRF-SKL are syntactically identical, we treat them as different primitives for clarity.

5.3 Digital Signatures with Secure Key Leasing

In this subsection, we define digital signatures with secure key leasing (DS-SKL) with classical lessors.

Definition 5.19 (DS-SKL with classical lessors). *A DS-SKL scheme* DSSKL *with classical lessors is a tuple of seven algorithms* $(\mathsf{Setup}, \mathit{IntKeyGen}, \mathsf{Sign}, \mathit{Sign}, \mathsf{SigVrfy}, \mathit{Del}, \mathsf{DelVrfy})$*. Below, let* $\mathcal{M}$ *be the message space of* DSSKL*.*

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

$\mathit{IntKeyGen}(\mathsf{svk}, \mathsf{msk}) \to (\tau, \mathit{sigk})$ **or** $(\tau, \bot)$**:** *The interactive key generation algorithm is an interactive protocol with classical communication run between a classical PPT lessor who takes as input* msk *and a QPT lessee on input* svk*. This either outputs a secret key* sigk *or* $\bot$*, which indicates that the protocol failed because the lessee cheats. After the protocol, the lessee obtains* sigk*. We write* τ *to denote the transcript of the execution of* $\mathit{IntKeyGen}$*.*

$\mathsf{Sign}(\mathsf{msk}, m) \to \sigma$**:** *The (lessor's) signing algorithm is a PPT algorithm that takes a master secret key* msk *and a message* $m \in \mathcal{M}$*, and outputs a signature* σ*.*

$\mathit{Sign}(\mathit{sigk}, m) \to (\mathit{sigk}', \sigma)$**:** *The (lessee's) signing algorithm is a QPT algorithm that takes a signing key* sigk *and a message* $m \in \mathcal{M}$*, and outputs a subsequent signing key* sigk' *and a signature* σ*.*

$\mathsf{SigVrfy}(\mathsf{svk}, m, \sigma) \to \top/\bot$**:** *The signature verification algorithm is a deterministic classical polynomial-time algorithm that takes a signature verification key* svk*, a message* $m \in \mathcal{M}$*, and a signature* σ*, and outputs* $\top$ *or* $\bot$*.*

$\mathit{Del}(\mathit{sigk}) \to \mathsf{cert}$**:** *The deletion algorithm is a QPT algorithm that takes a signing key* sigk*, and outputs a deletion certificate* cert*.*

$\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) \to \top/\bot$**:** *The deletion verification algorithm is a deterministic classical polynomial-time algorithm that takes a transcript* τ *of the interactive key generation, a deletion verification key* dvk*, and a deletion certificate* cert*, and outputs* $\top$ *or* $\bot$*.*

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

$$\Pr\left[\begin{array}{l|l} \mathsf{SigVrfy}(\mathsf{svk}, m, \sigma) = \top & (\mathsf{svk}, \mathsf{msk}, \mathsf{dvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \wedge & (\tau, \mathit{sigk}) \leftarrow \mathit{IntKeyGen}(\mathsf{svk}, \mathsf{msk}) \\ \mathsf{SigVrfy}(\mathsf{svk}, m, \sigma') = \top & (\mathit{sigk}', \sigma) \leftarrow \mathit{Sign}(\mathit{sigk}, m) \\ & \sigma' \leftarrow \mathsf{Sign}(\mathsf{msk}, m) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Deletion verification correctness: *We have*

$$\Pr\left[\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \top \;\middle|\; \begin{array}{l} (\mathsf{svk}, \mathsf{msk}, \mathsf{dvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\tau, \mathit{sigk}) \leftarrow \mathit{IntKeyGen}(\mathsf{svk}, \mathsf{msk}) \\ \mathsf{cert} \leftarrow \mathit{Del}(\mathit{sigk}) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Reusability with static signing keys: *Let* sigk *be an honestly generated signing key and* $m \in \mathcal{M}$ *be any message. Suppose that we run* $(\mathit{sigk}', \sigma) \leftarrow \mathit{Sign}(\mathit{sigk}, m)$*. Then we have*

$$\|\mathit{sigk} - \mathit{sigk}'\|_{tr} = \mathsf{negl}(\lambda).$$

Remark 5.20 (Syntax). Similarly to the case of PRF-SKL, one could consider the syntax where Setup and $\mathit{IntKeyGen}$ are merged. However, we chose the above syntax because it allows the signature to be generated without the interaction with the lessee.

Remark 5.21 (On the number of rounds in key generation). Similarly to the case of PRF-SKL, at least two rounds of communication are necessary for $\mathit{IntKeyGen}$. Otherwise, it cannot be secure as an adversary can generate multiple copies of the quantum signing key. Our construction only requires 2 rounds and thus is round-optimal in this sense.

Remark 5.22 (Reusability). Following [KMY25], our syntax makes the quantum signing algorithm output sigk' and requires that it should be close to sigk. We call this property reusability with static signing keys. This is stronger than the notion of reusability defined in [MPY24], where sigk' is not required to be close to sigk as long as it can be still used to generate signatures on further messages. Note that in contrast to PKE-SKL and PRF-SKL scenarios, we cannot take for granted the (even weaker form of) reusability without a loss of generality because the signing algorithm may not produce a unique signature.

We next introduce the security notions for DS-SKL with classical lessors.

Definition 5.23 (RUF-VRA security). *We say that a DS-SKL scheme* DSSKL *with classical lessor for the message space* $\mathcal{M}$ *is RUF-VRA secure,*[20] *if it satisfies the following requirement, formalized by the experiment* $\mathsf{Exp}^{\mathsf{ruf\text{-}vra}}_{\mathsf{DSSKL},\mathcal{A}}(1^\lambda)$ *between an adversary* $\mathcal{A}$ *and the challenger:*

1. *The challenger runs* $\mathsf{Setup}(1^\lambda) \to (\mathsf{svk}, \mathsf{msk}, \mathsf{dvk})$ *and sends* svk *to* $\mathcal{A}$.
2. *Given* svk *from the challenger,* $\mathcal{A}$ *starts to run. Throughout the game,* $\mathcal{A}$ *has an oracle access to the signing oracle* $\mathsf{Sign}(\mathsf{msk}, \cdot)$*, which takes as input* $m \in \mathcal{M}$ *and returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{msk}, m)$*, until it receives the challenge message* m^* *at Step 5 below.* $\mathcal{A}$ *can access the oracle arbitrary many times at any time.*
3. *The challenger plays the role of the lessor and runs* $\mathit{IntKeyGen}(\mathsf{svk}, \mathsf{msk})$ *with* $\mathcal{A}$*. If the output is* $\bot$*, the challenger outputs* 0 *as the output of the game. Let* τ *be the transcript of the execution of* $\mathit{IntKeyGen}$.
4. $\mathcal{A}$ *continues to have an oracle access to the signing oracle. At some point,* $\mathcal{A}$ *sends* cert *to the challenger.*
5. *If* $\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert}) = \bot$*, the challenger outputs* 0 *as the final output of this experiment. Otherwise, the challenger chooses* $m^* \leftarrow \mathcal{M}$*, and sends* dvk *and* m^* *to* $\mathcal{A}$*. After receiving the challenge input* m^**,* $\mathcal{A}$ *loses its access to the signing oracle.*
6. $\mathcal{A}$ *outputs a signature* σ'*. The challenger outputs* 1 *if* $\mathsf{SigVrfy}(\mathsf{svk}, m^*, \sigma') = \top$ *and otherwise outputs* 0.

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

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

Remark 5.24. We note that RUF-VRA security defined as above does not imply the standard EUF-CMA security as a plain digital signature. However, as observed in [KMY25], we can add EUF-CMA security by a very simple conversion: run the EUF-CMA secure signature scheme in parallel, signing the message with both signature schemes, and only accept verification when both signatures are valid. Thus, we focus on constructing RUF-VRA.

6 PKE-SKL with Classical Lessor

The construction of PKE-SKL with a classical lessor is presented in Section 6.1, following the syntax in Definition 5.1. We further show how this scheme can be modified to support non-interactive key generation, as defined in Definition 5.9. Both constructions are proven secure under the LWE assumption.

6.1 Construction with Interactive Key Generation

We construct a PKE-SKL scheme with a classical lessor that satisfies OW-VRA security. Roughly, it requires that if an adversary submits a valid deletion certificate, then it can no longer break the one-wayness of the scheme even given the deletion verification key. As shown by [KMY25], OW-VRA security can be easily upgraded into IND-VRA security, its indistinguishability-based counterpart, by the quantum Goldreich-Levin lemma (Lemma 5.8). For the construction, we use the following ingredients:

[20] "RUF" stands for "**R**andom message **U**n**F**orgeability".

- NTCF generators $\mathsf{NTCF} = \mathsf{NTCF}.(\mathsf{FuncGen}, \mathit{StateGen}, \mathsf{Check}, \mathsf{Invert})$ as defined in Section 2.2. It is associated with a deterministic algorithm NTCF.GoodSet (See Definition 2.10). We assume that NTCF.Check is a deterministic algorithm. We denote the input space by $\mathcal{X}$ and assume $\mathcal{X} \subseteq \{0,1\}^w$ for some polynomial $w = w(\lambda)$. It is known that NTCF generators can be constructed from LWE.
- Special Dual-mode SFE $\mathsf{SFE} = \mathsf{SFE}.(\mathsf{CRSGen}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$ that supports circuits with input space $\{0,1\}^w$ as defined in Section 4. It is associated with an extraction algorithm SFE.Ext, a simulation algorithm SFE.Sim, a state recovery algorithm SFE.StaRcv, and a "coherent version" $\mathsf{SFE}.\mathcal{Rec}_1$ of $\mathsf{SFE}.\mathsf{Rec}_1$, which has the efficient state superposition property. We require that $\mathsf{SFE}.\mathsf{Rec}_2$ and SFE.Ext are deterministic. We denote the length of each block $\alpha^b_{i,j}$ output by SFE.StaRcv by u. As we show in Appendix A, this can be constructed from LWE.
- Watermarkable PKE (with parallel extraction): $\mathsf{WPKE} = \mathsf{WPKE}.(\mathsf{KG}, \mathsf{Mark}, \mathsf{Enc}, \mathsf{Dec})$ as defined in Section 3.1. It is associated with a parallel mark extraction algorithm $\mathsf{WPKE}.\mathcal{Ext}$. We require that WPKE.Mark and WPKE.Dec are deterministic algorithms. We need the mark space to be $\{0,1\}^w$ and denote the message space by $\{0,1\}^\ell$. The length of the decryption key is denoted by v. As we show in Section 3.1, this can be constructed from any PKE.

Let $n = n(\lambda)$ be a positive integer such that $n = \omega(\log \lambda)$.
Below is the description of a OW-VRA secure scheme. The message space of the scheme is $\{0,1\}^{2n\ell}$.

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter, it proceeds as follows.

- For $i \in [2n]$, run $\mathsf{WPKE.KG}(1^\lambda) \to (\mathsf{WPKE.ek}_i, \mathsf{WPKE.msk}_i)$.
- Choose a uniform subset $S \subseteq [2n]$ such that $|S| = n$. In the following, we denote $[2n] \setminus S$ by $\overline{S}$.
- For $i \in [2n]$, run $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, \mathsf{mode})$, where $\mathsf{mode} = \begin{cases} 1 & \text{if } i \in S \\ 2 & \text{if } i \in \overline{S}. \end{cases}$
- Output $\mathsf{ek} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WPKE.ek}_i\}_{i\in[2n]}$, $\mathsf{dvk} = (S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}})$, and $\mathsf{msk} = \{\mathsf{WPKE.msk}_i\}_{i\in[2n]}$.

$\mathit{IntKeyGen}(\mathsf{ek}, \mathsf{msk})$**:** The lessor and lessee run the following interactive protocol to generate a decryption key dk.

Lessee's first operation: On input $\mathsf{ek} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WPKE.ek}_i\}_{i\in[2n]}$, the lessee runs the following algorithms.

- For each $i \in [2n]$, run $\mathsf{NTCF}.\mathit{StateGen}(\mathsf{NTCF.pp}_i) \to (y_i, |\psi_i\rangle)$.
- For each $i \in [2n]$, run $\mathsf{SFE}.\mathcal{Rec}_1(\mathsf{SFE.crs}_i, |\psi_i\rangle) \to (|\psi'_i\rangle, \mathsf{SFE.msg}^{(1)}_i)$.
- Send $\{y_i, \mathsf{SFE.msg}^{(1)}_i\}_{i\in[2n]}$ to the lessor.

Lessor's operation: Given the message $\{y_i, \mathsf{SFE.msg}^{(1)}_i\}_{i\in[2n]}$ from the lessee, the lessor proceeds as follows.

- For $i \in [2n]$, define $C[\mathsf{WPKE.msk}_i, y_i]$ to be a circuit that takes x as input and computes the following:

$$C[\mathsf{WPKE.msk}_i, y_i](x) = \begin{cases} \mathsf{WPKE.Mark}(\mathsf{WPKE.msk}_i, x) & \text{if } \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x, y_i) = \top \\ \bot & \text{if } \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x, y_i) = \bot. \end{cases}$$

  Without loss of generality, we assume that the size of $C[\mathsf{WPKE.msk}_i, y_i]$ does not depend on $\mathsf{WPKE.msk}_i$ and y_i, and denote it by $|C|$.
- For $i \in [2n]$, run $\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}^{(1)}_i, C[\mathsf{WPKE.msk}_i, y_i]) \to \mathsf{SFE.msg}^{(2)}_i$.
- Send $\{\mathsf{SFE.msg}^{(2)}_i\}_{i\in[2n]}$ to the lessee.

Lessee's second operation: Given the message $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$, the lessee proceeds as follows. It uses the secret state $\left|\psi_i'\right\rangle$ in the following.

- Construct the unitary $U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]$ defined by the following map:

$$|x\rangle\,|\mathsf{SFE.st}\rangle\,|0^v\rangle \mapsto |x\rangle\,|\mathsf{SFE.st}\rangle\left|\mathsf{SFE.Rec}_2(\mathsf{SFE.crs}_i, x, \mathsf{SFE.st}, \mathsf{SFE.msg}_i^{(2)})\right\rangle.$$

- For $i \in [2n]$, apply the unitary to compute

$$|\phi_i\rangle \stackrel{\text{def}}{=} U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]\left(\left|\psi_i'\right\rangle|0^v\rangle\right).$$

Final output: At the end of the protocol, the lessee privately obtains a decryption key

$$\mathit{dk} = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle.^{21}$$

We write τ to denote the transcript of the execution of $\mathit{IntKeyGen}$.

$\mathsf{Enc}(\mathsf{ek}, m)$: Upon receiving $\mathsf{ek} = \{\mathsf{WPKE.ek}_i\}_{i\in[2n]}$ and a message $m = m_1\|m_2\|\ldots\|m_{2n} \in \{0,1\}^{2n\ell}$, generate $\mathsf{WPKE.ct}_i \leftarrow \mathsf{WPKE.Enc}(\mathsf{WPKE.ek}_i, m_i)$ for $i \in [2n]$ and output $\mathsf{ct} = \{\mathsf{WPKE.ct}_i\}_{i\in[2n]}$.

$\mathsf{Dec}(\mathsf{msk}, \mathsf{ct})$: Given $\mathsf{msk} = \{\mathsf{WPKE.msk}_i\}_{i\in[2n]}$ and a ciphertext $\mathsf{ct} = \{\mathsf{WPKE.ct}_i\}_{i\in[2n]}$, the decryption algorithm runs $\mathsf{WPKE.Dec}(\mathsf{WPKE.msk}_i, \mathsf{WPKE.ct}_i) = m_i$ for $i \in [2n]$ and outputs $m = m_1\|m_2\|\ldots\|m_{2n}$.

$\mathcal{D}ec(\mathit{dk}, \mathsf{ct})$**:** Given $\mathit{dk} = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$ and a ciphertext $\mathsf{ct} = \{\mathsf{WPKE.ct}_i\}_{i\in[2n]}$, the decryption algorithm performs the following for each $i \in [2n]$.

- Construct the unitary $V[\mathsf{WPKE.ct}_i]$ defined by the following map:

$$|x\rangle\,|\mathsf{SFE.st}\rangle\,|\mathsf{WPKE.dk}\rangle\left|0^\ell\right\rangle \mapsto |x\rangle\,|\mathsf{SFE.st}\rangle\,|\mathsf{WPKE.dk}\rangle\,|\mathsf{WPKE.Dec}(\mathsf{WPKE.dk}, \mathsf{WPKE.ct}_i)\rangle.$$

- Compute $V[\mathsf{WPKE.ct}_i]\left(|\phi_i\rangle\left|0^\ell\right\rangle\right)$ and measure the last ℓ qubits to obtain m_i.

It finally outputs $m = m_1\|m_2\|\ldots\|m_{2n}$.

$\mathsf{Del}(\mathit{dk})$**:** Given $\mathit{dk} = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$, the deletion algorithm performs the following for each $i \in [2n]$.

- Measure the corresponding registers to recover the classical strings $\mathsf{SFE.crs}_i$ and $\mathsf{SFE.msg}_i^{(2)}$.
- Compute $U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]^\dagger\,|\phi_i\rangle$ and remove the last v qubits to obtain $\left|\psi_i''\right\rangle$.
- Measure $\left|\psi_i''\right\rangle$ in Hadamard basis to obtain (d_i, c_i), where each $d_i \in \{0,1\}^w$ is obtained by measuring the register corresponding to an element in $\mathcal{X}$ and each $c_i \in \{0,1\}^{uw}$ is obtained by measuring the register corresponding to the SFE state.

The final output of the algorithm is given by $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$,

$\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert})$**:** Given the transcript τ of $\mathit{IntKeyGen}$, the deletion verification key $\mathsf{dvk} = (S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\notin S})$, and the deletion certificate $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$, the deletion verification algorithm works as follows.

[21] $\mathsf{SFE.crs}_i$ and $\mathsf{SFE.msg}_i^{(2)}$ are not needed for decryption. They are used only for deletion.

- Retrieve $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in\overline{S}}$ from τ.
- For $i \in \overline{S}$, run $\mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i) \to (x_{i,0}, x_{i,1})$. If the inversion fails for any of i, output $\bot$.
- For $i \in \overline{S}$, run $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$, where $\alpha_{i,j}^b \in \{0,1\}^u$.
- For $i \in \overline{S}$, compute the vector $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as

$$d_i^*[j] \coloneqq \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle,$$

  where c_i is parsed as $c_i = c_{i,1}\| \cdots \| c_{i,w}$, where each block is in $\{0,1\}^u$.
- For $i \in \overline{S}$, check whether

$$\langle d_i \oplus d_i^*, (x_{i,0} \oplus x_{i,1})\rangle = 0 \ \wedge \ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top$$

  holds. If it holds for all $i \in \overline{S}$, it outputs $\top$. Otherwise, it outputs $\bot$.

The following theorem asserts the decryption correctness and the deletion verification correctness of the above scheme.

Theorem 6.1. *The above construction satisfies the decryption correctness if* NTCF *satisfies the state generation correctness,* SFE *satisfies evaluation correctness, and* WPKE *satisfies the decryption correctness. Furthermore, the above construction satisfies the deletion verification correctness if* NTCF *satisfies the state generation correctness,* SFE *satisfies the statistical security against malicious senders, and* SFE *supports state recovery in the hiding mode.*

Proof. To show the decryption correctness and deletion correctness, we first observe that all the algorithms run the subsystems indexed by $i \in [2n]$, which run in parallel. Therefore, it suffices to show that each subsystem succeeds in decryption (i.e., $m_i' = m_i$) and deletion verification (i.e., (d_i, c_i) passes the verification) with overwhelming probability assuming that the lessee follows the protocol honestly. Here, we show this for the case of $i \in \overline{S}$. The case of $i \in S$ is easier to handle, since in this case, $|\phi_i\rangle$ collapses to a classical state corresponding to $x_i = \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$ with overwhelming probability and repeating the same discussion as the following in its collapsed form suffices.

We first analyze the quantum state that the lessee has right after the interactive key generation protocol ends. We observe that by the correctness of state generation of NTCF, with overwhelming probability, $y_i \in \mathcal{Y}_{\mathsf{pp}}$. In addition, by the same property, $|\psi_i\rangle$ is negligibly close to $\frac{1}{\sqrt{2}}(|x_{i,0}\rangle + |x_{i,1}\rangle)$ in the trace distance, with overwhelming probability, where $(x_{i,0}, x_{i,1}) = \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$. In the following, we replace $|\psi_i\rangle$ with $\frac{1}{\sqrt{2}}(|x_{i,0}\rangle + |x_{i,1}\rangle)$ and continue the analysis, since this change alters the probability of the decryption/verification succeeds only with negligible amount. Then, $\mathsf{SFE}.\mathcal{R}\mathit{ec}_1$ is applied to the state. In the first step, we obtain a state that is negligibly close to the state obtained by applying the isometry defined by the following map to $\frac{1}{\sqrt{2}}(|x_{i,0}\rangle + |x_{i,1}\rangle)$:

$$|x_{i,b}\rangle \mapsto \sum_{\mathsf{SFE.msg}_i^{(1)}\in\mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}_i, x_{i,b}))} \sqrt{p_{x_{i,b}\to\mathsf{SFE.msg}_i^{(1)}}}\, |x_{i,b}\rangle \left|\mathsf{SFE.st}_{x_{i,b}\to\mathsf{SFE.msg}_i^{(1)}}\right\rangle \left|\mathsf{SFE.msg}_i^{(1)}\right\rangle,$$

where $p_{x_{i,b}\to\mathsf{SFE.msg}_i^{(1)}}$ is the probability that $\mathsf{SFE.Rec}_1(\mathsf{SFE.crs}_i, x_{i,b})$ outputs $\mathsf{SFE.msg}_i^{(1)}$ and $\mathsf{SFE.st}_{x_{i,b}\to\mathsf{SFE.msg}_i^{(1)}}$ is the unique state corresponding to $x_{i,b}$ and $\mathsf{SFE.msg}_i^{(1)}$. We again ignore this negligible difference and proceed with the analysis, as this difference affects the probability of decryption/verification succeeding only by a negligible amount. We then have the following state:

$$\sum_{z\in\mathcal{Z}} \sqrt{\frac{p_{i,0,z}}{2}}\, |x_{i,0}\rangle\, |\mathsf{st}_{i,0,z}\rangle\, |z\rangle + \sqrt{\frac{p_{i,1,z}}{2}}\, |x_{i,1}\rangle\, |\mathsf{st}_{i,1,z}\rangle\, |z\rangle,$$

where we simplify the notation by denoting $\mathsf{SFE.msg}_i^{(1)}$ by z, $\mathsf{SFE.st}_{x_{i,b}\to\mathsf{SFE.msg}_i^{(1)}}$ by $\mathsf{st}_{i,b,z}$, $p_{x_{i,b}\to\mathsf{SFE.msg}_i^{(1)}}$ by $p_{i,b,z}$, and $\mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}_i, x_{i,0})) \cup \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}_i, x_{i,1}))$ by $\mathcal{Z}$. We then measure the last register to obtain z^*, where the residual state is

$$|\psi_i'\rangle = q_{i,0,z^*}\, |x_{i,0}\rangle\, |\mathsf{st}_{i,0,z^*}\rangle + q_{i,1,z^*}\, |x_{i,1}\rangle\, |\mathsf{st}_{i,1,z^*}\rangle.$$

In the above, we define $q_{i,b,z^*} := \sqrt{p_{i,b,z^*}/(p_{i,0,z^*}+p_{i,1,z^*})}$. We note that the probability that we obtain z^* by the measurement is $(p_{i,0,z^*}+p_{i,1,z^*})/2$.

We then prove the decryption correctness and the deletion verification correctness separately.

Decryption correctness: We continue the analysis of the state that the lessee obtains. After z^* is sent to the lessor, $\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, z^*, C[\mathsf{WPKE.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$ is sent back to the lessee. We observe that by the evaluation correctness of SFE, we have $\mathsf{SFE.Rec}_2(\mathsf{SFE.crs}_i, x_{i,b}, \mathsf{st}_{i,b,z^*}, \mathsf{SFE.msg}_i^{(2)}) = \mathsf{WPKE.dk}(x_{i,b})$. By the definition of $|\phi_i\rangle$, we have

$$\begin{aligned}|\phi_i\rangle &= \sum_{b\in\{0,1\}} q_{i,b,z^*} |x_{i,b}\rangle |\mathsf{st}_{i,b,z^*}\rangle \left|\mathsf{SFE.Rec}_2(\mathsf{SFE.crs}_i, x_{i,b}, \mathsf{st}_{i,b,z^*}, \mathsf{SFE.msg}_i^{(2)})\right\rangle \\ &= \sum_{b\in\{0,1\}} q_{i,b,z^*} |x_{i,b}\rangle |\mathsf{st}_{i,b,z^*}\rangle |\mathsf{WPKE.dk}_i(x_{i,b})\rangle .\end{aligned}$$

After applying $V[\mathsf{WPKE.ct}_i]$ to $|\phi_i\rangle \left|0^\ell\right\rangle$, we get

$$\begin{aligned}&\sum_{b\in\{0,1\}} q_{i,b,z^*} |x_{i,b}\rangle |\mathsf{st}_{i,b,z^*}\rangle |\mathsf{WPKE.dk}_i(x_{i,b})\rangle |\mathsf{WPKE.Dec}(\mathsf{WPKE.dk}_i(x_{i,b}), \mathsf{WPKE.ct}_i)\rangle \\ &= \sum_{b\in\{0,1\}} q_{i,b,z^*} |x_{i,b}\rangle |\mathsf{st}_{i,b,z^*}\rangle |\mathsf{WPKE.dk}_i(x_{i,b})\rangle |m_i\rangle\end{aligned}$$

by the decryption correctness of WPKE. We can therefore obtain the decryption result m_i by measuring the last register.

Deletion verification correctness: We first observe that $\left|\psi_i''\right\rangle = \left|\psi_i'\right\rangle$, since $\left|\psi_i''\right\rangle$ is obtained by applying a unitary and then its conjugate to $\left|\psi_i'\right\rangle$. The rest of the proof involves 3 main steps: (i) We first show that the lessee obtains (d_i, c_i) such that

$$\langle d_i, (x_{i,0} \oplus x_{i,1})\rangle \oplus \langle c_i, (\mathsf{st}_{i,0,z^*} \oplus \mathsf{st}_{i,1,z^*})\rangle = 0 \tag{5}$$

with overwhelming probability, (ii) $\mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top$ with overwhelming probability, and then (iii) Equation (5) implies $\langle d_i \oplus d_i^*, x_{i,0} \oplus x_{i,1}\rangle = 0$.

We first show (i). By applying the Hadamard gates to $\left|\psi_i'\right\rangle$, we obtain

$$\begin{aligned}&\frac{1}{2^{L/2}} \sum_{(d_i,c_i)\in\{0,1\}^L} \left(q_{i,0,z^*} (-1)^{\langle d_i, x_{i,0}\rangle \oplus \langle c_i, \mathsf{st}_{i,0,z^*}\rangle} + q_{i,1,z^*} (-1)^{\langle d_i, x_{i,1}\rangle \oplus \langle c_i, \mathsf{st}_{i,1,z^*}\rangle}\right) |d_i\rangle |c_i\rangle \\ = \frac{1}{2^{L/2}} \Bigg(&\sum_{(d_i,c_i)\in T} (-1)^{\langle d_i, x_{i,0}\rangle \oplus \langle c_i, \mathsf{st}_{i,0,z^*}\rangle} (q_{i,0,z^*} + q_{i,1,z^*}) |d_i\rangle |c_i\rangle \\ &+ \sum_{(d_i,c_i)\notin T} (-1)^{\langle d_i, x_{i,0}\rangle \oplus \langle c_i, \mathsf{st}_{i,0,z^*}\rangle} (q_{i,0,z^*} - q_{i,1,z^*}) |d_i\rangle |c_i\rangle \Bigg),\end{aligned} \tag{6}$$

where $L = uw + w$ is the length of each string (d_i, c_i) and T is the set of (d_i, c_i) that satisfies Equation (5). Note that since $x_{i,0} \neq x_{i,1}$, we have $|T| = 2^{L-1}$. Therefore, the probability that Equation (5) does not hold, where the probability is taken over the choice of z^*, is:

$$\begin{aligned}\Pr[(d_i, c_i) \notin T] &= \sum_{z^*\in\mathcal{Z}} \frac{p_{i,0,z^*} + p_{i,1,z^*}}{2} \sum_{(d_i,c_i)\in T} \frac{1}{2^L} (q_{i,0,z^*} - q_{i,1,z^*})^2 \\ &= \frac{1}{4} \sum_{z^*\in\mathcal{Z}} (p_{i,0,z^*} + p_{i,1,z^*}) \cdot \frac{(\sqrt{p_{i,0,z^*}} - \sqrt{p_{i,1,z^*}})^2}{p_{i,0,z^*} + p_{i,1,z^*}} \\ &\leq \frac{1}{4} \sum_{z^*\in\mathcal{Z}} |p_{i,0,z^*} - p_{i,1,z^*}|\end{aligned}$$

where the second line follows from $|T| = 2^{L-1}$ and by the definition of q_{i,b,z^*} and the third line holds since we have $|a-b|^2 \leq |a^2 - b^2|$ for any $a, b \geq 0$. Finally, the last quantity is negligible by the statistical security of SFE against the malicious senders, since the statistical distance between the first output $\mathsf{SFE.msg}^{(1)}$ of $\mathsf{SFE.Rec}_1(\mathsf{SFE.crs}_i, x_{i,0})$ and that of $\mathsf{SFE.Rec}_1(\mathsf{SFE.crs}_i, x_{i,1})$ is $\frac{1}{2}\sum_{z^* \in \mathcal{Z}} |p_{i,0,z^*} - p_{i,1,z^*}| = \mathsf{negl}(\lambda)$.

We then prove (ii). We first fix y_i and z^*. These fix the values of $(x_{i,0}, x_{i,1})$ and $\{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$. We then assume $(d_i, c_i) \in T$, since this happens with overwhelming probability as we saw in (i). We then fix arbitrary c_i. This determines the value of d_i^*. By Equation (6), we can see that d_i is distributed uniformly at random under the condition that $(c_i, d_i) \in T$. Furthermore, for any c_i, half of $d_i \in \{0,1\}^w$ satisfies $(c_i, d_i) \in T$. We therefore have

$$\begin{aligned}
&\Pr_{d_i \text{ s.t. } (d_i,c_i)\in T}[\mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \bot] \\
=&2 \cdot \Pr_{d_i \leftarrow \{0,1\}^w}[\mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \bot] \\
=&2 \cdot \Pr_{d_i \leftarrow \{0,1\}^w}[\mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i) = \bot] \\
=&\mathsf{negl}(\lambda).
\end{aligned}$$

We finally prove (iii). Recall that by the state recovery property in the hiding mode of SFE, we have

$$\mathsf{st}_{i,b,z^*} = \alpha_{i,1}^{x_{i,b}[1]} \| \cdots \| \alpha_{i,w}^{x_{i,b}[w]},$$

where $x_{i,b}[j]$ is the j-th bit of $x_{i,b}$. We therefore have

$$\begin{aligned}
\langle c_i, \mathsf{st}_{i,0,z^*} \oplus \mathsf{st}_{i,1,z^*} \rangle &= \sum_{j\in[w]} \left\langle c_{i,j}, \alpha_{i,j}^{x_{i,0}[j]} \oplus \alpha_{i,j}^{x_{i,1}[j]} \right\rangle \\
&= \sum_{j\in[w]} \left\langle c_{i,j}, (x_{i,0}[j] \oplus x_{i,1}[j])(\alpha_{i,j}^0 \oplus \alpha_{i,j}^1) \right\rangle \\
&= \sum_{j\in[w]} (x_{i,0}[j] \oplus x_{i,1}[j]) d_i^*[j] \\
&= \langle d_i^*, x_{i,0} \oplus x_{i,1} \rangle,
\end{aligned}$$

where the second equation follows because $\alpha_{i,j}^{x_{i,0}[j]} \oplus \alpha_{i,j}^{x_{i,1}[j]}$ equals to 0^u if $x_{i,0}[j] = x_{i,1}[j]$ and $\alpha_{i,j}^0 \oplus \alpha_{i,j}^1$ otherwise. Therefore, Equation (5) implies $\langle d_i^* \oplus d_i, x_{i,0} \oplus x_{i,1} \rangle = 0$.

This completes the proof of Theorem 6.1. □

6.2 Security Proof

Theorem 6.2. *If* NTCF *satisfies cut-and-choose adaptive hardcore bit security,* SFE *satisfies the mode-indistinguishability and the extractability in the extraction mode, and* WPKE *satisfies one-wayness and parallel mark extractability, then our construction is OW-VRA secure.*

Proof. We prove the theorem by a sequence of hybrids. For the sake of the contradiction, let us assume an adversary $\mathcal{A}$ that wins the OW-VRA security game with non-negligible probability ϵ. In the following, the event that $\mathcal{A}$ wins in $\mathsf{Hyb}_{\mathrm{xx}}$ is denoted by E_{xx}.

Hyb_0**:** This is the original OW-VRA security game between the challenger and an adversary $\mathcal{A}$. Namely, the game proceeds as follows.

1. The challenger computes ek as follows.
   - It samples $(\mathsf{WPKE.ek}_i, \mathsf{WPKE.msk}_i)$ for $i \in [2n]$.

- It chooses random subset S of $[2n]$ with size n.
- It runs $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, \mathsf{mode})$ for $i \in [2n]$, where $\mathsf{mode} = 1$ if $i \in S$ and $\mathsf{mode} = 2$ if $i \in \overline{S}$.

Finally, it sends $\mathsf{ek} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WPKE.ek}_i\}_{i\in[2n]}$ to $\mathcal{A}$.

2. The adversary $\mathcal{A}$ sends $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ to the challenger.
3. The challenger computes the message to the adversary as follows. It first runs

$$\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WPKE.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$$

for $i \in [2n]$. Then, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.
4. Then, $\mathcal{A}$ submits the deletion certificate $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$.
5. Then, the challenger runs DelVrfy as follows.
   - For $i \in \overline{S}$, it runs $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$.
   - For $i \in \overline{S}$, it computes $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as $d_i^*[j] := \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle$.
   - For $i \in \overline{S}$, it runs $(x_{i,0}, x_{i,1}) \leftarrow \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$. If the inversion fails for any of $i \in \overline{S}$, the output of DelVrfy is $\bot$.
   - For $i \in \overline{S}$, it checks whether

$$(d_i \oplus d_i^*) \cdot (x_{i,0} \oplus x_{i,1}) = 0 \ \wedge \ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top \tag{7}$$

     holds using $\mathsf{NTCF.td}_i$. If Equation (7) holds for all $i \in \overline{S}$, the output is $\top$ and otherwise, $\bot$.

   If the output of the algorithm is $\bot$, then it indicates that the output of the game is 0 (i.e., $\mathcal{A}$ failed to win the game). Otherwise, the challenger returns $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$ to $\mathcal{A}$ and continues the game.
6. Then, the challenger chooses a random message $m = m_1 \| m_2 \| \cdots \| m_{2n} \leftarrow \{0,1\}^{2n\ell}$ and computes $\mathsf{WPKE.ct}_i \leftarrow \mathsf{WPKE.Enc}(\mathsf{WPKE.ek}_i, m_i)$ for $i \in [2n]$. Then, it sends $\mathsf{ct} = \{\mathsf{WPKE.ct}_i\}_{i\in[2n]}$ to $\mathcal{A}$.
7. $\mathcal{A}$ then outputs $m' = m'_1 \| \cdots \| m'_{2n}$. The adversary wins the game if $m' = m$.

By the definition, we have $\Pr[\mathsf{E}_0] = \epsilon$.

Hyb_1**:** In this hybrid, the challenger computes $\mathsf{SFE.msg}_i^{(2)}$ differently for $i \in S$ in Step 3 of the game. Namely, it is replaced with the following:

3' The challenger computes the message to the adversary as follows.
   - It runs $\mathsf{SFE.Ext}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to x_i$ and $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \mathsf{vrd}_i$ for $i \in S$.
   - It computes

$$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \gamma_i) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WPKE.msk}_i, y_i]) & \text{if } i \in \overline{S} \end{cases},$$

   where

$$\gamma_i = \begin{cases} \mathsf{WPKE.Mark}(\mathsf{WPKE.msk}_i, x_i) = \mathsf{WPKE.dk}_i(x_i) & \text{if } \mathsf{vrd}_i = \top \\ \bot & \text{if } \mathsf{vrd}_i = \bot. \end{cases}$$

   Finally, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.

Since we have $\gamma_i = C[\mathsf{WPKE.msk}_i, y_i](x_i)$, it follows that this game is indistinguishable from the previous game by a straightforward reduction to the extractability of SFE in the extraction mode. Thus, we have $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\lambda)$. Note that in this game, $\mathsf{WPKE.msk}_i$ is not used by the game for i such that $i \in S$ and $\mathsf{vrd}_i = \bot$.

Hyb_2**:** In this hybrid, the challenger aborts the game and sets the output of the game to be 0 (i.e., $\mathcal{A}$ loses the game), once it turns out that there is $i \in S$ such that $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \bot$. Namely, we replace Step 3' in the previous hybrid with the following:

3" The challenger computes the message to the adversary as follows.

- It runs $\mathsf{SFE.Ext}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to x_i$ and $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \mathsf{vrd}_i$ for $i \in S$.
- If $\mathsf{vrd}_i = \bot$ for any of $i \in S$, it aborts the game and sets the output of the game to be 0.
- It computes

$$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \mathsf{WPKE.dk}_i(x_i)) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WPKE.msk}_i, y_i]) & \text{if } i \in \overline{S}. \end{cases}$$

Finally, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.

We claim $\Pr[\mathsf{E}_2] \geq \Pr[\mathsf{E}_1] - \mathsf{negl}(\lambda)$. To show this, we observe that the only scenario where the adversary $\mathcal{A}$ wins the game in the previous hybrid but not in the current one occurs when $\mathcal{A}$ chooses y_i such that $\mathsf{vrd}_i = \bot$ for some $i \in S$, yet still succeeds in inverting the ciphertext to get $\{m_j\}_{j\in[2n]}$. However, this implies that $\mathcal{A}$ has inverted $\mathsf{WPKE.ct}_i$ to obtain m_i without getting any secret information corresponding to $\mathsf{WPKE.ek}_i$. By a straightforward reduction to the one-wayness of WPKE, the claim follows.

Hyb_3**:** In this hybrid, we change the winning condition of the adversary. Namely, we replace Step 6 and Step 7 of the game with the following.

6' The challenger parses the adversary $\mathcal{A}$ right before Step 6 of the game as a unitary circuit $U_{\mathcal{A}}$ and a quantum state $\rho_{\mathcal{A}}$. It then constructs a quantum decryption algorithm $\mathcal{D}$ that takes as input $\{\mathsf{WPKE.ct}_i\}_{i\in S}$ as input and outputs $\{m'_i\}_{i\in S}$ from $\{\mathsf{WPKE.ek}_i\}_{i\in\overline{S}}$, S, and $\mathcal{A}$ as follows.

$\mathcal{D}(\{\mathsf{WPKE.ct}_i\}_{i\in S})$**:** It chooses $m_i \leftarrow \mathcal{M}$ for $i \in \overline{S}$ and computes $\mathsf{WPKE.ct}_i \leftarrow \mathsf{WPKE.Enc}(\mathsf{WPKE.ek}_i, m_i)$ for $i \in \overline{S}$. Then, it runs $\mathcal{A}$ on input $\{\mathsf{WPKE.ct}_i\}_{i\in[2n]}$ to obtain $m' = m'_1\|\cdots\|m'_{2n}$. Finally, it outputs $\{m'_i\}_{i\in S}$.

7' The challenger parses $\mathcal{D}$ as a pair of unitary $U_{\mathcal{D}}$ and a quantum state $\rho_{\mathcal{D}}$. Then, it runs the extraction algorithm as $\mathsf{WPKE}.\mathcal{E}\mathcal{X}\mathsf{t}(\{\mathsf{WPKE.ek}_i\}_{i\in S}, (U_{\mathcal{D}}, \rho_{\mathcal{D}})) \to \{x'_i\}_{i\in S}$.

8' If $x_i = x'_i$ holds for all $i \in S$, the challenger sets the output of the game to be 1. Otherwise, it sets it to be 0.

As we show in Lemma 6.3, we have $\Pr[\mathsf{E}_3] \geq \Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda)$ by the parallel extractability of WPKE.

Hyb_4**:** In this game, we replace Step 8' of the above game with the following:

8" If $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x'_i, y_i) = \top$ holds for all $i \in S$, the challenger sets the output of the game to be 1. Otherwise, it sets it to be 0.

We observe that $x'_i = x_i$ in Step 8' implies $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x'_i, y_i) = \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \top$. To see the latter equality holds, recall that the game is set to abort if $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \bot$ for any $i \in S$ in Step 3". Thus, if the game continues to Step 8', the equality holds. We therefore have $\Pr[\mathsf{E}_4] \geq \Pr[\mathsf{E}_3]$. (Actually, $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_3]$ holds due to the uniqueness of x_i that passes the check. However, showing the inequality suffices for our proof.)

Hyb_5**:** In this game, we switch Step 3" in the previous hybrid back to Step 3'. The difference from the previous hybrid is that it continues the game even if $\mathsf{vrd}_i = \bot$ for some $i \in S$. Since the adversary had no chance in winning the game in such a case in the previous hybrid, this change only increases the chance of the adversary winning. We therefore have $\Pr[\mathsf{E}_5] \geq \Pr[\mathsf{E}_4]$.

Hyb_6**:** In this hybrid, we switch Step 3' in the previous hybrid back to Step 3. Namely, it computes $\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WPKE.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$ for all $i \in [2n]$ again. Similarly to the change from Hyb_0 to Hyb_1, the winning probability of $\mathcal{A}$ in this game does not change significantly by the extractability of SFE in the extraction mode. Namely, we have $|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_5]| = \mathsf{negl}(\lambda)$. Note that we do not need $\{\mathsf{SFE.td}_i\}_{i\in S}$ any more to run the game.

Hyb_7**:** In this hybrid, we change the way $\mathsf{SFE.crs}_i$ is sampled for $i \in S$. Namely, it is sampled as $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, 2)$ for all $i \in [2n]$. Concretely, we replace Step 1 with the following:

1' The challenger computes the first message to the adversary as follows.

- It samples $(\mathsf{WPKE.ek}_i, \mathsf{WPKE.msk}_i)$ for $i \in [2n]$.
- It chooses random subset S of $[2n]$ with size n.
- It runs $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, 2)$ for $i \in [2n]$, where $\mathsf{mode} = 1$ if $i \in S$ and $\mathsf{mode} = 2$ if $i \in \overline{S}$.

Finally, it sends $\mathsf{ek} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WPKE.ek}_i\}_{i\in[2n]}$ to $\mathcal{A}$.

Since we do not need $\mathsf{SFE.td}_i$ for $i \in S$ to run Hyb_6 and Hyb_7, a straightforward reduction to the mode indistinguishability of SFE implies that Hyb_6 and Hyb_7 are computationally indistinguishable. Therefore, we have $|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_7]| = \mathsf{negl}(\lambda)$.

Hyb_8**:** In this game, we change the way how the challenger runs DelVrfy. In particular, the challenger computes $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j,b}$ and d_i^* for all $i \in [2n]$, instead of only for $i \in \overline{S}$. Still, the inversion of y_i and the check of Equation (7) are performed only for $i \in \overline{S}$. Concretely, we replace Step 5 with the following.

5' Then, the challenger runs (the modified version of) DelVrfy as follows.

- For $i \in [2n]$, it runs $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$.
- For $i \in [2n]$, it computes $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as $d_i^*[j] := \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle$.
- For $i \in \overline{S}$, it runs $(x_{i,0}, x_{i,1}) \leftarrow \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$. If the inversion fails for any of $i \in \overline{S}$, the output of DelVrfy is $\bot$.
- For $i \in \overline{S}$, it checks whether

$$(d_i \oplus d_i^*) \cdot (x_{i,0} \oplus x_{i,1}) = 0 \ \wedge \ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top \tag{8}$$

holds using $\mathsf{NTCF.td}_i$. If Equation (8) holds for all $i \in \overline{S}$, the output is $\top$ and otherwise, $\bot$.

If the output of the algorithm is $\bot$, then it indicates that the output of the game is 0 (i.e., $\mathcal{A}$ failed to win the game). Otherwise, the challenger returns $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$ to $\mathcal{A}$ and continues the game.

We can see that this change is only conceptual, since the outcome of the verification is unchanged. We may therefore have $\Pr[\mathsf{E}_8] = \Pr[\mathsf{E}_7]$.

Based on the above discussion, it is established that $\Pr[\mathsf{E}_8] \geq \epsilon - \mathsf{negl}(\lambda)$, which is non-negligible by our assumption. However, as we show in Lemma 6.4, we have $\Pr[\mathsf{E}_8] = \mathsf{negl}(\lambda)$ assuming cut-and-choose adaptive hardcore bit property of NTCF. This results in a contradiction as desired. □

It remains to prove Lemma 6.3 and Lemma 6.4.

Lemma 6.3. *If* WPKE *satisfies parallel extractability, we have* $\Pr[\mathsf{E}_3] \geq \Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda)$.

Proof. We consider an adversary $\mathcal{B}$ who plays the role of the challenger in Hyb_2 for $\mathcal{A}$, while working as an adversary against the parallel one-way inversion experiment with the number of instances n. It proceeds as follows:

1. Given $\{\mathsf{WPKE.ek}'_j\}_{j\in[n]}$ from the challenger, $\mathcal{B}$ proceeds as follows.
   - $\mathcal{B}$ chooses $S = \{i_1, \ldots, i_n\} \subset [2n]$ and samples $\{\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i, \mathsf{SFE.crs}_i, \mathsf{SFE.td}_i\}_{i\in[2n]}$ as specified in Hyb_2.
   - It sets $\mathsf{WPKE.ek}_{i_j} := \mathsf{WPKE.ek}'_j$ for $j \in [n]$.
   - It then samples $(\mathsf{WPKE.ek}_i, \mathsf{WPKE.msk}_i) \leftarrow \mathsf{WPKE.KG}(1^\lambda)$ for $i \in \overline{S}$.
   - Then, it sends $\mathsf{ek} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WPKE.ek}_i\}_{i\in[2n]}$ to $\mathcal{A}$.
   - Then, the adversary $\mathcal{A}$ sends $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ to $\mathcal{B}$.
   - $\mathcal{B}$ then computes x_i for $i \in S$ and vrd_i using the trapdoors for SFE and NTCF as in Hyb_2. If $\mathsf{vrd}_i = \bot$ for any of $i \in S$, it aborts.
   - It then sets $\{x'_1, \ldots, x'_n\} := \{x_{i_1}, \ldots, x_{i_n}\}$ and sends it to its challenger.
2. Given $\{\mathsf{WPKE.dk}'_j(x'_j)\}_{j\in[n]}$ from the challenger, $\mathcal{B}$ proceeds as follows.
   - It sets $\mathsf{WPKE.dk}_{i_j}(x_{i_j}) := \mathsf{WPKE.dk}'_j(x'_j)$ for $j \in [n]$.
   - It computes $\mathsf{SFE.msg}_i^{(2)}$ as follows:
   $$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \mathsf{WPKE.dk}_i(x_i)) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WPKE.msk}_i, y_i]) & \text{if } i \in \overline{S}. \end{cases}$$
   - $\mathcal{B}$ then sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.
   - Then, $\mathcal{A}$ submits the deletion certificate cert to $\mathcal{B}$. $\mathcal{B}$ runs DelVrfy using the trapdoors for NTCF and SFE. If the output is $\bot$, $\mathcal{B}$ also outputs $\bot$ and aborts. Otherwise, $\mathcal{B}$ returns $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i\}_{i\in\overline{S}}\right)$ to $\mathcal{A}$.
3. Then, the challenger sends $\{\mathsf{WPKE.ct}'_j\}_{j\in[n]}$ to $\mathcal{B}$.
4. $\mathcal{B}$ computes its output as follows.
   - It sets $\mathsf{WPKE.ct}_{i_j} := \mathsf{WPKE.ct}'_j$ for $j \in [n]$.
   - It samples $m_i \leftarrow \mathcal{M}$ and computes $\mathsf{WPKE.ct}_i \leftarrow \mathsf{WPKE.Enc}(\mathsf{WPKE.ek}_i, m_i)$ for $i \in \overline{S}$.
   - It then sends $\{\mathsf{WPKE.ct}_i\}_{i\in[2n]}$ to $\mathcal{A}$.
   - $\mathcal{A}$ output $(m'_1, \ldots, m'_{2n})$. It then outputs $(m'_{i_1}, \ldots, m'_{i_n})$ as its output.

We can see that $\mathcal{B}$ defined above perfectly simulates Hyb_2 for $\mathcal{A}$ and $\mathcal{B}$ wins the game if so does $\mathcal{A}$. We therefore have $\mathsf{Adv}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{B},n}(\lambda) \geq \Pr[\mathsf{E}_2]$. Furthermore, by the parallel extractability of WPKE, we have $\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{B},\mathcal{EXT},n}(\lambda) \geq \mathsf{Adv}^{\mathsf{par\text{-}ow}}_{\mathsf{WPKE},\mathcal{B},n}(\lambda) - \mathsf{negl}(\lambda)$. We can also see that $\Pr[\mathsf{E}_3] = \mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WPKE},\mathcal{B},\mathcal{EXT},n}(\lambda)$ holds, because the way $\mathcal{B}$ inverts the ciphertexts is exactly the same as the way $\mathcal{D}$ does in Hyb_3. By combining these equations, the lemma follows. □

Lemma 6.4. *If* NTCF *satisfies the cut-and-choose adaptive hardcore bit property, we have* $\Pr[\mathsf{E}_8] = \mathsf{negl}(\lambda)$.

Proof. To show the lemma, we construct an adversary $\mathcal{B}$ who plays the role of the challenger in Hyb_8 for $\mathcal{A}$ and tries to break the cut-and-choose adaptive hardcore bit property of NTCF. $\mathcal{B}$ proceeds as follows.

1. Given $\{\mathsf{NTCF.pp}_i\}_{i\in[2n]}$ from its challenger, $\mathcal{B}$ computes the first message to its challenger as follows.

- It runs $(\mathsf{WPKE.ek}_i, \mathsf{WPKE.msk}_i) \leftarrow \mathsf{WPKE}(1^\lambda)$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, 2)$ for $i \in [2n]$ and sends $\mathsf{ek} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WPKE.ek}_i\}_{i\in[2n]}$ to $\mathcal{A}$.
- $\mathcal{A}$ returns $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ to $\mathcal{B}$.
- $\mathcal{B}$ computes $\mathsf{SFE.Send}(\mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WPKE.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$ for $i \in [2n]$. It then sends $\{\mathsf{SFE.msg}^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.
- $\mathcal{A}$ then submits the deletion certificate $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$ to $\mathcal{B}$.
- $\mathcal{B}$ runs $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$ for $i \in [2n]$. It then computes d_i^* from c_i and $\{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$ for $i \in [2n]$ and sends $\{y_i, d_i \oplus d_i^*\}_{i\in[2n]}$ to the challenger.

2. If $\mathcal{B}$ passes the verification, the challenger gives S and $\{\mathsf{NTCF.td}_i\}_{i\in\overline{S}}$ to $\mathcal{B}$. Then, $\mathcal{B}$ gives $\mathsf{dvk} = (S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}})$ to $\mathcal{A}$.
3. $\mathcal{B}$ then constructs a quantum decryption algorithm $\mathcal{D} = (U_\mathcal{D}, \rho_\mathcal{D})$ from $\{\mathsf{WPKE.ek}_i\}_{i\in\overline{S}}$, S, and $\mathcal{A}$ as specified in Hyb_3. Then, it runs the extraction algorithm as $\mathsf{WPKE}.\mathcal{E}\mathcal{X}\mathcal{t}(\{\mathsf{WPKE.ek}_i\}_{i\in S}, (U_\mathcal{D}, \rho_\mathcal{D})) \to \{x_i'\}_{i\in S}$. Then, $\mathcal{B}$ submits $\{x_i'\}_{i\in S}$ to its challenger.

We can see that $\mathcal{B}$ perfectly simulates Hyb_8, where a part of the deletion verification algorithm (i.e., inversion of y_i and checking of Equation (7) for $i \in \overline{S}$) is delegated to its challenger. Furthermore, it can be checked that $\mathcal{B}$ wins the cut-and-choose adaptive hardcore bit game if and only if $\mathcal{A}$ wins in Hyb_8. We therefore have $\mathsf{Adv}^{\mathsf{cut\text{-}and\text{-}choose}}_{\mathsf{NTCF},\mathcal{B}}(\lambda, n) = \Pr[\mathsf{E}_8]$. Hence, the lemma follows. □

6.3 Construction with Non-Interactive Key Generation

Here, we discuss that the construction of PKE-SKL with interactive key generation that we showed in Section 6.1 can be readily modified into a construction with non-interactive key generation as per Definition 5.9. The idea is to merging the lessor's operation with the encryption algorithm. For this change to be possible, we need that the encryptor has to know the decryption keys of WPKE. To accomplish this, we postpone the generation of the WPKE encryption keys until the execution of the encryption algorithm.

More concretely, we give the description of the construction in the following. Let $\mathsf{PKESKL}_{\mathsf{int}} = \mathsf{PKESKL}_{\mathsf{int}}.(\mathsf{Setup}, \mathit{IntKeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathcal{D}\mathit{ec}, \mathsf{Del}, \mathsf{DelVrfy})$ be the construction in Section 6.1.

$\mathsf{Setup}_{\mathsf{NI}}(1^\lambda)$**:** It samples $\{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$ as in $\mathsf{PKESKL}_{\mathsf{int}}.\mathsf{Setup}(1^\lambda)$. However, it does *not* sample WPKE encryption keys. It sets $\mathsf{pp} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$, $\mathsf{msk} = \bot$ and $\mathsf{dvk} = \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}$.

$\mathcal{KG}_{\mathsf{NI}}(\mathsf{pp})$**:** Upon receiving the input $\mathsf{pp} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$, it runs the first operation of the lessee in $\mathsf{PKESKL}_{\mathsf{int}}.\mathit{IntKeyGen}$ to obtain $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ and $|\psi'\rangle$. Note that it is possible to run the lessee's algorithm even though encryption keys of WPKE are not generated. It sets $\mathsf{ek} = \{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ and $\mathit{dk} = |\psi'\rangle$.

$\mathsf{Enc}(\mathsf{ek}, m)$**:** It takes as input $\mathsf{ek} = \{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ and proceeds as follows.

- It first samples $\mathsf{WPKE.KG}(1^\lambda) \to (\mathsf{WPKE.ek}_i, \mathsf{WPKE.msk}_i)$ for $i \in [2n]$.
- It then runs the lessor's operation of $\mathsf{PKESKL}_{\mathsf{int}}.\mathit{IntKeyGen}$ to obtain $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$. Note that it is possible to run the lessor's operation since it knows $\mathsf{WPKE.msk}_i$ for all i.
- It then runs $\mathsf{PKESKL}_{\mathsf{int}}.\mathsf{ct} \leftarrow \mathsf{PKESKL}_{\mathsf{int}}.\mathsf{Enc}(\mathsf{PKESKL}_{\mathsf{int}}.\mathsf{ek}, m)$.
- Finally, it outputs the ciphertext $\mathsf{ct} = (\{\mathsf{WPKE.ek}_i, \mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}, \mathsf{PKESKL}_{\mathsf{int}}.\mathsf{ct})$.

$\mathcal{D}ec(\mathcal{dK}, \mathsf{ct})$: It takes as input a ciphertext $\mathsf{ct} = (\{\mathsf{WPKE.ek}_i, \mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}, \mathsf{PKESKL_{int}.ct})$ and the decryption key $\mathcal{dK} = |\psi'\rangle$, it runs the lessee's second operation on $\{\mathsf{WPKE.ek}_i, \mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ and $|\psi'\rangle$ to obtain $\otimes_{i\in[2n]} |\phi_i\rangle$. It then sets $\mathsf{PKESKL_{int}}.\mathcal{dK} = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle \left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$ and runs $\mathsf{PKESKL_{int}}.\mathcal{Dec}(\mathsf{PKESKL_{int}}.\mathcal{dK}, \mathsf{PKESKL_{int}.ct}) = m$. It finally outputs m.

$\mathsf{Del}(\mathcal{dK})$: It measures $\mathcal{dK} = |\psi'\rangle$ in Hadamard basis to obtain $\{d_i, c_i\}_{i\in[2n]}$. It then outputs $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$.

$\mathsf{DelVrfy}(\mathsf{ek}, \mathsf{dvk}, \mathsf{cert})$: It parses $\mathsf{ek} = \{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$. It runs $\mathsf{PKESKL_{int}.DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert})$, where τ is a transcript that starts from $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$,[22] and outputs what it outputs.

The decryption correctness and deletion verification correctness of the scheme readily follows from that of $\mathsf{PKESKL_{int}}$.[23] The following theorem asserts the security of the above construction.

Theorem 6.5. *If* $\mathsf{PKESKL_{int}}$ *is OW-VRA secure, then so is the above construction.*

Proof. We observe that the OW-VRA security game for the above construction is the same as that for $\mathsf{PKESKL_{int}}$, except that the adversary has to submit the deletion certificate in the former before getting $\{\mathsf{WPKE.ek}_i, \mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$. Since the former restricts the adversary more than the latter, the security of $\mathsf{PKESKL_{int}}$ immediately implies that of the above construction. □

Using Lemma 5.10, we can upgrade the construction to be IND-VRA secure.

7 PRF-SKL with Classical Lessor

In this section, we show the construction of PRF-SKL with a classical lessor from the LWE assumption.

7.1 Construction

We construct a UPF-SKL scheme with a classical lessor that satisfies UP-VRA security as per Definition 5.17. Roughly, it requires that if an adversary submits a valid deletion certificate, then it can no longer break unpredictability of the scheme even given the deletion verification key. As shown by [KMY25], UP-VRA security can be easily upgraded into PR-VRA security, its indistinguishability-based counterpart, by the quantum Goldreich-Levin lemma (Lemma 5.8). The construction is very similar to our PKE-SKL construction, where we replace the watermarkable PKE with watermarkable UPF. Concretely, we use the following ingredients:

- NTCF generators $\mathsf{NTCF} = \mathsf{NTCF}.(\mathsf{FuncGen}, \mathcal{StateGen}, \mathsf{Check}, \mathsf{Invert})$ defined as in Section 2.2. It is associated with a deterministic algorithm $\mathsf{NTCF.GoodSet}$ (See Definition 2.10). We assume that $\mathsf{NTCF.Check}$ is a deterministic algorithm. We denote the input space by $\mathcal{X}$ and assume $\mathcal{X} \subseteq \{0,1\}^w$ for some polynomial $w = w(\lambda)$. It is known that NTCF generators can be constructed from LWE.
- Special Dual-mode SFE $\mathsf{SFE} = \mathsf{SFE}.(\mathsf{CRSGen}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$ that supports circuits with input space $\{0,1\}^w$ as defined in Section 4. It is associated with an extraction algorithm $\mathsf{SFE.Ext}$, a simulation algorithm $\mathsf{SFE.Sim}$, a state recovery algorithm $\mathsf{SFE.StaRcv}$, and a "coherent version" $\mathsf{SFE}.\mathcal{Rec}_1$ of $\mathsf{SFE.Rec}_1$, which has the efficient state superposition property. We require that $\mathsf{SFE.Rec}_2$ and $\mathsf{SFE.Ext}$ are deterministic. We denote that the length of each block $\alpha_{i,j}^b$ output by $\mathsf{SFE.StaRcv}$ by u. As we show in Appendix A, this can be constructed from LWE.

[22] While this does not fully specify τ, this is sufficient to run $\mathsf{PKESKL_{int}.DelVrfy}$ since it only uses $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in\overline{S}}$ of τ.

[23] The above scheme does not support the decryption algorithm Dec using msk. However, adding this is straightforward: We have $\mathsf{Setup_{NI}}$ generate an encryption key $\mathsf{PKE.ek}$ and a decryption key $\mathsf{PKE.dk}$ for the plain PKE. We then include $\mathsf{PKE.ek}$ in pp, and $\mathsf{PKE.dk}$ in msk. The encryption algorithm encrypts m using $\mathsf{PKE.ek}$, and incorporates the resulting ciphertext $\mathsf{PKE.ct}$ into ct. Consequently, the decryption algorithm can decrypt $\mathsf{PKE.ct}$ when provided with msk.

- Watermarkable UPF (with parallel extraction): $\mathsf{WUPF} = \mathsf{WUPF}.(\mathsf{KG}, \mathsf{Mark}, \mathsf{Eval})$ as defined in Section 3.2. Without loss of generality, we assume that it also satisfies security as a plain UPF (See Remark 3.3). It is associated with a parallel mark extraction algorithm $\mathsf{WUPF}.\mathcal{Ext}$. We require that $\mathsf{WUPF.Mark}$ and $\mathsf{WUPF.Eval}$ are deterministic algorithms. We need the mark space to be $\{0,1\}^w$ and denote the input space (resp., output space) by $\{0,1\}^\ell$ (resp., $\{0,1\}^{\ell'}$). The bit-length of the marked key is denoted by v. As we show in Section 3.2, this can be constructed from any OWF.

Let $n = n(\lambda)$ be a positive integer such that $n = \omega(\log \lambda)$.
Below is the description of a UP-VRA secure UPF-SKL scheme. The input space of the scheme is $\{0,1\}^{2n\ell}$.

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter, it proceeds as follows.

- For $i \in [2n]$, run $\mathsf{WUPF.KG}(1^\lambda) \to (\mathsf{WUPF.msk}_i, \mathsf{WUPF.xk}_i)$.
- Choose a uniform subset $S \subseteq [2n]$ such that $|S| = n$. In the following, we denote $[2n]\backslash S$ by $\overline{S}$.
- For $i \in [2n]$, run $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, \mathsf{mode})$, where $\mathsf{mode} = \begin{cases} 1 & \text{if } i \in S \\ 2 & \text{if } i \in \overline{S}. \end{cases}$
- Output $\mathsf{pp} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$, $\mathsf{msk} = \{\mathsf{WUPF.msk}_i\}_{i\in[2n]}$, and $\mathsf{dvk} = (S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}})$.

$\mathit{IntKeyGen}(\mathsf{pp}, \mathsf{msk})$**:** The lessor and lessee run the following interactive protocol to generate sk. Here, the lessor takes msk as input and the lessee takes pp as input.

Lessee's first operation: The lessee parses $\mathsf{pp} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$ and runs the following algorithms.

- For each $i \in [2n]$, run $\mathsf{NTCF}.\mathcal{S}\mathit{tateGen}(\mathsf{NTCF.pp}_i) \to (y_i, |\psi_i\rangle)$.
- For each $i \in [2n]$, run $\mathsf{SFE}.\mathcal{R}\mathit{ec}_1(\mathsf{SFE.crs}_i, |\psi_i\rangle) \to (|\psi'_i\rangle, \mathsf{SFE.msg}_i^{(1)})$.
- Send $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ to the lessor.

Lessor's operation: Given the message $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ from the lessee, the lessor proceeds as follows.

- For $i \in [2n]$, define $C[\mathsf{WUPF.msk}_i, y_i]$ to be a circuit that takes x as input and computes the following:

$$C[\mathsf{WUPF.msk}_i, y_i](x) = \begin{cases} \mathsf{WUPF.Mark}(\mathsf{WUPF.msk}_i, x) & \text{if } \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x, y_i) = \top \\ \bot & \text{if } \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x, y_i) = \bot. \end{cases}$$

  Without loss of generality, we assume that the size of $C[\mathsf{WUPF.msk}_i, y_i]$ does not depend on $\mathsf{WUPF.msk}_i$ and y_i, and denote it by $|C|$.
- For $i \in [2n]$, run $\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WUPF.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$.
- Send $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to the lessee.

Lessee's second operation: Given the message $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$, the lessee proceeds as follows. It uses the secret state $|\psi'_i\rangle$ in the following.

- Construct the unitary $U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]$ defined by the following map:

$$|x\rangle\,|\mathsf{SFE.st}\rangle\,|0^v\rangle \mapsto |x\rangle\,|\mathsf{SFE.st}\rangle\,\left|\mathsf{SFE.Rec}_2(\mathsf{SFE.crs}_i, x, \mathsf{SFE.st}, \mathsf{SFE.msg}_i^{(2)})\right\rangle.$$

- For $i \in [2n]$, apply the unitary to compute

$$|\phi_i\rangle \stackrel{\text{def}}{=} U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]\left(|\psi'_i\rangle\,|0^v\rangle\right).$$

Final output: The lessee privately obtains a secret key

$$sk = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle.^{24}$$

We write τ to denote the transcript of the execution of $\mathit{IntKeyGen}$.

$\mathsf{Eval}(\mathsf{msk}, s)$: Upon receiving $\mathsf{msk} = \{\mathsf{WUPF.msk}_i\}_{i\in[2n]}$ and an input $s = s_1\|s_2\|\dots\|s_{2n} \in \{0,1\}^{2n\ell}$, compute $t_i := \mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, s_i)$ for $i \in [2n]$ and output $t = t_1\|\cdots\|t_{2n}$.

$\mathcal{LE}\mathit{val}(sk, s)$**:** Given $sk = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$ and an input $s = s_1\|s_2\|\dots\|s_{2n}$, the evaluation algorithm performs the following for each $i \in [2n]$.

- Construct the unitary $V[s_i]$ defined by the following map:

$$|x\rangle\,|\mathsf{SFE.st}\rangle\,|\mathsf{WUPF.key}\rangle\left|0^{\ell'}\right\rangle \mapsto |x\rangle\,|\mathsf{SFE.st}\rangle\,|\mathsf{WUPF.key}\rangle\,|\mathsf{WUPF.Eval}(\mathsf{WUPF.key}, s_i)\rangle\,.$$

- Compute $V[s_i]\left(|\phi_i\rangle\left|0^{\ell'}\right\rangle\right)$ and measure the last ℓ' qubits to obtain t_i.

It finally outputs $t = t_1\|t_2\|\dots\|t_{2n}$.

$\mathsf{Del}(dk)$**:** Given $dk = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$, the deletion algorithm performs the following for each $i \in [2n]$.

- Measure the corresponding registers to recover the classical strings $\mathsf{SFE.crs}_i$ and $\mathsf{SFE.msg}_i^{(2)}$.
- Compute $U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]^\dagger\,|\phi_i\rangle$ and remove the last v qubits to obtain $\left|\psi_i''\right\rangle$.
- Measure $\left|\psi_i''\right\rangle$ in Hadamard basis to obtain (d_i, c_i), where each $d_i \in \{0,1\}^w$ is obtained by measuring the register corresponding to an element in $\mathcal{X}$ and each $c_i \in \{0,1\}^{uw}$ is obtained by measuring the register corresponding to the SFE state.

The final output of the algorithm is given by $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$,

$\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert})$**:** Given the transcript τ of $\mathit{IntKeyGen}$, the deletion verification key $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$, and the deletion certificate $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$, the deletion verification algorithm works as follows.

- Retrieve $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in\overline{S}}$ from τ.
- For $i \in \overline{S}$, run $\mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i) \to (x_{i,0}, x_{i,1})$. If the inversion fails for any i, output $\bot$.
- For $i \in \overline{S}$, run $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$, where $\alpha_{i,j}^b \in \{0,1\}^u$.
- For $i \in \overline{S}$, compute the vector $d_i^* = (d_i^*[1], \dots, d_i^*[w]) \in \{0,1\}^w$ as

$$d_i^*[j] := \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle,$$

  where c_i is parsed as $c_i = c_{i,1}\|\cdots\|c_{i,w}$, where each block is in $\{0,1\}^u$.
- For $i \in \overline{S}$, check whether

$$\langle d_i \oplus d_i^*, (x_{i,0} \oplus x_{i,1})\rangle = 0 \ \wedge\ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top$$

  holds. If it holds for all $i \in \overline{S}$, it outputs $\top$. Otherwise, it outputs $\bot$.

[24] $\mathsf{SFE.crs}_i$ and $\mathsf{SFE.msg}_i^{(2)}$ are not needed for the evaluation. They are used only for the deletion.

Correctness The following theorem asserts the evaluation correctness and the deletion verification correctness of the above scheme.

Theorem 7.1. *The above construction satisfies the evaluation correctness if* NTCF *satisfies the state generation correctness,* SFE *satisfies evaluation correctness, and* WUPF *satisfies the evaluation correctness. Furthermore, the above construction satisfies the deletion verification correctness if* NTCF *satisfies the state generation correctness,* SFE *satisfies the statistical security against malicious senders, and* SFE *supports state recovery in the hiding mode.*

Proof. We omit the proof of deletion verification correctness, since it is completely the same as that for PKE-SKL shown in Theorem 6.1. We then prove the evaluation correctness. As observed in the proof for Theorem 6.1, all the algorithms run the subsystems indexed by $i \in [2n]$, which run in parallel. Therefore, it suffices to show that each subsystem succeeds in evaluation with overwhelming probability assuming that the lessee follows the protocol honestly. By the same argument as Theorem 6.1, we can represent $\left|\psi_i'\right\rangle$ as $\left|\psi_i'\right\rangle = q_{i,0,z^*}\left|x_{i,0}\right\rangle\left|\mathsf{st}_{i,0,z^*}\right\rangle + q_{i,1,z^*}\left|x_{i,1}\right\rangle\left|\mathsf{st}_{i,1,z^*}\right\rangle$, where $(x_{i,0}, x_{i,1}) = \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$, $z^* = \mathsf{SFE.msg}_i^{(1)}$, and st_{i,b,z^*} is the unique state corresponding to $x_{i,b}$ and $z^* = \mathsf{SFE.msg}_i^{(1)}$. The definition of q_{i,b,z^*} is not relevant here, but it can be found in the proof of Theorem 6.1. By the evaluation correctness of SFE, we have[25]

$$\begin{aligned}|\phi_i\rangle &= \sum_{b\in\{0,1\}} q_{i,b,z^*}\left|x_{i,b}\right\rangle\left|\mathsf{st}_{i,b,z^*}\right\rangle\left|\mathsf{SFE.Rec}_2(\mathsf{SFE.crs}_i, x_{i,b}, \mathsf{st}_{i,b,z^*}, \mathsf{SFE.msg}_i^{(2)})\right\rangle \\ &= \sum_{b\in\{0,1\}} q_{i,b,z^*}\left|x_{i,b}\right\rangle\left|\mathsf{st}_{i,b,z^*}\right\rangle\left|\mathsf{WUPF.key}_i(x_{i,b})\right\rangle.\end{aligned}$$

After applying $V[s_i]$, the above state becomes

$$\begin{aligned}&\sum_{b\in\{0,1\}} q_{i,b,z^*}\left|x_{i,b}\right\rangle\left|\mathsf{st}_{i,b,z^*}\right\rangle\left|\mathsf{WUPF.key}_i(x_{i,b})\right\rangle\left|\mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i(x_{i,b}), s_i)\right\rangle \\ &= \sum_{b\in\{0,1\}} q_{i,b,z^*}\left|x_{i,b}\right\rangle\left|\mathsf{st}_{i,b,z^*}\right\rangle\left|\mathsf{WUPF.key}_i(x_{i,b})\right\rangle\left|\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}, s_i)\right\rangle.\end{aligned}$$

Here, we use the fact that $\mathsf{WUPF.Eval}(\mathsf{WUPF.key}_i(x_{i,b}), s_i) = \mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, s_i)$ holds for *all* x with overwhelming probability over the randomness of WUPF.KG by the evaluation correctness of WUPF. By measuring the last register, one can get $\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}, s_i)$ as desired. □

7.2 Security Proof

Theorem 7.2. *If* NTCF *satisfies cut-and-choose adaptive hardcore bit security,* SFE *satisfies the mode-indistinguishability and the extractability in the extraction mode, and* WUPF *satisfies unpredictability as a plain UPF and parallel mark extractability, then our construction is UP-VRA secure.*

Proof. We prove the theorem by a sequence of hybrids. For the sake of the contradiction, let us assume an adversary $\mathcal{A}$ that wins the UP-VRA security game with non-negligible probability ϵ. In the following, the event that $\mathcal{A}$ wins in $\mathsf{Hyb}_{\mathrm{xx}}$ is denoted by E_{xx}.

Hyb_0**:** This is the original UP-VRA security game between the challenger and an adversary $\mathcal{A}$. Namely, the game proceeds as follows.

1. The challenger computes the first message to the adversary as follows.
   - For $i \in [2n]$, it runs $\mathsf{WUPF.KG}(1^\lambda) \to (\mathsf{WUPF.msk}_i, \mathsf{WUPF.xk}_i)$.
   - It chooses a random subset S of $[2n]$ with size n.

[25] Precisely, the equation does not hold in strict sense, since we ignore negligible difference in trace distance in the proof of Theorem 6.1. However, this can be ignored as this difference only negligibly affects the probability that the verification passes.

- It runs $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, \mathsf{mode})$ for $i \in [2n]$, where $\mathsf{mode} = 1$ if $i \in S$ and $\mathsf{mode} = 2$ if $i \in \overline{S}$.

Finally, it sends $\mathsf{pp} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$ to $\mathcal{A}$. It privately keeps the master secret key $\mathsf{msk} = \{\mathsf{WUPF.msk}_i\}_{i\in[2n]}$ and the deletion verification key $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$.

2. Until $\mathcal{A}$ receives the challenge input at Step 7 of the game, $\mathcal{A}$ has access to the evaluation oracle $\mathsf{Eval}(\mathsf{msk}, \cdot)$, which takes as input $\hat{s} = \hat{s}_1 \| \cdots \| \hat{s}_{2n} \in \{0,1\}^{2n\ell}$ and returns $\hat{t} = \hat{t}_1 \| \cdots \| \hat{t}_{2n}$, where $\hat{t}_i = \mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, \hat{s}_i)$.
3. The adversary $\mathcal{A}$ sends $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ to the challenger.
4. The challenger computes the message to the adversary as follows. It first runs

$$\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WUPF.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$$

for $i \in [2n]$. Then, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.
5. Then, $\mathcal{A}$ submits the deletion certificate $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$.
6. Then, the challenger runs DelVrfy as follows.
   - For $i \in \overline{S}$, it runs $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$.
   - For $i \in \overline{S}$, it computes $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as $d_i^*[j] := \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle$.
   - For $i \in \overline{S}$, it runs $(x_{i,0}, x_{i,1}) \leftarrow \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$. If the inversion fails for any $i \in \overline{S}$, the output of DelVrfy is $\bot$.
   - For $i \in \overline{S}$, it checks whether

$$\langle d_i \oplus d_i^*, x_{i,0} \oplus x_{i,1}\rangle = 0 \ \wedge \ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top \tag{9}$$

   holds using $\mathsf{NTCF.td}_i$. If Equation (9) holds for all $i \in \overline{S}$, the output is $\top$ and otherwise, $\bot$.

   If the output of the algorithm is $\bot$, then it indicates that the output of the game is 0 (i.e., $\mathcal{A}$ failed to win the game). Otherwise, the challenger returns dvk to $\mathcal{A}$ and continues the game.
7. Then, the challenger chooses a random input $s = s_1 \| s_2 \| \cdots \| s_{2n} \leftarrow \{0,1\}^{2n\ell}$ and sends s to $\mathcal{A}$. After receiving the challenge input, $\mathcal{A}$ loses its oracle access to the evaluation oracle.
8. $\mathcal{A}$ then outputs $t' = t'_1 \| \cdots \| t'_{2n}$. The adversary wins the game if $t' = t$.

By the definition, we have $\Pr[\mathsf{E}_0] = \epsilon$.

Hyb_1**:** In this hybrid, the challenger computes $\mathsf{SFE.msg}_i^{(2)}$ differently for $i \in S$ in Step 4 of the game. Namely, it is replaced with the following:

4' The challenger computes the second message to the adversary as follows.
   - It runs $\mathsf{SFE.Ext}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to x_i$ and $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \mathsf{vrd}_i$ for $i \in S$.
   - It computes

$$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \gamma_i) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WUPF.msk}_i, y_i]) & \text{if } i \in \overline{S} \end{cases},$$

   where

$$\gamma_i = \begin{cases} \mathsf{WUPF.Mark}(\mathsf{WUPF.msk}_i, x_i) = \mathsf{WUPF.key}_i(x_i) & \text{if } \mathsf{vrd}_i = \top \\ \bot & \text{if } \mathsf{vrd}_i = \bot. \end{cases}$$

   Finally, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.

Since we have $\gamma_i = C[\mathsf{WUPF.msk}_i, y_i](x_i)$, it follows that this game is indistinguishability from the previous game by the straightforward reduction to the extractability of SFE in the extraction mode. Thus, we have $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\lambda)$. Note that in this hybrid, no marked key of WUPF is necessary for running the game for i such that $i \in S$ and $\mathsf{vrd}_i = \bot$.

Hyb_2**:** In this hybrid, the challenger aborts the game and sets the output of the game to be 0 (i.e., $\mathcal{A}$ loses the game), once it turns out that there is $i \in S$ such that $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \bot$. Namely, we replace Step 4' in the previous hybrid with the following:

4" The challenger computes the second message to the adversary as follows.

- It runs $\mathsf{SFE.Ext}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to x_i$ and $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \mathsf{vrd}_i$ for $i \in S$.
- If $\mathsf{vrd}_i = \bot$ for any $i \in S$, it aborts the game and sets the output of the game to be 0.
- It computes

$$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \mathsf{WUPF.key}_i(x_i)) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WUPF.msk}_i, y_i]) & \text{if } i \in \overline{S}. \end{cases}$$

Finally, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.

We claim $\Pr[\mathsf{E}_2] \geq \Pr[\mathsf{E}_1] - \mathsf{negl}(\lambda)$. To show this, we observe that the only scenario where the adversary $\mathcal{A}$ wins the game in the previous hybrid but not in the current one occurs when $\mathcal{A}$ chooses y_i such that $\mathsf{vrd}_i = \bot$ for some $i \in S$, yet still succeeds in predicting t. However, this implies that $\mathcal{A}$ has predicted the value of $\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, s_i)$ without getting any marked key for the i-th instance of WUPF. Note that the probability that $\mathcal{A}$ has made an evaluation query to $\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, \cdot)$ on input s_i is negligible, since s_i is chosen uniformly at random and $\mathcal{A}$ is not allowed to make evaluation queries once it receives s_i. By a straightforward reduction to the unpredicatability of WUPF, the claim follows.

Hyb_3**:** In this hybrid, we change the winning condition of the adversary. Namely, we replace Step 7 and Step 8 of the game with the following.

7' The challenger parses the adversary $\mathcal{A}$ right before Step 7 of the game as a unitary circuit $U_{\mathcal{A}}$ and a quantum state $\rho_{\mathcal{A}}$. It then constructs a quantum predictor $\mathcal{P}$ for WUPF that takes as input $\{s_i\}_{i\in S}$ as input and outputs $\{t'_i\}_{i\in S}$ from $\mathcal{A}$ as follows.

$\mathcal{P}(\{s_i\}_{i\in S})$**:** It chooses $s_i \leftarrow \{0,1\}^\ell$ for $i \in \overline{S}$ and runs $\mathcal{A}$ on input $\{s_i\}_{i\in[2n]}$ to obtain $\{t'_i\}_{i\in 2n}$. Finally, it outputs $\{t'_i\}_{i\in S}$.

8' The challenger parses $\mathcal{P}$ as a pair of unitary $U_{\mathcal{P}}$ and a quantum state $\rho_{\mathcal{P}}$. Then, it runs the extraction algorithm as $\mathsf{WUPF}.\mathcal{E}\mathsf{xt}(\{\mathsf{WUPF.xk}_i\}_{i\in S}, (U_{\mathcal{P}}, \rho_{\mathcal{P}})) \to \{x'_i\}_{i\in S}$.

9' If $x_i = x'_i$ holds for all $i \in S$, the challenger sets the output of the game to be 1. Otherwise, it sets it to be 0.

As we show in Lemma 7.3, we have $\Pr[\mathsf{E}_3] \geq \Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda)$ by the parallel mark extractability of WUPF.

Hyb_4**:** In this game, we replace Step 9' of the above game with the following:

9" If $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x'_i, y_i) = \top$ holds for all $i \in S$, the challenger sets the output of the game to be 1. Otherwise, it sets it to be 0.

We observe that $x'_i = x_i$ in Step 9' implies $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x'_i, y_i) = \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \top$. To see the latter equality holds, recall that the game is set to abort if $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \bot$ for any $i \in S$ in Step 4". Thus, if the game continues to Step 9', the equality holds. We therefore have $\Pr[\mathsf{E}_4] \geq \Pr[\mathsf{E}_3]$. (Actually, $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_3]$ holds due to the uniqueness of x_i that passes the check. However, showing the inequality suffices for our proof.)

Hyb_5: In this game, we switch Step 4" in the previous hybrid back to Step 4'. The difference from the previous hybrid is that it continues the game even if $\mathsf{vrd}_i = \bot$ for some $i \in S$. Since the adversary had no chance in winning the game in such a case in the previous hybrid, this change only increases the chance of the adversary winning. We therefore have $\Pr[\mathsf{E}_5] \geq \Pr[\mathsf{E}_4]$.

Hyb_6: In this hybrid, we switch Step 4' in the previous hybrid back to Step 4. Namely, it computes $\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WUPF.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$ for all $i \in [2n]$ again. Similarly to the change from Hyb_0 to Hyb_1, the winning probability of $\mathcal{A}$ in this game does not change significantly by the extractability of SFE in the extraction mode. Namely, we have $|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_5]| = \mathsf{negl}(\lambda)$. Note that we do not need $\{\mathsf{SFE.td}_i\}_{i\in S}$ any more to run the game.

Hyb_7: In this hybrid, we change the way $\mathsf{SFE.crs}_i$ is sampled for $i \in S$. Namely, it is sampled as $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, 2)$ for all $i \in [2n]$. Concretely, we replace Step 1 with the following:

1' The challenger computes the first message to the adversary as follows.
- For $i \in [2n]$, it runs $\mathsf{WUPF.KG}(1^\lambda) \to (\mathsf{WUPF.msk}_i, \mathsf{WUPF.xk}_i)$.
- It chooses a random subset S of $[2n]$ with size n.
- It runs $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, 2)$ for $i \in [2n]$, where $\mathsf{mode} = 1$ if $i \in S$ and $\mathsf{mode} = 2$ if $i \in \overline{S}$.

Finally, it sends $\mathsf{pp} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$ to $\mathcal{A}$. It privately keeps the master secret key $\mathsf{msk} = \{\mathsf{WUPF.msk}_i\}_{i\in[2n]}$ and the deletion verification key $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$.

Since we do not need $\mathsf{SFE.td}_i$ for $i \in S$ to run Hyb_6 and Hyb_7, a straightforward reduction to the mode indistinguishability of SFE implies that Hyb_6 and Hyb_7 are computationally indistinguishable. Therefore, we have $|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_7]| = \mathsf{negl}(\lambda)$.

Hyb_8: In this game, we change the way how the challenger runs DelVrfy. In particular, the challenger computes $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j,b}$ and d_i^* for all $i \in [2n]$, instead of only for $i \in \overline{S}$. Still, the inversion of y_i and the check of Equation (9) are performed only for $i \in \overline{S}$. Concretely, we replace Step 6 with the following.

6' Then, the challenger runs (the modified version of) DelVrfy as follows.
- For $i \in [2n]$, it runs $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$.
- For $i \in [2n]$, it computes $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as $d_i^*[j] := \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle$.
- For $i \in \overline{S}$, it runs $(x_{i,0}, x_{i,1}) \leftarrow \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$. If the inversion fails for any $i \in \overline{S}$, the output of DelVrfy is $\bot$.
- For $i \in \overline{S}$, it checks whether

$$e_i = (d_i \oplus d_i^*) \cdot (x_{i,0} \oplus x_{i,1}) \ \wedge\ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top$$

holds using $\mathsf{NTCF.td}_i$. If the above holds for all $i \in \overline{S}$, the output is $\top$ and otherwise, $\bot$.

If the output of the algorithm is $\bot$, then it indicates that the output of the game is 0 (i.e., $\mathcal{A}$ failed to win the game). Otherwise, the challenger returns $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$ to $\mathcal{A}$ and continues the game.

We can see that this change is only conceptual, since the outcome of the verification is unchanged. We may therefore have $\Pr[\mathsf{E}_8] = \Pr[\mathsf{E}_7]$.

Based on the above discussion, it is established that $\Pr[\mathsf{E}_8] \geq \epsilon - \mathsf{negl}(\lambda)$, which is non-negligible by our assumption. However, we can show $\Pr[\mathsf{E}_8] = \mathsf{negl}(\lambda)$ assuming the cut-and-choose adaptive hardcore bit property of NTCF, by a very similar reduction to Lemma 6.4. This results in a contradiction as desired. □

It remains to prove Lemma 7.3.

Lemma 7.3. *If* WUPF *satisfies parallel mark extractability, we have* $\Pr[\mathsf{E}_3] \geq \Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda)$.

Proof. We consider an adversary $\mathcal{B}$ who plays the role of the challenger in Hyb_2 for $\mathcal{A}$, while working as an adversary against the parallel prediction experiment with the number of instances n. It proceeds as follows:

1. At the beginning of the game, the challenger runs $\mathsf{WUPF.KG}(1^\lambda) \to (\mathsf{WUPF.msk}'_j, \mathsf{WUPF.xk}'_j)$ for $j \in [n]$. $\mathcal{B}$ is then given oracle access to $\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}'_j, \cdot)$ for each $j \in [n]$. $\mathcal{B}$ proceeds as follows:
   - $\mathcal{B}$ chooses a random subset $S = \{i_1, \dots, i_n\}$ of $[2n]$ with size n.
   - It implicitly sets $\mathsf{WUPF.msk}_{i_j} = \mathsf{WUPF.msk}'_j$ and $\mathsf{WUPF.xk}_{i_j} = \mathsf{WUPF.xk}'_j$ for $j \in [n]$.
   - It runs $\mathsf{WUPF.KG}(1^\lambda) \to (\mathsf{WUPF.msk}_i, \mathsf{WUPF.xk}_i)$ for $i \in \overline{S}$.
   - It chooses $\{\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i\}_{i\in[2n]}$ and $\{\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i\}_{i\in[2n]}$ as in Hyb_2. Finally, $\mathcal{B}$ sends $\mathsf{pp} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i\}_{i\in[2n]}$ to $\mathcal{A}$.
2. Until $\mathcal{A}$ receives the challenge input from $\mathcal{B}$, $\mathcal{A}$ has access to the evaluation oracle $\mathsf{Eval}(\mathsf{msk}, \cdot)$ throughout the game. To answer the query on input $\hat{s} = \hat{s}_1 \| \cdots \| \hat{s}_{2n} \in \{0,1\}^{2n\ell}$, $\mathcal{B}$ computes $\hat{t}_i = \mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, \hat{s}_i)$ by itself if $i \in \overline{S}$ and by making an evaluation query to the oracle $\mathsf{WUPF.Eval}(\mathsf{WUPF.msk}_i, \cdot)$ on input $\hat{s}_i$ if $i \in S$. Then it returns $\hat{t} = \hat{t}_1 \| \cdots \| \hat{t}_{2n}$ to $\mathcal{A}$.
3. At some point, $\mathcal{A}$ sends $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ to $\mathcal{B}$. $\mathcal{B}$ then chooses its initial message to its challenger by the following steps.
   - It then computes x_j and vrd_j for $j \in S$ using the trapdoors for NTCF and SFE as in Hyb_2. If $\mathsf{vrd}_j = \bot$ for any $j \in S$, it aborts.
   - It then submits $\{x'_1, \dots, x'_n\} \coloneqq \{x_{i_1}, \dots, x_{i_n}\}$ to its challenger.
4. Given $\{\mathsf{WUPF.key}'_j(x'_j)\}_{j\in[n]}$ from the challenger, $\mathcal{B}$ proceeds as follows.
   - It sets $\mathsf{WUPF.key}_{i_j}(x_{i_j}) \coloneqq \mathsf{WUPF.key}'_j(x'_j)$ for $j \in [n]$.
   - It computes $\mathsf{SFE.msg}_i^{(2)}$ as follows:
   $$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \mathsf{WUPF.key}_i(x_i)) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WUPF.msk}_i, y_i]) & \text{if } i \in \overline{S}. \end{cases}$$
   - $\mathcal{B}$ then sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.
   - Then, $\mathcal{A}$ submits the deletion certificate cert to $\mathcal{B}$. $\mathcal{B}$ runs DelVrfy using the trapdoors for NTCF and SFE. If the output is $\bot$, $\mathcal{B}$ also outputs $\bot$ and aborts. Otherwise, $\mathcal{B}$ returns $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$ to $\mathcal{A}$.
5. Then, the challenger sends $\{s'_j\}_{j\in[n]}$ to $\mathcal{B}$.
6. $\mathcal{B}$ computes its output as follows.
   - It sets $s_{i_j} \coloneqq s'_j$ for $j \in [n]$ and samples $s_i \leftarrow \{0,1\}^\ell$ for $i \in \overline{S}$.
   - It then sends $\{s_i\}_{i\in[2n]}$ to $\mathcal{A}$.
   - $\mathcal{A}$ outputs $(t'_1, \dots, t'_{2n})$. It then outputs $(t'_{i_1}, \dots, t'_{i_n})$ as its output.

We can see that $\mathcal{B}$ defined above perfectly simulates Hyb_2 for $\mathcal{A}$ above and $\mathcal{B}$ wins the game whenever $\mathcal{A}$ does. We therefore have $\mathsf{Adv}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{B},n}(\lambda) \geq \Pr[\mathsf{E}_2]$. Furthermore, by the parallel mark extractability of WUPF, we have $\mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WUPF},\mathcal{B},\mathcal{E}\mathcal{X}\mathcal{t},n}(\lambda) \geq \mathsf{Adv}^{\mathsf{par\text{-}pre}}_{\mathsf{WUPF},\mathcal{B},n}(\lambda) - \mathsf{negl}(\lambda)$. We can also see that $\Pr[\mathsf{E}_3] = \mathsf{Adv}^{\mathsf{par\text{-}ext}}_{\mathsf{WUPF},\mathcal{B},\mathcal{E}\mathcal{X}\mathcal{t},n}(\lambda)$ holds, because the way $\mathcal{B}$ predicts the outputs is exactly the same as the way $\mathcal{P}$ does in Hyb_3. By combining these equations, the lemma follows. □

8 DS-SKL with Classical Lessor

In this section, we show the construction of DS-SKL with a classical lessor from the LWE assumption.

8.1 Construction

We construct a DS-SKL scheme with a classical lessor that satisfies RUF-VRA security as per Definition 5.23. The construction is very similar to our PKE-SKL construction, where we replace the watermarkable PKE with watermarkable DS. Concretely, we use the following ingredients:

- NTCF generators $\mathsf{NTCF} = \mathsf{NTCF}.(\mathsf{FuncGen}, \mathcal{StateGen}, \mathsf{Check}, \mathsf{Invert})$ defined as in Section 2.2. It is associated with a deterministic algorithm $\mathsf{NTCF.GoodSet}$ (See Definition 2.10). We assume that $\mathsf{NTCF.Check}$ is a deterministic algorithm. We denote the input space by $\mathcal{X}$ and assume $\mathcal{X} \subseteq \{0,1\}^w$ for some polynomial $w = w(\lambda)$. It is known that NTCF generators can be constructed from LWE.
- Special Dual-mode SFE $\mathsf{SFE} = \mathsf{SFE}.(\mathsf{CRSGen}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$ that supports circuits with input space $\{0,1\}^w$ as defined in Section 4. It is associated with an extraction algorithm $\mathsf{SFE.Ext}$, a simulation algorithm $\mathsf{SFE.Sim}$, a state recovery algorithm $\mathsf{SFE.StaRcv}$, and a "coherent version" $\mathsf{SFE}.\mathcal{Rec}_1$ of $\mathsf{SFE.Rec}_1$, which has the efficient state superposition property. We require that $\mathsf{SFE.Rec}_2$ and $\mathsf{SFE.Ext}$ are deterministic. We denote that the length of each block $\alpha^b_{i,j}$ output by $\mathsf{SFE.StaRcv}$ by u. As we show in Appendix A, this can be constructed from LWE.
- Watermarkable DS (with parallel extraction): $\mathsf{WDS} = \mathsf{WDS}.(\mathsf{KG}, \mathsf{Mark}, \mathsf{Sign}, \mathsf{Vrfy})$ as defined in Section 3.3. Without loss of generality, we assume that this satisfies EUF-CMA security as a plain DS (See Remark 3.9). It is associated with a parallel mark extraction algorithm $\mathsf{WDS}.\mathcal{Ext}$. We require that $\mathsf{WDS.Mark}$ be a deterministic algorithm. This can be assumed without loss of generality because the randomness needed to run $\mathsf{WDS.Mark}$ can be included into $\mathsf{WDS.msk}$. We assume that it has coherent-signability and thus it is associated with a coherent signing quantum algorithm $\mathsf{WDS}.\mathcal{QSign}$. We need the mark space to be $\{0,1\}^w$ and denote the message space by $\{0,1\}^\ell$. The bit length of the marked signing key is denoted by v. As we show in Section 3.3, this can be constructed from the SIS assumption.

Let $n = n(\lambda)$ be a positive integer such that $n = \omega(\log \lambda)$.

Below is the description of the construction.

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter, it proceeds as follows.

- For $i \in [2n]$, run $\mathsf{WDS.KG}(1^\lambda) \to (\mathsf{WDS.vk}_i, \mathsf{WDS.msk}_i, \mathsf{WDS.xk}_i)$.
- Choose a uniform subset $S \subseteq [2n]$ such that $|S| = n$. In the following, we denote $[2n] \backslash S$ by $\overline{S}$.
- For $i \in [2n]$, run $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, \mathsf{mode})$, where $\mathsf{mode} = \begin{cases} 1 & \text{if } i \in S \\ 2 & \text{if } i \in \overline{S}. \end{cases}$
- Output $\mathsf{svk} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WDS.vk}_i\}_{i\in[2n]}$, $\mathsf{msk} = \{\mathsf{WDS.msk}_i\}_{i\in[2n]}$, and $\mathsf{dvk} = (S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}})$.

$\mathit{IntKeyGen}(\mathsf{svk}, \mathsf{msk})$**:** The lessor and lessee run the following interactive protocol to generate sigk. Here, the lessor takes msk as input and the lessee takes svk as input.

Lessee's first operation: The lessee parses $\mathsf{svk} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WDS.vk}_i\}_{i\in[2n]}$ and runs the following algorithms.

- For each $i \in [2n]$, run $\mathsf{NTCF}.\mathcal{StateGen}(\mathsf{NTCF.pp}_i) \to (y_i, |\psi_i\rangle)$.
- For each $i \in [2n]$, run $\mathsf{SFE}.\mathcal{Rec}_1(\mathsf{SFE.crs}_i, |\psi_i\rangle) \to (|\psi'_i\rangle, \mathsf{SFE.msg}^{(1)}_i)$.
- Send $\{y_i, \mathsf{SFE.msg}^{(1)}_i\}_{i\in[2n]}$ to the lessor.

Lessor's operation: Given the message $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ from the lessee, the lessor proceeds as follows.

- For $i \in [2n]$, define $C[\mathsf{WDS.msk}_i, y_i]$ to be a circuit that takes x as input and computes the following:

$$C[\mathsf{WDS.msk}_i, y_i](x) = \begin{cases} \mathsf{WDS.Mark}(\mathsf{WDS.msk}_i, x) & \text{if } \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x, y_i) = \top \\ \bot & \text{if } \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x, y_i) = \bot. \end{cases}$$

  Without loss of generality, we assume that the size of $C[\mathsf{WDS.msk}_i, y_i]$ does not depend on $\mathsf{WDS.msk}_i$ and y_i, and denote it by $|C|$.
- For $i \in [2n]$, run $\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WDS.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$.
- Send $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to the lessee.

Lessee's second operation: Given the message $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$, the lessee proceeds as follows. It uses the secret state $|\psi_i'\rangle$ in the following.

- Construct the unitary $U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]$ defined by the following map:

$$|x\rangle\,|\mathsf{SFE.st}\rangle\,|0^v\rangle \mapsto |x\rangle\,|\mathsf{SFE.st}\rangle\left|\mathsf{SFE.Rec}_2(\mathsf{SFE.crs}_i, x, \mathsf{SFE.st}, \mathsf{SFE.msg}_i^{(2)})\right\rangle.$$

- For $i \in [2n]$, apply the unitary to compute

$$|\phi_i\rangle \stackrel{\text{def}}{=} U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]\left(|\psi_i'\rangle\,|0^v\rangle\right).$$

Final output: The lessee privately obtains a secret key

$$\mathit{sigk} = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle.^{26}$$

We write τ to denote the transcript of the execution of $\mathit{IntKeyGen}$.

$\mathsf{Sign}(\mathsf{msk}, m)$: Upon receiving $\mathsf{msk} = \{\mathsf{WDS.msk}_i\}_{i\in[2n]}$ and an input $m = m_1\|m_2\|\ldots\|m_{2n} \in \{0,1\}^{2n\ell}$, compute $\mathsf{WDS}.\sigma_i \leftarrow \mathsf{WDS.Sign}(\mathsf{WDS.msk}_i, m_i)$ for $i \in [2n]$ and output $\sigma = (\mathsf{WDS}.\sigma_1, \ldots, \mathsf{WDS}.\sigma_{2n})$.

$\mathit{Sign}(\mathit{sigk}, m)$**:** Given $\mathit{sigk} = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$ and an input $m = m_1\|m_2\|\ldots\|m_{2n}$, the signing algorithm performs the following for each $i \in [2n]$. Then, it runs

$$\mathsf{WDS}.\mathcal{QSign}(|\phi_i\rangle, m_i) \to (|\phi_i'\rangle, \mathsf{WDS}.\sigma_i)$$

for $i \in [2n]$. It finally outputs $\mathit{sigk}' = \bigotimes_{i\in[2n]} |\phi_i'\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$ and $\sigma = (\mathsf{WDS}.\sigma_1, \ldots, \mathsf{WDS}.\sigma_{2n})$.

$\mathsf{SigVrfy}(\mathsf{svk}, m, \sigma) \to \top/\bot$**:** Given $\mathsf{svk} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WDS.vk}_i\}_{i\in[2n]}$, $m = m_1\|m_2\|\ldots\|m_{2n}$, and $\sigma = (\mathsf{WDS}.\sigma_1, \ldots, \mathsf{WDS}.\sigma_{2n})$, it outputs $\top$ if $\mathsf{WDS.Vrfy}(\mathsf{WDS.vk}_i, m_i, \mathsf{WDS}.\sigma_i) = \top$ holds for all $i \in [2n]$. Otherwise, it outputs $\bot$.

$\mathsf{Del}(\mathit{sigk})$**:** Given $\mathit{sigk} = \bigotimes_{i\in[2n]} |\phi_i\rangle\,|\mathsf{SFE.crs}_i\rangle\left|\mathsf{SFE.msg}_i^{(2)}\right\rangle$, the deletion algorithm performs the following for each $i \in [2n]$.

- Measure the corresponding registers to recover the classical strings $\mathsf{SFE.crs}_i$ and $\mathsf{SFE.msg}_i^{(2)}$.

[26] $\mathsf{SFE.crs}_i$ and $\mathsf{SFE.msg}_i^{(2)}$ are not needed for signing. They are used only for deletion.

- Compute $U[\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(2)}]^\dagger \left|\phi_i\right\rangle$ and remove the last v qubits to obtain $\left|\psi_i''\right\rangle$.
- Measure $\left|\psi_i''\right\rangle$ in Hadamard basis to obtain (d_i, c_i), where each $d_i \in \{0,1\}^w$ is obtained by measuring the register corresponding to an element in $\mathcal{X}$ and each $c_i \in \{0,1\}^{uw}$ is obtained by measuring the register corresponding to the SFE state.

The final output of the algorithm is given by $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$,

$\mathsf{DelVrfy}(\tau, \mathsf{dvk}, \mathsf{cert})$: Given the transcript τ of $\mathit{IntKeyGen}$, the deletion verification key $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$, and the deletion certificate $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$, the deletion verification algorithm works as follows.

- Retrieve $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in\overline{S}}$ from τ.
- For $i \in \overline{S}$, run $\mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i) \to (x_{i,0}, x_{i,1})$. If the inversion fails for any i, output $\bot$.
- For $i \in \overline{S}$, run $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$, where $\alpha_{i,j}^b \in \{0,1\}^u$.
- For $i \in \overline{S}$, compute the vector $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as

$$d_i^*[j] \coloneqq \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle,$$

  where c_i is parsed as $c_i = c_{i,1} \| \cdots \| c_{i,w}$, where each block is in $\{0,1\}^u$.
- For $i \in \overline{S}$, check whether

$$\langle d_i \oplus d_i^*, (x_{i,0} \oplus x_{i,1})\rangle = 0 \ \wedge\ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top$$

  holds. If it holds for all $i \in \overline{S}$, it outputs $\top$. Otherwise, it outputs $\bot$.

Correctness The following theorem asserts the signature verification correctness, the reusability with static signing keys, and the deletion verification correctness of the above scheme.

Theorem 8.1. *The above construction satisfies the signature verification correctness if* NTCF *satisfies the state generation correctness,* SFE *satisfies evaluation correctness, and* WDS *satisfies the correctness and the coherent signability. Under the same condition, the construction satisfies the reusability with static signing keys. Furthermore, the construction satisfies the deletion verification correctness if* NTCF *satisfies the state generation correctness,* SFE *satisfies the statistical security against malicious senders, and* SFE *supports state recovery in the hiding mode.*

Proof. We omit the proof of deletion verification correctness, since it is completely the same as that for PKE-SKL shown in Theorem 6.1. We therefore show the signature verification correctness and the reusability with static signing keys. It is straightforward to see that a signature generated by msk passes the verification. We therefore consider the case where a signature is generated by sigk. As observed in the proof for Theorem 6.1, all the algorithms run the subsystems indexed by $i \in [2n]$, which run in parallel. Therefore, it suffices to show that each $\mathsf{WDS}.\sigma_i$ passes the verification with overwhelming probability and generation of $\mathsf{WDS}.\sigma_i$ almost does not change the state $\left|\phi_i\right\rangle$, assuming that the lessee follows the protocol honestly.

By the same argument as Theorem 6.1, we can represent $\left|\psi_i'\right\rangle$ as $\left|\psi_i'\right\rangle = q_{i,0,z^*} \left|x_{i,0}\right\rangle \left|\mathsf{st}_{i,0,z^*}\right\rangle + q_{i,1,z^*} \left|x_{i,1}\right\rangle \left|\mathsf{st}_{i,1,z^*}\right\rangle$, where $(x_{i,0}, x_{i,1}) = \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$, where $z^* = \mathsf{SFE.msg}_i^{(1)}$, and st_{i,b,z^*} is the unique state corresponding to $x_{i,b}$ and $z^* = \mathsf{SFE.msg}_i^{(1)}$. The definition of q_{i,b,z^*} is not relevant here, but it can be found in the proof of Theorem 6.1. By the evaluation correctness of SFE, we have[27]

$$\begin{aligned}\left|\phi_i\right\rangle &= \sum_{b\in\{0,1\}} q_{i,b,z^*} \left|x_{i,b}\right\rangle \left|\mathsf{st}_{i,b,z^*}\right\rangle \left|\mathsf{SFE.Rec}_2(\mathsf{SFE.crs}_i, x_{i,b}, \mathsf{st}_{i,b,z^*}, \mathsf{SFE.msg}_i^{(2)})\right\rangle \\ &= \sum_{b\in\{0,1\}} q_{i,b,z^*} \left|x_{i,b}\right\rangle \left|\mathsf{st}_{i,b,z^*}\right\rangle \left|\mathsf{WDS.sigk}_i(x_{i,b})\right\rangle.\end{aligned}$$

[27] Precisely, the equation does not hold in the strict sense, since we ignore negligible difference in trace distance in the proof of Theorem 6.1. However, this can be ignored as this difference only negligibly affects the probability that the verification passes.

Then, $\mathsf{WDS}.\mathcal{QSign}$ is applied on $|\phi_i\rangle$ and m_i. Let us consider $L = w + uw$ and consider $\{\alpha_t \in \mathbb{C}\}_{t\in\{0,1\}^L}$ and $\{x'_t \in \{0,1\}^w\}_{t\in\{0,1\}^L}$ indexed by t such that

$$\alpha_t = \begin{cases} q_{i,b,z^*} & \text{if } t = x_{i,b}\|\mathsf{st}_{i,b,z^*} \text{ for some } b \in \{0,1\} \\ 0 & \text{otherwise} \end{cases}, \quad x'_t = \begin{cases} x_{i,b} & \text{if } t = x_{i,b}\|\mathsf{st}_{i,b,z^*} \text{ for some } b \in \{0,1\} \\ x^* & \text{otherwise} \end{cases}$$

where x^* is an arbitrary string. Then, we can write $|\phi_i\rangle$ as $|\phi_i\rangle = \sum_{t\in\{0,1\}^L} \alpha_t |t\rangle |\mathsf{WDS.sigk}_i(x'_t)\rangle$. By the first property of coherent signability satisfied by WDS, $\mathsf{WDS}.\sigma_i$ is in $\cup_t \mathsf{Supp}(\mathsf{WDS.Sign}(\mathsf{WDS.sigk}_i(x'_t), m_i))$. By the correctness of WDS, $\mathsf{WDS}.\sigma_i$ passes the verification. Furthermore, by the second property of coherent signability satisfied by WDS, $|\phi'_i\rangle$ obtained by applying $\mathsf{WDS}.\mathcal{QSign}$ is negligibly close to $|\phi_i\rangle$ in trace distance. □

8.2 Security Proof

The following theorem asserts the security of the above construction.

Theorem 8.2. *If* NTCF *satisfies cut-and-choose adaptive hardcore bit security,* SFE *satisfies the mode-indistinguishability and the extractability in the extraction mode, and* WDS *satisfies parallel mark extractability and EUF-CMA security as a plain DS, then our construction is RUF-VRA secure.*

Proof. We prove the theorem by a sequence of hybrids. For the sake of the contradiction, let us assume an adversary $\mathcal{A}$ that wins the RUF-VRA security game with non-negligible probability ϵ. In the following, the event that $\mathcal{A}$ wins in $\mathsf{Hyb}_{\mathsf{xx}}$ is denoted by E_{xx}.

Hyb_0**:** This is the original RUF-VRA security game between the challenger and an adversary $\mathcal{A}$. Namely, the game proceeds as follows.

1. The challenger computes the first message to the adversary as follows.
   - For $i \in [2n]$, it runs $\mathsf{WDS.KG}(1^\lambda) \to (\mathsf{WDS.vk}_i, \mathsf{WDS.msk}_i, \mathsf{WDS.xk}_i)$.
   - It chooses random subset S of $[2n]$ with size n.
   - It runs $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, \mathsf{mode})$ for $i \in [2n]$, where $\mathsf{mode} = 1$ if $i \in S$ and $\mathsf{mode} = 2$ if $i \in \overline{S}$.

   Finally, it sends $\mathsf{svk} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WDS.vk}_i\}_{i\in[2n]}$ to $\mathcal{A}$. It privately keeps the master secret key $\mathsf{msk} = \{\mathsf{WDS.msk}_i\}_{i\in[2n]}$ and the deletion verification key $\mathsf{dvk} = \big(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\big)$.
2. Until $\mathcal{A}$ receives the challenge input at Step 7 of the game, $\mathcal{A}$ has an access to the signing oracle $\mathsf{Sign}(\mathsf{msk}, \cdot)$, which takes as input $\hat{m} = \hat{m}_1\| \cdots \|\hat{m}_{2n} \in \{0,1\}^{2n\ell}$ and returns $\hat{\sigma} = \mathsf{WDS}.\hat{\sigma}_1\| \cdots \|\mathsf{WDS}.\hat{\sigma}_{2n}$, where $\mathsf{WDS}.\hat{\sigma}_i \leftarrow \mathsf{WDS.Sign}(\mathsf{WDS.msk}_i, \hat{m}_i)$.
3. The adversary $\mathcal{A}$ sends $\{y_i, \mathsf{SFE.msg}_i^{(1)}\}_{i\in[2n]}$ to the challenger.
4. The challenger computes the message to the adversary as follows. It first runs

$$\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WDS.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$$

   for $i \in [2n]$. Then, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i\in[2n]}$ to $\mathcal{A}$.
5. Then, $\mathcal{A}$ submits the deletion certificate $\mathsf{cert} = \{d_i, c_i\}_{i\in[2n]}$.
6. Then, the challenger runs DelVrfy as follows.
   - For $i \in \overline{S}$, it runs $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$.
   - For $i \in \overline{S}$, it computes $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as $d_i^*[j] := \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle$.
   - For $i \in \overline{S}$, it runs $(x_{i,0}, x_{i,1}) \leftarrow \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$. If the inversion fails for any $i \in \overline{S}$, the output of DelVrfy is $\perp$.

- For $i \in \overline{S}$, it checks whether

$$(d_i \oplus d_i^*) \cdot (x_{i,0} \oplus x_{i,1}) = 0 \ \wedge \ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top \tag{10}$$

holds using $\mathsf{NTCF.td}_i$. If Equation (10) holds for all $i \in \overline{S}$, the output is $\top$ and otherwise, $\bot$.

If the output of the algorithm is $\bot$, then it indicates that the output of the game is 0 (i.e., $\mathcal{A}$ failed to win the game). Otherwise, the challenger returns dvk to $\mathcal{A}$ and continues the game.

7. Then, the challenger chooses a random input $m = m_1 \| m_2 \| \cdots \| m_{2n} \leftarrow \{0,1\}^{2n\ell}$ and sends m to $\mathcal{A}$. After receiving the challenge input, $\mathcal{A}$ loses its oracle access to the signing oracle.
8. $\mathcal{A}$ then outputs $\sigma' = \mathsf{WDS}.\sigma'_1 \| \cdots \| \mathsf{WDS}.\sigma'_{2n}$. The challenger sets the output of the game to be 1 (i.e., indicating that $\mathcal{A}$ wins the game) if $\mathsf{WDS.Vrfy}(\mathsf{WDS.vk}_i, m_i, \mathsf{WDS}.\sigma'_i) = \top$ holds for all $i \in [2n]$. Otherwise, it outputs $\bot$.

By the definition, we have $\Pr[\mathsf{E}_0] = \epsilon$.

Hyb_1: In this hybrid, the challenger computes $\mathsf{SFE.msg}_i^{(2)}$ differently for $i \in S$ in Step 4 of the game. Namely, it is replaced with the following:

4' The challenger computes the second message to the adversary as follows.

- It runs $\mathsf{SFE.Ext}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to x_i$ and $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \mathsf{vrd}_i$ for $i \in S$.
- It computes

$$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \gamma_i) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WDS.msk}_i, y_i]) & \text{if } i \in \overline{S} \end{cases},$$

where

$$\gamma_i = \begin{cases} \mathsf{WDS.Mark}(\mathsf{WDS.msk}_i, x_i) = \mathsf{WDS.sigk}_i(x_i) & \text{if } \mathsf{vrd}_i = \top \\ \bot & \text{if } \mathsf{vrd}_i = \bot. \end{cases}$$

Finally, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i \in [2n]}$ to $\mathcal{A}$.

Since we have $\gamma_i = C[\mathsf{WDS.msk}_i, y_i](x_i)$, it follows that this game is indistinguishable from the previous game by the straightforward reduction to the extractability of SFE in the extraction mode. Thus, we have $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\lambda)$. Note that the simulation of this game does not require any marked signing key for the i-th instance of WDS if $i \in S$ and $\mathsf{vrd}_i = \bot$.

Hyb_2: In this hybrid, the challenger aborts the game and sets the output of the game to be 0 (i.e., $\mathcal{A}$ loses the game), once it turns out that there is $i \in S$ such that $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \bot$. Namely, we replace Step 4' in the previous hybrid with the following:

4" The challenger computes the second message to the adversary as follows.

- It runs $\mathsf{SFE.Ext}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to x_i$ and $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \mathsf{vrd}_i$ for $i \in S$.
- If $\mathsf{vrd}_i = \bot$ for any $i \in S$, it aborts the game and sets the output of the game to be 0.
- It computes

$$\mathsf{SFE.msg}_i^{(2)} \leftarrow \begin{cases} \mathsf{SFE.Sim}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, 1^{|C|}, \mathsf{WDS.sigk}_i(x_i)) & \text{if } i \in S \\ \mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WDS.msk}_i, y_i]) & \text{if } i \in \overline{S}. \end{cases}$$

Finally, it sends $\{\mathsf{SFE.msg}_i^{(2)}\}_{i \in [2n]}$ to $\mathcal{A}$.

We claim $\Pr[\mathsf{E}_2] \geq \Pr[\mathsf{E}_1] - \mathsf{negl}(\lambda)$. To show this, we observe that the only scenario where the adversary $\mathcal{A}$ wins the game in the previous hybrid but not in the current one occurs when $\mathcal{A}$ chooses y_i such that $\mathsf{vrd}_i = \bot$ for some $i \in S$, yet still succeeds in signing on $\{m_j\}_{j\in[2n]}$. However, this implies that $\mathcal{A}$ has successfully forged a signature σ_i satisfying $\mathsf{WDS.Vrfy}(\mathsf{WDS.vk}_i, m_i, \sigma_i)$ without getting marked signing key for the i-th instance of WDS. Note that the probability that $\mathcal{A}$ has made a signing query to $\mathsf{WDS.Sign}(\mathsf{WDS.msk}_i, \cdot)$ on message m_i is negligible, since m_i is chosen uniformly at random and $\mathcal{A}$ is not allowed to make signing queries once it receives m_i. By a straightforward reduction to the EUF-CMA security of WDS, the claim follows.

Hyb_3**:** In this hybrid, we change the winning condition of the adversary. Namely, we replace Step 7 and Step 8 of the game with the following.

7' The challenger parses the adversary $\mathcal{A}$ right before Step 7 of the game as a unitary circuit $U_{\mathcal{A}}$ and a quantum state $\rho_{\mathcal{A}}$. It then constructs a quantum forger $\mathcal{F}$ for WDS that takes $\{m_i\}_{i\in S}$ as input and outputs $\{\mathsf{WDS}.\sigma'_i\}_{i\in S}$ from $\mathcal{A}$ as follows.

$\mathcal{F}(\{m_i\}_{i\in S})$**:** It chooses $m_i \leftarrow \{0,1\}^\ell$ for $i \in \overline{S}$ and runs $\mathcal{A}$ on input $\{m_i\}_{i\in[2n]}$ to obtain $\{\mathsf{WDS}.\sigma'_i\}_{i\in[2n]}$. Finally, it outputs $\{\mathsf{WDS}.\sigma'_i\}_{i\in S}$.

8' The challenger parses $\mathcal{F}$ as a pair of unitary $U_{\mathcal{F}}$ and a quantum state $\rho_{\mathcal{F}}$. Then, it runs the extraction algorithm as $\mathsf{WDS}.\mathcal{E}\mathcal{X}\mathsf{t}(\{\mathsf{WDS.xk}_i\}_{i\in S}, (U_{\mathcal{F}}, \rho_{\mathcal{F}})) \to \{x'_i\}_{i\in S}$.

9' If $x_i = x'_i$ holds for all $i \in S$, the challenger sets the output of the game to be 1. Otherwise, it sets it to be 0.

By the parallel extractability of WDS, we have $\Pr[\mathsf{E}_3] \geq \Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda)$. The proof is almost the same as Lemma 7.3 and thus omitted.

Hyb_4**:** In this game, we replace Step 9' of the above game with the following:

9" If $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x'_i, y_i) = \top$ holds for all $i \in S$, the challenger sets the output of the game to be 1. Otherwise, it sets it to be 0.

We observe that $x'_i = x_i$ in Step 9' implies $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x'_i, y_i) = \mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \top$. To see the latter equality holds, recall that the game is set to abort if $\mathsf{NTCF.Check}(\mathsf{NTCF.pp}_i, x_i, y_i) = \bot$ for any $i \in S$ in Step 4". Thus, if the game continues to Step 9', the equality holds. We therefore have $\Pr[\mathsf{E}_4] \geq \Pr[\mathsf{E}_3]$. (Actually, $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_3]$ holds due to the uniqueness of x_i that passes the check. However, showing the inequality suffices for our proof.)

Hyb_5**:** In this game, we switch Step 4" in the previous hybrid back to Step 4'. The difference from the previous hybrid is that it continues the game even if $\mathsf{vrd}_i = \bot$ for some $i \in S$. Since the adversary had no chance in winning the game in such a case in the previous hybrid, this change only increases the chance of the adversary winning. We therefore have $\Pr[\mathsf{E}_5] \geq \Pr[\mathsf{E}_4]$.

Hyb_6**:** In this hybrid, we switch Step 4' in the previous hybrid back to Step 4. Namely, it computes $\mathsf{SFE.Send}(\mathsf{SFE.crs}_i, \mathsf{SFE.msg}_i^{(1)}, C[\mathsf{WDS.msk}_i, y_i]) \to \mathsf{SFE.msg}_i^{(2)}$ for all $i \in [2n]$ again. Similarly to the change from Hyb_0 to Hyb_1, the winning probability of $\mathcal{A}$ in this game does not change significantly by the extractability of SFE in the extraction mode. Namely, we have $|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_5]| = \mathsf{negl}(\lambda)$. Note that we do not need $\{\mathsf{SFE.td}_i\}_{i\in S}$ any more to run the game.

Hyb_7**:** In this hybrid, we change the way $\mathsf{SFE.crs}_i$ is sampled for $i \in S$. Namely, it is sampled as $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, 2)$ for all $i \in [2n]$. Concretely, we replace Step 1 with the following:

1' The challenger computes the first message to the adversary as follows.

- For $i \in [2n]$, it runs $\mathsf{WDS.KG}(1^\lambda) \to (\mathsf{WDS.vk}_i, \mathsf{WDS.msk}_i, \mathsf{WDS.xk}_i)$.
- It chooses random subset S of $[2n]$ with size n.
- It runs $(\mathsf{NTCF.pp}_i, \mathsf{NTCF.td}_i) \leftarrow \mathsf{NTCF.FuncGen}(1^\lambda, \mathsf{mode})$ and $(\mathsf{SFE.crs}_i, \mathsf{SFE.td}_i) \leftarrow \mathsf{SFE.CRSGen}(1^\lambda, 2)$ for $i \in [2n]$, where $\mathsf{mode} = 1$ if $i \in S$ and $\mathsf{mode} = 2$ if $i \in \overline{S}$.

Finally, it sends $\mathsf{svk} = \{\mathsf{NTCF.pp}_i, \mathsf{SFE.crs}_i, \mathsf{WDS.vk}_i\}_{i\in[2n]}$ to $\mathcal{A}$. It privately keeps the master secret key $\mathsf{msk} = \{\mathsf{WDS.msk}_i\}_{i\in[2n]}$ and the deletion verification key $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$.

Since we do not need $\mathsf{SFE.td}_i$ for $i \in S$ to run Hyb_6 and Hyb_7, a straightforward reduction to the mode indistinguishability of SFE implies that Hyb_6 and Hyb_7 are computationally indistinguishable. Therefore, we have $|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_7]| = \mathsf{negl}(\lambda)$.

Hyb_8**:** In this game, we change the way how the challenger runs DelVrfy. In particular, the challenger computes $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j,b}$ and d_i^* for all $i \in [2n]$, instead of only for $i \in \overline{S}$. Still, the inversion of y_i and the check of Equation (10) are performed only for $i \in \overline{S}$. Concretely, we replace Step 6 with the following.

6' Then, the challenger runs (the modified version of) DelVrfy as follows.

- For $i \in [2n]$, it runs $\mathsf{SFE.StaRcv}(\mathsf{SFE.td}_i, \mathsf{SFE.msg}_i^{(1)}) \to \{\alpha_{i,j}^b\}_{j\in[w],b\in\{0,1\}}$.
- For $i \in [2n]$, it computes $d_i^* = (d_i^*[1], \ldots, d_i^*[w]) \in \{0,1\}^w$ as $d_i^*[j] := \langle c_{i,j}, (\alpha_{i,j}^0 \oplus \alpha_{i,j}^1)\rangle$.
- For $i \in \overline{S}$, it runs $(x_{i,0}, x_{i,1}) \leftarrow \mathsf{NTCF.Invert}(\mathsf{NTCF.td}_i, y_i)$. If the inversion fails for any $i \in \overline{S}$, the output of DelVrfy is $\bot$.
- For $i \in \overline{S}$, it checks whether

$$(d_i \oplus d_i^*) \cdot (x_{i,0} \oplus x_{i,1}) = 0 \ \wedge \ \mathsf{NTCF.GoodSet}(\mathsf{NTCF.td}_i, y_i, d_i \oplus d_i^*) = \top$$

holds using $\mathsf{NTCF.td}_i$. If the above holds for all $i \in \overline{S}$, the output is $\top$ and otherwise, $\bot$.

If the output of the algorithm is $\bot$, then it indicates that the output of the game is 0 (i.e., $\mathcal{A}$ failed to win the game). Otherwise, the challenger returns $\mathsf{dvk} = \left(S, \{\mathsf{NTCF.td}_i, \mathsf{SFE.td}_i\}_{i\in\overline{S}}\right)$ to $\mathcal{A}$ and continues the game.

We can see that this change is only conceptual, since the outcome of the verification is unchanged. We therefore have $\Pr[\mathsf{E}_8] = \Pr[\mathsf{E}_7]$.

Based on the above discussion, it is established that $\Pr[\mathsf{E}_8] \geq \epsilon - \mathsf{negl}(\lambda)$, which is non-negligible by our assumption. However, we can show $\Pr[\mathsf{E}_8] = \mathsf{negl}(\lambda)$ assuming the cut-and-choose adaptive hardcore bit property of NTCF, by a very similar reduction to Lemma 6.4. This results in a contradiction as desired. □

Acknowledgment. Generative AI tools were employed to improve the readability and presentation of this manuscript. We thank Duo Xu for answering our questions about [TX25]. We thank anonymous reviewers of CRYPTO 2026 and TQC 2026 for their helpful comments. Shota Yamada is supported by JPMJCR22M1 and JPMJKP24U3.

References

[AC02] Mark Adcock and Richard Cleve. A quantum goldreich-levin theorem with cryptographic applications. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002. (Cited on page 41, 43.)

[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, 13.)

[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, 6, 40, 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.)

[AMR22] Navid Alamati, Giulio Malavolta, and Ahmadreza Rahimi. Candidate trapdoor claw-free functions from group actions with applications to quantum protocols. In *Theory of Cryptography Conference*, pages 266–293. Springer, 2022. (Cited on page 4.)

[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, 13, 40.)

[BB20] Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *arXiv preprint arXiv:2003.06557*, 2020. (Cited on page 6.)

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th FOCS*, pages 320–331. IEEE Computer Society Press, October 2018. (Cited on page 8, 18.)

[BCM+21] Zvika Brakerski, Paul F. Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. *J. ACM*, 68(5):31:1–31:47, 2021. (Cited on page 8, 76, 79.)

[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 4, 5.)

[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 5.)

[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 5.)

[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 5.)

[BSS21] Amit Behera, Or Sattath, and Uriel Shinar. Noise-tolerant quantum tokens for MAC. Cryptology ePrint Archive, Report 2021/1353, 2021. (Cited on page 7, 11.)

[CGJL25] Orestis Chardouvelis, Vipul Goyal, Aayush Jain, and Jiahui Liu. Quantum key leasing for pke and fhe with a classical lessor. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 248–277. Springer, 2025. (Cited on page 3, 4, 5, 8, 10, 14, 15.)

[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 41, 43.)

[DDN14] Bernardo David, Rafael Dowsley, and Anderson CA Nascimento. Universally composable oblivious transfer based on a variant of lpn. In *International Conference on Cryptology and Network Security*, pages 143–158. Springer, 2014. (Cited on page 4.)

[DVDGMQN08] Rafael Dowsley, Jeroen Van De Graaf, Jörn Müller-Quade, and Anderson CA Nascimento. Oblivious transfer based on the mceliece assumptions. In *International Conference on Information Theoretic Security*, pages 107–117. Springer, 2008. (Cited on page 4.)

[GMP23] Alexandru Gheorghiu, Tony Metger, and Alexander Poremba. Quantum cryptography with classical communication: Parallel remote state preparation for copy-protection, verification, and more. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *ICALP 2023*, volume 261 of *LIPIcs*, pages 67:1–67:17. Schloss Dagstuhl, July 2023. (Cited on page 7.)

[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 76.)

[GR02] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. arXiv, quant-ph/0208112, 2002. (Cited on page 79.)

[GV19] Alexandru Gheorghiu and Thomas Vidick. Computationally-secure and composable remote state preparation. In David Zuckerman, editor, *60th FOCS*, pages 1024–1033. IEEE Computer Society Press, November 2019. (Cited on page 7.)

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015. (Cited on page 17.)

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Berlin, Heidelberg, August 2016. (Cited on page 13, 16, 23.)

[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 5.)

[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 5, 11, 19.)

[JQSY19] Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun. General linear group action on tensors: A candidate for post-quantum cryptography. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 251–281. Springer, Cham, December 2019. (Cited on page 5.)

[KMY25] Fuyuki Kitagawa, Tomoyuki Morimae, and Takashi Yamakawa. A simple framework for secure key leasing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 217–247. Springer, 2025. (Cited on page 3, 4, 5, 6, 7, 11, 12, 13, 14, 16, 17, 20, 23, 26, 30, 32, 33, 34, 39, 40, 43, 45, 56.)

[KNP25] Fuyuki Kitagawa, Ryo Nishimaki, and Nikhil Pappu. Pke and abe with collusion-resistant secure key leasing. In *Annual International Cryptology Conference*, pages 35–68. Springer, 2025. (Cited on page 3, 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 5.)

[Mah18] Urmila Mahadev. Classical verification of quantum computations. In Mikkel Thorup, editor, *59th FOCS*, pages 259–267. IEEE Computer Society Press, October 2018. (Cited on page 8, 18, 19.)

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. (Cited on page 76.)

[MPY24] Tomoyuki Morimae, Alexander Poremba, and Takashi Yamakawa. Revocable quantum digital signatures. In Frédéric Magniez and Alex Bredariol Grilo, editors, *19th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2024, September 9-13, 2024, Okinawa, Japan*, volume 310 of *LIPIcs*, pages 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. (Cited on page 5, 6, 13, 45.)

[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 5.)

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, August 2008. (Cited on page 14, 39, 74, 75, 76.)

[PWYZ24] Duong Hieu Phan, Weiqiang Wen, Xingyu Yan, and Jinwei Zheng. Adaptive hardcore bit and quantum key leasing over classical channel from LWE with polynomial modulus. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IX*, volume 15492 of *LNCS*, pages 185–214. Springer, Singapore, December 2024. (Cited on page 4, 5, 14, 15.)

[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 79.)

[Tsa17] Rotem Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 489–518. Springer, Cham, November 2017. (Cited on page 17.)

[TX25] Yuki Takeuchi and Duo Xu. Computational certified deletion property of magic square game and its application to classical secure key leasing, 2025. (Cited on page 5, 70.)

[Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999. (Cited on page 40, 42.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 14, 80.)

[Zha22] Jiayu Zhang. Classical verification of quantum computations in linear time. In *63rd FOCS*, pages 46–57. IEEE Computer Society Press, October / November 2022. (Cited on page 7.)

[Zha25] Jiayu Zhang. Formulations and constructions of remote state preparation with verifiability, with applications. In Raghu Meka, editor, *ITCS 2025*, volume 325, pages 96:1–96:19. LIPIcs, January 2025. (Cited on page 7.)

A Special Dual-Mode SFE from LWE

In this section, we show a construction of special dual-mode SFE from the LWE assumption.

In general, by relying on well-known Yao's protocol, SFE can be constructed from oblivious transfer (OT) by additionally using garbled circuits which can be based on OWFs. We can prove that the resulting scheme becomes special dual-mode SFE if the underlying OT satisfies an analogous special dual-mode property. In this section, we first formally define special dual-mode OT and provide its realization assuming the LWE assumption based on the construction by [PVW08]. Then, we provide a sketch on how to achieve dual-mode SFE from dual-mode OT using GC.

A.1 Definition of Special Dual-mode OT

In this subsection, we define special dual-mode OT.

Definition A.1 (Special Dual-mode OT). *A special dual-mode OT scheme* OT *is a tuple of four algorithms* $\mathsf{OT} = \mathsf{OT}.(\mathsf{CRSGen}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$. *Below, let* $\{0,1\}^\ell$ *be the string space of* OT.

$\mathsf{CRSGen}(1^\lambda, \mathsf{mode}) \to (\mathsf{crs}, \mathsf{td})$**:** *The CRS generation algorithm is a PPT algorithm that takes a security parameter* 1^λ *and* $\mathsf{mode} \in \{1,2\}$*, and outputs strings* crs *and* td. $\mathsf{mode} = 1$ *indicates that the system in the "extractable mode", while* $\mathsf{mode} = 2$ *indicates "hiding mode".*

$\mathsf{Rec}_1(\mathsf{crs}, b) \to (\mathsf{msg}^{(1)}, \mathsf{st})$**:** *This is a classical PPT algorithm supposed to be run by a receiver to generate the first message of the protocol. It takes as input the CRS* crs *and a bit* $b \in \{0,1\}$ *and outputs the first message* $\mathsf{msg}^{(1)}$ *and a secret state* st.

$\mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, (z_0, z_1)) \to \mathsf{msg}^{(2)}$**:** *This is a classical PPT algorithm supposed to be run by a sender to generate the second message of the protocol. It takes as input the CRS* crs*, a message* $\mathsf{msg}^{(1)}$ *from the receiver, and a pair of strings* $(z_0, z_1) \in \{0,1\}^{\ell \times 2}$ *and outputs the second message* $\mathsf{msg}^{(2)}$.

$\mathsf{Rec}_2(\mathsf{crs}, b, \mathsf{st}, \mathsf{msg}^{(2)}) \to z$**:** *This is a classical PPT algorithm supposed to be run by a receiver to derive the output. It takes as input the CRS* crs*, a bit* $b \in \{0,1\}$*, a state* st*, and a message* $\mathsf{msg}^{(2)}$ *from the sender, and outputs a string* z.

We require dual-mode OT to satisfy the following properties.

Evaluation correctness: *For all* $b \in \{0,1\}$, $\mathsf{mode} \in \{1,2\}$*, and a pair of strings* $(z_0, z_1) \in \{0,1\}^{\ell \times 2}$*, we have*

$$\Pr\left[\mathsf{Rec}_2(\mathsf{crs}, b, \mathsf{st}, \mathsf{msg}^{(2)}) = z_b \;\middle|\; \begin{array}{l}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda, \mathsf{mode}) \\ (\mathsf{msg}^{(1)}, \mathsf{st}) \leftarrow \mathsf{Rec}_1(\mathsf{crs}, b) \\ \mathsf{msg}^{(2)} \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, (z_0, z_1))\end{array}\right] = 1.$$

Unique state: *For all* $\mathsf{mode} \in \{1,2\}$, $(\mathsf{crs}, \mathsf{td}) \in \mathsf{Supp}(\mathsf{CRSGen}(1^\lambda, \mathsf{mode}))$, $b \in \{0,1\}$, *and* $(\mathsf{msg}^{(1)}, \mathsf{st}) \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}, b))$, st *is the unique state that corresponds to* crs*,* b*, and* $\mathsf{msg}^{(1)}$*, that is, there is no* $\mathsf{st}' \neq \mathsf{st}$ *such that* $(\mathsf{msg}^{(1)}, \mathsf{st}') \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}, b))$. *We denote the unique state* st *such that* $(\mathsf{msg}^{(1)}, \mathsf{st}) \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}, b))$ *by* $\mathsf{st}_{b \to \mathsf{msg}^{(1)}}$.[28]

Mode indistinguishability: *For all QPT algorithm* $\mathcal{A}$*, we have*

$$\left|\Pr\left[\mathcal{A}(\mathsf{crs}) = 1 : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda, 1)\right] - \Pr\left[\mathcal{A}(\mathsf{crs}) = 1 : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda, 2)\right]\right| \leq \mathsf{negl}(\lambda).$$

Statistical security against malicious senders in the hiding mode: *For all* crs *output by* $\mathsf{CRSGen}(1^\lambda, 2)$*, we require the following statistical indistinguishability:*

$$\mathsf{msg}_0^{(1)} \approx_s \mathsf{msg}_1^{(1)} \quad \textit{where} \quad (\mathsf{msg}_b^{(1)}, \mathsf{st}) \leftarrow \mathsf{Rec}_1(\mathsf{crs}, b) \quad \textit{for} \quad b \in \{0,1\}.$$

[28] The state also depend on crs, but we omit it from the notation.

State recoverability in the hiding mode: *There exists a classical PPT algorithm* StaRcv *that takes as input the trapdoor* td *and the first message of the receiver* $\mathsf{msg}^{(1)}$ *and outputs a pair of strings* (α^0, α^1) *or* $\bot$*, where the latter indicates that the state recovery failed. We require that for all* $b \in \{0,1\}$*, the following holds:*

$$\Pr\left[\mathsf{st} = \alpha^b \;\middle|\; \begin{array}{l}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda, 2)\\ (\mathsf{msg}^{(1)}, \mathsf{st}) \leftarrow \mathsf{Rec}_1(\mathsf{crs}, b)\\ (\alpha^0, \alpha^1) \leftarrow \mathsf{StaRcv}(\mathsf{td}, \mathsf{msg}^{(1)})\end{array}\right] = 1.$$

Extractability in the extraction mode: *For classical algorithms* Ext *and* Sim*, let us consider the following experiment formalized by the experiment* $\mathsf{Exp}^{\mathsf{ext\text{-}sim}}_{\mathsf{OT},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(1^\lambda, \mathsf{coin})$ *between an adversary* $\mathcal{A}$ *and the challenger:*

1. *The challenger runs* $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda, 1)$ *and sends* $(\mathsf{crs}, \mathsf{td})$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends* $\mathsf{msg}^{(1)}$ *and a pair of messages* $(z_0, z_1) \in \{0,1\}^{\ell \times 2}$*.*
3. *The challenger runs* $\mathsf{Ext}(\mathsf{td}, \mathsf{msg}^{(1)}) \to b$ *and it computes*

$$\mathsf{msg}^{(2)} \leftarrow \begin{cases}\mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, (z_0, z_1)), & \text{if } \mathsf{coin} = 0\\ \mathsf{Sim}(\mathsf{crs}, \mathsf{msg}^{(1)}, z_b) & \text{if } \mathsf{coin} = 1.\end{cases}$$

   Then, it gives $\mathsf{msg}^{(2)}$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* coin' *for* coin*. The challenger outputs* coin' *as the final output of the experiment.*

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

$$\mathsf{Adv}^{\mathsf{ext\text{-}sim}}_{\mathsf{OT},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ext\text{-}sim}}_{\mathsf{OT},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ext\text{-}sim}}_{\mathsf{OT},\mathcal{A},\mathsf{Ext},\mathsf{Sim}}(1^\lambda, 1) = 1\right]\right| \le \mathsf{negl}(\lambda).$$

Efficient state superposition: *We require that there exists a QPT "coherent version"* $\mathcal{Rec}_1$ *of* Rec_1 *that is given a quantum state* $|\psi\rangle = \sum_{b\in\{0,1\}} \alpha_b |b\rangle$ *and* crs *and works as follows:*

It first generates a state that is within negligible trace distance of the following state:

$$\sum_{b\in\{0,1\}} \alpha_b \sum_{\mathsf{msg}^{(1)} \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs}, b))} \sqrt{p_{b\to\mathsf{msg}^{(1)}}}\, |b\rangle \left|\mathsf{st}_{b\to\mathsf{msg}^{(1)}}\right\rangle \left|\mathsf{msg}^{(1)}\right\rangle$$

where $p_{b\to\mathsf{msg}^{(1)}}$ *is the probability that* $\mathsf{Rec}_1(\mathsf{crs}, b)$ *outputs* $\mathsf{msg}^{(1)}$*. Then, it measures the last register to obtain* $\mathsf{msg}^{(1)}$ *and outputs the residual quantum state* $|\psi'\rangle$*.*

We remark that implementing the above by simply running the classical algorithm in the superposition does not work, since it may leave the entanglement with the randomness for Rec_1*.*

A.2 Dual-mode OT from LWE

In this subsection, we show that the dual-mode OT based on LWE in [PVW08] satisfies our requirements of special dual-mode OT as in Definition A.1 under a certain parameter regime.

Lattice preliminaries We review definitions and lemmata on lattice-based cryptography.

For $q \in \mathbb{N}$, we write $\mathbb{Z}_q$ to mean the additive group of integers modulo q. We represent elements of $\mathbb{Z}_q$ by elements in $\mathbb{Z} \cap (-q/2, q/2]$ and often identify the elements of $\mathbb{Z}_q$ and their representatives. For a vector $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|$ denotes the Euclidean norm of $\mathbf{x}$ and $\|\mathbf{x}\|_\infty$ denotes the infinity norm of $\mathbf{x}$, that is, $\|\mathbf{x}\|_\infty := \max_{i\in[n]} |x_i|$.

For $m \in \mathbb{N}$, the Gaussian function $\rho_\sigma : \mathbb{R}^m \to [0,1]$ with a scaling parameter $\sigma > 0$ is defined as follows:

$$\rho_\sigma(\mathbf{x}) := \exp\left(-\pi\|\mathbf{x}\|^2/\sigma^2\right).$$

For any $q \in \mathbb{N}$ and $\sigma > 0$, we define the truncated discrete Gaussian distribution $D_{\mathbb{Z}_q,\sigma}$ with the support $\{x \in \mathbb{Z}_q : |x| \leq \sigma\}$ by the following probability density function:

$$D_{\mathbb{Z}_q,\sigma}(x) = \frac{\rho_\sigma(x)}{\sum_{y \in \mathbb{Z}_q : |y| \leq \sigma} \rho_\sigma(y)}.$$

More generally, for any $q, m \in \mathbb{N}$ and $\sigma > 0$, we define the truncated discrete Gaussian distribution $D_{\mathbb{Z}_q^m,\sigma}$ with the support $\{\mathbf{x} \in \mathbb{Z}_q^m : \|\mathbf{x}\|_\infty \leq \sigma\}$ by the following probability density function:

$$D_{\mathbb{Z}_q^m,\sigma}(x_1, \ldots, x_m) = \prod_{i \in [m]} D_{\mathbb{Z}_q,\sigma}(x_i).$$

Lemma A.2 (Noise smudging [BCM+21]). *For $q, m \in \mathbb{N}$, $\sigma > 0$, and $\mathbf{e} \in \mathbb{Z}_q^m$ such that $\|\mathbf{e}\| \leq \sigma\sqrt{m}$, the statistical distance between $D_{\mathbb{Z}_q^m,\sigma}$ and $D_{\mathbb{Z}_q^m,\sigma} + \mathbf{e}$ is at most $\sqrt{2\left(1 - \exp\left(-2\pi\sqrt{m}\|\mathbf{e}\|/\sigma\right)\right)}$*

Definition A.3 (Learning with errors). *Let $n, m, q \in \mathbb{N}$ and χ be a distribution over $\mathbb{Z}_q^m$, where n, m, q, χ depends on the security parameter λ. We say that the learning with errors (LWE) problem $\mathrm{LWE}_{n,m,q,\chi}$ is quantumly hard if for any QPT adversary $\mathcal{A}$,*

$$\left|\Pr\left[\mathcal{A}(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top) = 1\right] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and $\mathbf{e} \leftarrow \chi$.

Lemma A.4 (Lattice trapdoors [MP12]). *Let $n, m, q \in \mathbb{N}$ with $m = \Omega(n \log q)$. There exists a pair of PPT algorithms $(\mathsf{TrapGen}, \mathsf{Invert})$ satisfying the following.*

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{A}, \mathbf{T})$: *On input 1^n, 1^m, and q, it outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor $\mathbf{T}$. The statistical distance between the distribution of $\mathbf{A}$ generated by $\mathsf{TrapGen}(1^n, 1^m, q)$ and the uniform distribution over $\mathbb{Z}_q^{n \times m}$ is negligible in n.*
- $\mathsf{Invert}(\mathbf{A}, \mathbf{T}, \mathbf{v}) \to (\mathbf{s}', \mathbf{e}')$: *On input $\mathbf{A}$, $\mathbf{T}$, and $\mathbf{v}$, it returns vectors $\mathbf{s}' \in \mathbb{Z}_q^n$ and $\mathbf{e}' \in \mathbb{Z}_q^m$. The following holds for some constant $C > 0$: If $(\mathbf{A}, \mathbf{T})$ is generated by $\mathsf{TrapGen}(1^n, 1^m, q)$ and $\mathbf{v} = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ for $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}_q^m$ such that $\|\mathbf{e}\| \leq q/(C\sqrt{n \log q})$, $\mathsf{Invert}(\mathbf{A}, \mathbf{T}, \mathbf{v})$ outputs $(\mathbf{s}, \mathbf{e})$ with an overwhelming probability in n.*

Lemma A.5 ([GPV08, PVW08][29]). *Let $n, m \in \mathbb{N}$ and q be a prime. For any $\delta, \tau > 0$, let $M_{\delta,\tau} \subseteq \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ be the subset consisting of $(\mathbf{A}, \mathbf{v}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ such that the distribution $\{(\mathbf{At}, \mathbf{v}^\top \mathbf{t}) : \mathbf{t} \leftarrow D_{\mathbb{Z}_q^m,\tau}\}$ has statistical distance within at most δ from the uniform distribution. There is a PPT algorithm $\mathsf{IsMessy}$ with single-bit outputs that satisfies the following. If $m \geq 2(n+1)\log q$ and $\tau \geq \sqrt{qm}\log^2 m$, there is $\delta = \mathsf{negl}(n)$ such that for an overwhelming fraction of $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, the following properties hold.*

- *For all but an at most $(1/2\sqrt{q})^m$ fraction of $\mathbf{v} \in \mathbb{Z}_q^m$, $\mathsf{IsMessy}(\mathbf{A}, \mathbf{T}, \mathbf{v})$ returns 1 with overwhelming probability.*
- *For any $\mathbf{v} \in \mathbb{Z}_q^m$ such that $(\mathbf{A}, \mathbf{v}) \notin M_{\delta,\tau}$, $\mathsf{IsMessy}(\mathbf{A}, \mathbf{T}, \mathbf{v})$ returns 0 with an overwhelming probability.*

Construction It is easy to see that all the properties required for special dual-mode OT are preserved under parallel repetition. Therefore, it suffices to construct it for the case of $\ell = 1$. The scheme is described below, where the parameters are specified below the description.

$\mathsf{CRSGen}(1^\lambda, \mathsf{mode})$**:** Given the security parameter 1^λ and $\mathsf{mode} \in \{1, 2\}$, generate $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.

- If $\mathsf{mode} = 1$ (i.e., in the extractable mode), choose $\mathbf{v} \leftarrow \mathbb{Z}_q^m$.

[29] In [GPV08, PVW08], they consider the untruncated discrete Gaussian distribution but this does not affect the statement since the truncated and untruncated discrete Gaussian distributions are negligibly close.

- If mode $= 2$ (i.e., in the hiding mode), choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z}_q^m,\sigma}$ and compute $\mathbf{v}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$.

Output $\mathsf{crs} = (\mathbf{A}, \mathbf{v})$ and $\mathsf{td} = (\mathbf{A}, \mathbf{T}, \mathbf{v})$.

$\mathsf{Rec}_1(\mathsf{crs}, b)$**:** On input $\mathsf{crs} = (\mathbf{A}, \mathbf{v})$ and $b \in \{0,1\}$, choose $\mathbf{r} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e}' \leftarrow D_{\mathbb{Z}_q^m,\sigma'}$, compute $\mathbf{u}_b^\top = \mathbf{r}^\top \mathbf{A} + \mathbf{e}'^\top$ and $\mathbf{u}_{1-b}^\top = \mathbf{u}_b^\top + (-1)^b \mathbf{v}^\top$, and output $\mathsf{msg}^{(1)} = \mathbf{u}_0$ and $\mathsf{st} = \mathbf{r}$.

$\mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, (z_0, z_1)) \to \mathsf{msg}^{(2)}$**:** On input crs, $\mathsf{msg}^{(1)} = \mathbf{u}_0$, and $(z_0, z_1) \in \{0,1\}^2$, let $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{v}$, choose $\mathbf{t}_b \leftarrow D_{\mathbb{Z}_q^m,\tau}$, compute $\mathbf{w}_b = \begin{pmatrix} \mathbf{A} \\ \mathbf{u}_b^\top \end{pmatrix} \mathbf{t}_b + \begin{pmatrix} \mathbf{0} \\ z_b \lfloor q/2 \rfloor \mod q \end{pmatrix}$ for $b \in \{0,1\}$, and output $\mathsf{msg}^{(2)} = (\mathbf{w}_0, \mathbf{w}_1)$.

$\mathsf{Rec}_2(\mathsf{crs}, b, \mathsf{st}, \mathsf{msg}^{(2)}) \to z$**:** On input crs, $b \in \{0,1\}$, $\mathsf{st} = \mathbf{r}$, and $\mathsf{msg}^{(2)} = (\mathbf{w}_0, \mathbf{w}_1)$, let $\mathbf{w}_{b,[1,n]} \in \mathbb{Z}_q^n$ be the vector consisting of the first n coordinates of $\mathbf{w}_b$ and $w_{b,n+1} \in \mathbb{Z}_q$ be the remaining coordinate, output 0 if $\left| w_{b,n+1} - \mathbf{r}^\top \mathbf{w}_{b,[1,n]} \mod q \right| < q/4$ and output 1 otherwise.

Parameters We choose the parameters $n, m, q, \sigma, \sigma', \tau$ so that the following hold.

- $m \geq 2(n+1)\log q$
- $\mathrm{LWE}_{n,m,q,D_{\mathbb{Z}_q^m,\sigma}}$ is quantumly hard. We need $O(q/\sigma) < 2^{n^\theta}$ for some $0 < \theta < 1$.
- $\sigma'/\sigma = 2^{\omega(\log \lambda)}$
- $\sigma' \leq q/(C\sqrt{nm \log q})$ where C is the constant in Lemma A.4.
- $\tau \geq \sqrt{qm} \log^2 m$
- $m^2 \sigma' \tau \leq q/8$

For example, we can choose the parameters as follows (for sufficiently large λ):

$$n = \lambda^c, \quad m = 3n^2, \quad q \approx 2^\lambda, \quad \sigma = n^2 \quad \sigma' = 2^{\lambda/6}, \quad \tau = 2^{2\lambda/3}.$$

where $c > 1$ is some constant.

Security

Theorem A.6. *If* $\mathrm{LWE}_{n,m,q,D_{\mathbb{Z}_q^m,\sigma}}$ *is quantumly hard, then the above scheme satisfies the requirements of special dual-mode OT (Definition A.1).*

Evaluation correctness: For any $b \in \{0,1\}$, $\mathsf{mode} \in \{1,2\}$, and $(z_0, z_1) \in \{0,1\}^2$, suppose that $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda, \mathsf{mode})$, $(\mathsf{msg}^{(1)}, \mathsf{st}) \leftarrow \mathsf{Rec}_1(\mathsf{crs}, b)$, $\mathsf{msg}^{(2)} \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, (z_0, z_1))$. Using the notations in the description of the scheme, we have

$$\begin{aligned}
&\left| w_{b,n+1} - \mathbf{r}^\top \mathbf{w}_{b,[1,n]} \mod q \right| \\
&= \left| \mathbf{u}_b^\top \mathbf{t}_b + z_b \lfloor q/2 \rfloor - \mathbf{r}^\top \mathbf{A} \mathbf{t}_b \mod q \right| \\
&= \left| (\mathbf{r}^\top \mathbf{A} + \mathbf{e}'^\top) \mathbf{t}_b + z_b \lfloor q/2 \rfloor - \mathbf{r}^\top \mathbf{A} \mathbf{t}_b \mod q \right| \\
&= \left| \mathbf{e}'^\top \mathbf{t}_b + z_b \lfloor q/2 \rfloor \mod q \right|
\end{aligned}$$

Since $\mathbf{e}'$ is sampled from $D_{\mathbb{Z}_q^m,\sigma'}$ and $\mathbf{t}$ is sampled from $D_{\mathbb{Z}_q^m,\tau}$, we have $\|\mathbf{e}'\|_\infty \le \sqrt{m}\sigma'$ and $\|\mathbf{t}_b\|_\infty \le \sqrt{m}\tau$. Thus, $|\mathbf{e}'^\top \mathbf{t}_b| \le m^2\sigma'\tau \le q/8$. Thus, when $z_b = 0$, we have $\left|w_{b,n+1} - \mathbf{r}^\top \mathbf{w}_{b,[1,n]} \mod q\right| \le q/8 < q/4$. Similarly, when $z_b = 1$, we have $\left|w_{b,n+1} - \mathbf{r}^\top \mathbf{w}_{b,[1,n]} \mod q\right| \ge \lfloor q/2 \rfloor - q/8 > q/4$. The above together imply the evaluation correctness.

Unique state: Let $\mathsf{mode} \in \{1,2\}$, $(\mathsf{crs} = (\mathbf{A},\mathbf{v}), \mathsf{td} = (\mathbf{A},\mathbf{T},\mathbf{v})) \in \mathsf{Supp}(\mathsf{CRSGen}(1^\lambda,\mathsf{mode}))$, $b \in \{0,1\}$, and $(\mathsf{msg}^{(1)} = \mathbf{u}_0, \mathsf{st} = \mathbf{r}) \in \mathsf{Supp}(\mathsf{Rec}_1(\mathsf{crs},b))$. Then we have $\mathbf{u}_0 = \mathbf{r}^\top\mathbf{A} + \mathbf{e}'^\top - b\mathbf{v}^\top$ for some $\mathbf{e}'$ such that $\|\mathbf{e}'\|_\infty \le \sigma' \le q/(C\sqrt{nm\log q})$. By Lemma A.4, there is a PPT algorithm that recovers $\mathsf{st} = \mathbf{r}$ from $\mathbf{A}$, $\mathbf{T}$, b, $\mathbf{v}$, and $\mathsf{msg}^{(1)} = \mathbf{u}_0$. This implies the uniqueness of the state.

Mode indistinguishability: The only difference between the two modes is that $\mathbf{v}$ is uniformly random when $\mathsf{mode} = 1$ and is an LWE instance when $\mathsf{mode} = 2$. Thus, the mode indistinguishability directly follows from the assumed quantum hardness of $\mathrm{LWE}_{n,m,q,D_{\mathbb{Z}_q^m,\sigma}}$.

Statistical security against malicious senders in the hiding mode: Fix $\mathsf{crs} = (\mathbf{A},\mathbf{v})$ in the support of $\mathsf{CRSGen}(1^\lambda,2)$. It satisfies $\mathbf{v}^\top = \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}_q^m$ such that $\|\mathbf{e}\|_\infty \le \sigma$. Suppose $(\mathsf{msg}_b^{(1)}, \mathsf{st}) \leftarrow \mathsf{Rec}_1(\mathsf{crs},b)$ for $b \in \{0,1\}$. Then we can write

$$\mathsf{msg}_0^{(1)} = \mathbf{r}_0^\top\mathbf{A} + \mathbf{e}_0'^\top$$

and

$$\mathsf{msg}_1^{(1)} = \mathbf{r}_1^\top\mathbf{A} + \mathbf{e}_1'^\top - \mathbf{v} = (\mathbf{r}_1^\top - \mathbf{s}^\top)\mathbf{A} + (\mathbf{e}_1'^\top - \mathbf{e}^\top)$$

for $\mathbf{r}_0,\mathbf{r}_1 \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e}_0',\mathbf{e}_1' \leftarrow D_{\mathbb{Z}_q^m,\sigma'}$. Since $\mathbf{r}_1^\top$ is uniformly random, $(\mathbf{r}_1^\top - \mathbf{s}^\top)$ is also uniformly random. Moreover, since $\|\mathbf{e}\| \le \sqrt{m}\sigma$, the statistical distance between the distributions of $\mathbf{e}_0'^\top$ and $(\mathbf{e}_1'^\top - \mathbf{e}^\top)$ is at most $\sqrt{2\left(1-\exp\left(-2\pi m\sigma/\sigma'\right)\right)} \le \sqrt{4\pi m\sigma/\sigma'} = \mathsf{negl}(\lambda)$ by Lemma A.2 and $\sigma'/\sigma = 2^{\omega(\log\lambda)}$. Thus, the distributions of $\mathsf{msg}_0^{(1)}$ and $\mathsf{msg}_1^{(1)}$ are statistically negligibly close in λ.

State recoverability in the hiding mode: Let StaRcv be an algorithm that works as follows.

$\mathsf{StaRcv}(\mathsf{td},\mathsf{msg}^{(1)})$**:** On input $\mathsf{td} = (\mathbf{A},\mathbf{T},\mathbf{v})$ and $\mathsf{msg}^{(1)} = \mathbf{u}_0$, compute $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{v}$, run $(\mathbf{r}_b,\mathbf{e}_b') \leftarrow \mathsf{Invert}(\mathbf{A},\mathbf{T},\mathbf{u}_b)$ for $b \in \{0,1\}$ and output $(\mathbf{r}_0,\mathbf{r}_1)$.

Suppose $(\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{CRSGen}(1^\lambda,2)$ and $(\mathsf{msg}^{(1)},\mathsf{st}) \leftarrow \mathsf{Rec}_1(\mathsf{crs},b)$. Then $\mathsf{msg}^{(1)} = \mathbf{u}_0 = \mathbf{r}^\top\mathbf{A} + \mathbf{e}'^\top - b\mathbf{v}$ and $\mathsf{st} = \mathbf{r}$ where $\mathbf{r} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e}' \leftarrow D_{\mathbb{Z}_q^m,\sigma'}$. Then we have $\mathbf{u}_b^\top = \mathbf{u}_0^\top + b\mathbf{v} = \mathbf{r}^\top\mathbf{A} + \mathbf{e}'^\top$. Thus, for $(\mathbf{r}_0,\mathbf{r}_1)$ output by $\mathsf{StaRcv}(\mathsf{td},\mathsf{msg}^{(1)})$, we have $\mathbf{r}_b = \mathbf{r}$ by Lemma A.4 and $\|\mathbf{e}'\| \le \sqrt{m}\sigma' \le q/(C\sqrt{n\log q})$.

Extractability in the extraction mode: Let Ext and Sim be algorithms that work as follows.

$\mathsf{Ext}(\mathsf{td},\mathsf{msg}^{(1)})$**:** On input $\mathsf{td} = (\mathbf{A},\mathbf{T},\mathbf{v})$ and $\mathsf{msg}^{(1)} = \mathbf{u}_0$, compute $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{v}$ and run $\beta_b \leftarrow \mathsf{IsMessy}(\mathbf{A},\mathbf{T},\mathbf{u}_b)$. If $\beta_0 = 1$, output 1, else if $\beta_1 = 1$, output 0, otherwise output $\bot$.

$\mathsf{Sim}(\mathsf{crs},\mathsf{msg}^{(1)},z_b)$**:** On input $\mathsf{crs} = (\mathbf{A},\mathbf{v})$, $\mathsf{msg}^{(1)} = \mathbf{u}_0$, and $z_b \in \{0,1\}$, Run $(\mathbf{w}_0,\mathbf{w}_1) \leftarrow \mathsf{Send}(\mathsf{crs},\mathsf{msg}^{(1)},(z_b,z_b))$ and output $(\mathbf{w}_0,\mathbf{w}_1)$. Note that it uses the same z_b for generating both $\mathbf{w}_0$ and $\mathbf{w}_1$.

Fix $(\mathbf{A},\mathbf{T})$ that satisfies the two items of Lemma A.5. Let $G \subseteq \mathbb{Z}_q^m$ be the subset in the first item of Lemma A.5, that is, for any $\mathbf{v} \notin G$, $\mathsf{IsMessy}(\mathbf{A},\mathbf{T},\mathbf{v})$ returns 1 with overwhelming probability. Since G consists of at most $(1/2\sqrt{q})^m$ fraction of elements of $\mathbb{Z}_q^m$, $|G| \le (\sqrt{q}/2)^m$. Therefore, we have

$$\#\{\mathbf{v} \in \mathbb{Z}_q^m : \exists \mathbf{u}_0 \in \mathbb{Z}_q^m \text{ s.t. } \mathbf{u}_0 \in G \wedge \mathbf{u}_1 \in G\} \le |G|^2 \le (q/4)^m$$

where $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{v}$. Thus, we have

$$\Pr_{\mathbf{v}\leftarrow\mathbb{Z}_q^m}[\exists \mathbf{u}_0 \in \mathbb{Z}_q^m \text{ s.t. } \mathbf{u}_0 \in G \wedge \mathbf{u}_0 + \mathbf{v} \in G] \leq 4^{-m}.$$

Thus, when td is honestly generated, for any $\mathsf{msg}^{(1)}$, $\mathsf{Ext}(\mathsf{td}, \mathsf{msg}^{(1)})$ returns a non-$\bot$ bit with overwhelming probability. By the second item of Lemma A.5, when $\mathsf{Ext}(\mathsf{td}, \mathsf{msg}^{(1)})$ returns $b \in \{0,1\}$, $(\mathbf{A}, \mathbf{u}_{1-b}) \in M_{\delta,\tau}$ with overwhelming probability where $M_{\delta,\tau}$ is as defined in Lemma A.5. Note that the only difference between $\mathsf{Send}(\mathsf{crs}, \mathsf{msg}^{(1)}, (z_0, z_1))$ and $\mathsf{Sim}(\mathsf{crs}, \mathsf{msg}^{(1)}, z_b)$ is that $\mathbf{w}_{1-b}$ is generated using z_{1-b} in the former but using z_b in the latter. By the definition of $M_{\delta,\tau}$, when $(\mathbf{A}, \mathbf{u}_{1-b}) \in M_{\delta,\tau}$, the distribution of $\mathbf{w}_{1-b}$ generated in either way has statistical distance at most $\delta = \mathsf{negl}(n)$ from the uniform distribution. Thus, the above implies extractability in the extraction mode.

Efficient state superposition: It suffices to show that there is a QPT algorithm that is given $b \in \{0,1\}$, $\mathbf{A}$, and $\mathbf{v}$, and generates a state that is within negligible trace distance of the following state:

$$\sum_{\mathbf{r}\in\mathbb{Z}_q^n, \mathbf{e}'\in \mathrm{Supp}(D_{\mathbb{Z}_q^m,\sigma'})} \sqrt{p_{\mathbf{r},\mathbf{e}'}}\, |b\rangle\, |\mathbf{r}\rangle \left|\mathbf{r}^\top \mathbf{A} + \mathbf{e}'^\top + (-1)^b \mathbf{v}^\top\right\rangle$$

where $p_{\mathbf{r},\mathbf{e}'}$ is the probability that $\mathsf{Rec}_1(\mathsf{crs}, b)$ chooses the randomness $(\mathbf{r}, \mathbf{e}')$.

This can be done as follows. First, as shown in [Reg09, BCM+21], the Grover-Rudolph algorithm [GR02] enables us to generate the following Gaussian superposition up to negligible error:

$$\sum_{\mathbf{e}'\in \mathrm{Supp}(D_{\mathbb{Z}_q^m,\sigma'})} \sqrt{D_{\mathbb{Z}_q^m,\sigma'}(\mathbf{e}')}\, \left|\mathbf{e}'\right\rangle.$$

We prepare $|b\rangle$ in another register and generate a uniform superposition over $\mathbf{r} \in \mathbb{Z}_q^n$ in yet another register. Then, the whole system can be written as follows:

$$\begin{aligned}
&|b\rangle \otimes q^{-n/2} \sum_{\mathbf{r}\in\mathbb{Z}_q^n} |\mathbf{r}\rangle \otimes \sum_{\mathbf{e}'\in \mathrm{Supp}(D_{\mathbb{Z}_q^m,\sigma'})} \sqrt{D_{\mathbb{Z}_q^m,\sigma'}(\mathbf{e}')}\, \left|\mathbf{e}'\right\rangle \\
=& \sum_{\mathbf{r}\in\mathbb{Z}_q^n, \mathbf{e}'\in \mathrm{Supp}(D_{\mathbb{Z}_q^m,\sigma'})} \sqrt{p_{\mathbf{r},\mathbf{e}'}}\, |b\rangle\, |\mathbf{r}\rangle\, \left|\mathbf{e}'\right\rangle
\end{aligned}$$

where the equality follows from $p_{\mathbf{r},\mathbf{e}'} = q^{-n} D_{\mathbb{Z}_q^m,\sigma'}(\mathbf{e}')$.

By using $\mathbf{A}$ and $\mathbf{v}$, we can coherently compute $\mathbf{r}^\top \mathbf{A} + \mathbf{e}'^\top + (-1)^b \mathbf{v}^\top$ in another register. This yields the following state:

$$\sum_{\mathbf{r}\in\mathbb{Z}_q^n, \mathbf{e}'\in \mathrm{Supp}(D_{\mathbb{Z}_q^m,\sigma'})} \sqrt{p_{\mathbf{r},\mathbf{e}'}}\, |b\rangle\, |\mathbf{r}\rangle\, \left|\mathbf{e}'\right\rangle \left|\mathbf{r}^\top \mathbf{A} + \mathbf{e}'^\top + (-1)^b \mathbf{v}^\top\right\rangle.$$

Finally, uncomputing the third register results in the desired state.

A.3 From OT to SFE via Garbled Circuit

We explain how to transform special dual-mode OT into special dual-mode SFE using garbled circuits. We only provide a sketch since the technique is fairly standard.

Garbled circuit. We first quickly review the notion of garbled circuit. A circuit garbling scheme GC for circuits is a tuple $(\mathsf{Grbl}, \mathsf{Eval})$ of two PPT algorithms with the following syntax.

- A garbling algorithm Grbl, given a security parameter 1^λ and a circuit C with n-bit input, outputs a garbled circuit $\widetilde{C}$ and input labels $\{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}$.
- The evaluation algorithm, given a garbled circuit $\widetilde{C}$ and n input labels $\{\mathsf{lab}_i\}_{i\in[n]}$, outputs y.

As correctness, we require $\mathsf{Eval}(\widetilde{C}, \{\mathsf{lab}_{i,x[i]}\}_{i\in[n]}) = C(x)$ holds for every circuit C with n-bit input and $x \in \{0,1\}^n$, where $(\widetilde{C}, \{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, C)$. As security, we require that for any circuit C and input x, $(\widetilde{C}, \{\mathsf{lab}_{i,x[i]}\}_{i\in[n]})$ can be simulated by a computationally indistinguishable distribution given only $1^{|C|}$ and $C(x)$. A circuit garbling scheme can be realized based on OWFs [Yao86].

Transformation based on garbled circuit. We now sketch how to transform a special dual-mode OT scheme $\mathsf{OT} = \mathsf{OT}.(\mathsf{CRSGen}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$ into a special dual-mode SFE scheme $\mathsf{SFE} = \mathsf{SFE}.(\mathsf{CRSGen}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$ using a circuit garbling scheme $\mathsf{GC} = (\mathsf{Grbl}, \mathsf{Eval})$.

$\mathsf{SFE.CRSGen}$**:** It runs $\mathsf{OT.CRSGen}$ n times and gets $(\mathsf{OT.crs}_i, \mathsf{OT.td}_i)_{i\in[n]}$. It sets $\mathsf{SFE.crs} := (\mathsf{OT.crs}_i)_{i\in[n]}$ and $\mathsf{SFE.td} := (\mathsf{OT.td}_i)_{i\in[n]}$.

$\mathsf{SFE.Rec}_1$**:** Given $\mathsf{SFE.crs} := (\mathsf{OT.crs}_i)_{i\in[n]}$ and $x \in \{0,1\}^n$, it executes $(\mathsf{OT.msg}_i^{(1)}, \mathsf{OT.st}_i) \leftarrow \mathsf{OT.Rec}_1(\mathsf{OT.crs}_i, x[i])$ for every $i \in [n]$ and outputs $\mathsf{SFE.msg}^{(1)} := (\mathsf{OT.msg}_i^{(1)})_{i\in[n]}$ and $\mathsf{SFE.st} := (\mathsf{OT.st}_i)_{i\in[n]}$.

$\mathsf{SFE.Send}$**:** Given $\mathsf{SFE.crs} := (\mathsf{OT.crs}_i)_{i\in[n]}$, $\mathsf{SFE.msg}^{(1)} := (\mathsf{OT.msg}_i^{(1)})_{i\in[n]}$, and a circuit C, it first executes $(\widetilde{C}, (\mathsf{lab}_{i,b})_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, C)$. It then runs $\mathsf{OT.msg}_i^{(2)} \leftarrow \mathsf{OT.Send}(\mathsf{OT.crs}_i, \mathsf{OT.msg}_i^{(1)}, (\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}))$ for every $i \in [n]$ and outputs $\mathsf{SFE.msg}^{(2)} := (\widetilde{C}, (\mathsf{OT.msg}_i^{(2)})_{i\in[n]})$.

$\mathsf{SFE.Rec}_2$**:** Given $\mathsf{SFE.crs} := (\mathsf{OT.crs}_i)_{i\in[n]}$, $x \in \{0,1\}^n$, $\mathsf{SFE.st} := (\mathsf{OT.st}_i)_{i\in[n]}$, and $\mathsf{SFE.msg}^{(2)} := (\widetilde{C}, (\mathsf{OT.msg}_i^{(2)})_{i\in[n]})$, it executes $\mathsf{lab}_i \leftarrow \mathsf{OT.Rec}_2(\mathsf{OT.crs}_i, x[i], \mathsf{OT.st}_i, \mathsf{OT.msg}_i^{(2)})$ for every $i \in [n]$ and outputs $y \leftarrow \mathsf{Eval}(\widetilde{C}, (\mathsf{lab}_i)_{i\in[n]})$.

We can easily check that the evaluation correctness of SFE follows from that of OT and GC. SFE uses n instances of OT in parallel and uses GC on top of it. Especially, the CRS, the trapdoor, and the first message of SFE consist of just n copies of those of OT. By this construction, the unique state property, mode indistinguishability, statistical security against malicious senders in the hiding mode, state recoverability in the hiding mode, and efficient state superposition property of SFE directly follow from those of OT, respectively. Finally, we can prove the extractability in the extraction mode of SFE based on that of OT and the security of GC. Roughly speaking, the sender message $\mathsf{SFE.msg}^{(2)}$ of SFE is composed of $(\widetilde{C}, (\mathsf{OT.msg}_i^{(2)})_{i\in[n]})$, and the first component can be simulated from $1^{|C|}$ and $C(x)$ by the security of GC and the second one can be simulated by the extractability in the extraction mode of OT together with the simulated labels of GC.