Public-Key Watermarking Schemes for Pseudorandom Functions

Rupeng Yang[1,2]*, Zuoxia Yu[1,2], Man Ho Au[1], and Willy Susilo[2]

[1] Department of Computer Science, The University of Hong Kong, Hong Kong, China
orbbyrp@gmail.com, zuoxia.yu@gmail.com, allenau@cs.hku.hk
[2] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong NSW, Australia
wsusilo@uow.edu.au

Abstract. A software watermarking scheme can embed a message into a program while preserving its functionality. The embedded message can be extracted later by an extraction algorithm, and no one could remove it without significantly changing the functionality of the program. A watermarking scheme is public key if neither the marking procedure nor the extraction procedure needs a watermarking secret key. Prior constructions of watermarking schemes mainly focus on watermarking pseudorandom functions (PRFs), and the major open problem in this direction is to construct a public-key watermarkable PRF.

In this work, we solve the open problem via constructing public-key watermarkable PRFs with different trade-offs from various assumptions, ranging from standard lattice assumptions to the existence of indistinguishability obfuscation. To achieve the results, we first construct watermarking schemes in a weaker model, where the extraction algorithm is provided with a "hint" about the watermarked PRF key. Then we upgrade the constructions to standard watermarking schemes using a robust unobfuscatable PRF. We also provide the first construction of robust unobfuscatable PRF in this work, which is of independent interest.

* Corresponding author.

Table of Contents

1 Introduction

A software watermarking scheme allows one to embed a message into a program without significantly changing its functionality. Moreover, any attempt to remove the embedded message would destroy the functionality of the watermarked program. Watermarking schemes have many real-world applications, including ownership protection, traitor tracing, etc., and recently, it is also applied in new applications such as quantum copy-protection [ALL+21, KNY21].

The theoretical study of software watermarking is initiated by Barak et al. [BGI+01] and Hopper et al. [HMW07], where formal definitions are presented. They also explore the (im)possibility to achieve certain definitions of watermarking and study connections between different definitions. However, neither of them provides a concrete construction. It is notoriously hard to construct watermarking schemes with provable security, and early constructions [NSS99, YF11, Nis13] are only proven secure against restricted adversaries, which are not allowed to change the format of the watermarked object.

Cohen et al. [CHN+16] propose the first watermarking scheme with provable security against arbitrary removal strategies. They also show that it is impossible to watermark learnable functions. A natural class of non-learnable functions are the cryptographic ones, such as pseudorandom function (PRF). Therefore, Cohen et al. and subsequent works mainly study watermarking for cryptographic functionalities, with a primary focus on watermarkable PRFs, which can be applied to construct watermarking schemes for various primitives in minicrypt and has many real-world applications as discussed in [CHN+16]. In this work, we also consider watermarking schemes for PRFs and refer the readers to Appendix A for other related work.

Watermarking PRFs. A watermarkable PRF is a PRF family F with two additional algorithms, namely, the marking algorithm and the extraction algorithm. The marking algorithm takes as input the mark key, a message, and a PRF key k, and outputs a watermarked circuit, which approximately evaluates $\mathsf{F}_k(\cdot)$. The extraction algorithm extracts the embedded message from a watermarked circuit with an extraction key. Its main security property is unremovability, which requires that given a watermarked circuit $\mathtt{C}^*$ for a random PRF key (namely, the challenge key), the adversary is not able to produce a circuit that agrees with $\mathtt{C}^*$ on almost all inputs, yet the extraction algorithm fails to extract the original message from it. The mark key and the extraction key are generated when setting up the scheme, and a watermarking scheme is *public key* if both the mark key and the extraction key can be made public. Also, a secret-key watermarking scheme has *public extraction* (resp. *public marking*) if it is secure against an adversary with the extraction key (resp. *mark key*).

The first construction of watermarkable PRF is presented by Cohen et al. in [CHN+16]. The construction is based on an indistinguishability obfuscation (iO) and has public extraction. Then in [YAL+19], Yang et al. improve Cohen et al.'s scheme to further achieve collusion resistant security, where the adversary is allowed to view multiple watermarked circuits for the challenge key. However, in both constructions, the mark key should be kept private.

In another line of work, Boneh et al. [BLW17] propose a new approach that builds watermarkable PRF from variants of constrained PRFs [BW13, KPTZ13, BGI14]. The scheme in [BLW17] is still instantiated from iO. Then in [KW17], Kim and Wu present the first watermarkable PRF from standard assumptions. Later, in [PS18, PS20], Peikert and Shiehian also instantiate the construction in [BLW17] from standard lattice assumptions. However, these schemes need a secret key in both the marking algorithm and the extraction algorithm.

Subsequent works explore how to construct watermarkable PRF with stronger security from standard assumptions. In [QWZ18, KW19], watermarkable PRFs that have public marking are constructed. The schemes also achieve security with extraction queries, where the adversary can learn extraction results of its generated circuits. However, they do not have standard pseudorandomness against an adversary with the extraction key. Recently, in [YAYX20], Yang et al. upgrade previous watermarkable PRFs from standard assumptions to further achieve collusion resistance. Nonetheless, none of these schemes support public extraction.

Motivation. There are no candidate constructions of public-key watermarkable PRFs in the literature. Even worse, in previous secret-key watermarkable PRFs, the watermarking authority, who holds the secret key, can remove the watermark embedded in any watermarked circuit. This is a severe threat to all users. In contrast, in a public-key watermarking scheme, no one has this privilege since the scheme does not have such secret key. Therefore, no trust assumption is needed in a public-key watermarking scheme and it can provide a much better security guarantee in practice. This raises the following natural question:

Can we construct public-key watermarkable PRFs?

There are a few technical barriers towards this goal. First, existing approaches for achieving public marking [QWZ18, KW19] will lead to a watermarkable PRF that is only pseudorandom against adversaries without the extraction key of the scheme, and one can compromise its pseudorandomness using the extraction key. This relaxed pseudorandomness is acceptable in the secret extraction setting since the extraction key is held by an authority. However, there is no authority for a public-key watermarking scheme. Thus, if we combine previous ideas for obtaining public marking and that for obtaining public extraction, we will get a public-key watermarkable "PRF" without pseudorandomness.

Moreover, known techniques for constructing watermarkable PRFs with public extraction rely on iO. Despite recent breakthrough [JLS21] that constructs indistinguishability obfuscations from well-founded assumptions, the construction is not post-quantum secure. Thus, new ideas that construct watermarkable PRFs with public extraction from standard lattice assumptions are desired.

Our Results. In this work, we affirmatively answer the above question and present constructions of public-key watermarking schemes for PRFs. To overcome the technical issues, we introduce a new framework that constructs watermarkable PRFs from an *unobfuscatable PRF* [BGI+01] with *robust learnability* [BP13] and a new primitive called *hinting watermarkable PRF*, which relaxes a standard watermarking scheme by allowing its extraction algorithm to use an extra "hint" about the watermarked PRF key. We remark that via

	Message Embedding	Public Marking	Public Extraction	Unremovability		Pseudorandomness		Assumptions
				ϵ	CR	UK	MK	
[CHN+16]	✓	✗	✓	$\approx \frac{1}{2}$	✗	✓	✗	Lattice+iO
[BLW17]	✓	✗	✗	negl	✗	✓	✗	Lattice+iO
[YAL+19]	✓	✗	✓	negl	✓	✓	✗	Lattice+iO
[KW17]	✓	✗	✗	negl	✗	✓	✗	Lattice
[QWZ18]	✓	✓	✗	$\approx \frac{1}{2}$	✗	✗	✗	Lattice
[KW19]	✓	✓	✗	$\approx \frac{1}{2}$	✗	✓‡	✗	Lattice
[YAYX20]	✓	✗	✗	negl	✓	✓	✗	Lattice
	✓	✓	✗	$\approx \frac{1}{2}$	✓	✓‡	✗	Lattice
This Work	✗	✓	✓	negl	-	✓	✓	Lattice
	✓	✓	✓	1/exp	✓	✓	✓	Lattice
	✗	✓	✓	$\approx \frac{1}{6}$	-	✓	✓	Lattice+FHE
	✓	✓	✓	negl	✗	✓	✓	Lattice+iO
	✓	✓	✓	$\approx \frac{1}{6}$	✗	✓	✓	Lattice+FHE+iO

‡: A weaker T-restricted pseudorandomness (see [KW19]) is achieved.

Table 1: Properties achieved by existing watermarkable PRFs. For the parameter ϵ, the term "$\approx \frac{1}{2}$" denotes that $\epsilon = \frac{1}{2} - \frac{1}{\text{poly}}$, the term "negl" denotes that ϵ can be any negligible function, the term "1/exp" denotes that ϵ is equal to a concrete value that is exponentially-small, and the term "$\approx \frac{1}{6}$" denotes that $\epsilon = \frac{1}{6} - \frac{1}{\text{poly}}$. We use "CR" to denote collusion resistant unremovability. We consider pseudorandomness against an adversary with the mark key and the extraction key (even for a secret-key watermarking scheme). We use "UK" to denote pseudorandomness of PRF evaluations using unmarked keys and use "MK" to denote pseudorandomness of PRF evaluations using marked keys.

our framework, we can obtain (public-key) watermarkable PRFs with standard pseudorandomness from (public-key) hinting watermarkable PRFs with relaxed pseudorandomness, and this solves the first technical issue described above. We then construct public-key hinting watermarkable PRFs from either standard lattice assumptions or iO, with different trade-offs that will be discussed below. To obtain the lattice based constructions, we introduce some new techniques for achieving public extraction from standard lattice assumptions. Besides, we construct the first unobfuscatable PRF with robust learnability in this work. The new framework, notion and constructions may find further applications.[1]

By instantiating our constructions, we obtain public-key watermarkable PRFs from different assumptions. We consider three types of assumptions in this work, namely, standard lattice assumptions, the assumption that the GSW encryption scheme [GSW13] is circular secure[2], and the existence of iO. The three assumptions are denoted as "Lattice", "FHE", and "iO" respectively. Also, we consider constructions in either the *mark-embedding* setting, where a program is either marked or unmarked, or the *message-embedding* setting, where a marked program is embedded with a message. Besides, we use $\boldsymbol{\epsilon}$ to denote the fraction of

[1] For example, we can apply our new framework to upgrade the watermarking schemes in [QWZ18, KW19] to achieve full pseudorandomness, by viewing them as (secret-key) hinting watermarkable PRFs. This solves an open problem in these two works.

[2] Formal definition for this assumption can be found in Definition E.1.

inputs of the watermarked circuits that can be modified by the adversary when defining unremovability. More precisely, let λ be the security parameter, we have:

- From Lattice, we construct a public-key watermarkable PRF in the mark-embedding setting, where $\epsilon = negl(\lambda)$, i.e., the scheme guarantees that an adversary cannot remove the mark in a watermarked circuit if it modifies the circuit on a negligible fraction of inputs.
- From Lattice, we construct a public-key watermarkable PRF in the message-embedding setting. The scheme also has collusion resistant security. A caveat of this construction is that it only has exponentially-small ϵ, i.e., the adversary can modify the watermarked circuit on at most $M = 2^n/2^{poly(\lambda)}$ inputs, where n is the input length. Nonetheless, we still have $M = 2^{poly(\lambda)}$.
- From Lattice and FHE, we construct a public-key mark-embedding watermarkable PRF with $\epsilon = 1/6 - 1/poly(\lambda)$.
- From Lattice and iO, we construct a public-key message-embedding watermarkable PRF with $\epsilon = negl(\lambda)$.
- From Lattice, FHE and iO, we construct a public-key message-embedding watermarkable PRF with $\epsilon = 1/6 - 1/poly(\lambda)$.

Features of our constructions, together with comparison with previous watermarkable PRFs are presented in Table 1. Also, we illustrate how to instantiate our public-key watermarkable PRFs from concrete assumptions in Figure 1.

We stress that all public-key watermarkable PRFs constructed in this work have pseudorandomness for marked keys, i.e., no one could distinguish outputs of a watermarked PRF key and outputs of a random function. This property is not achieved in previous watermarkable PRFs with public extraction. This is because in these constructions, an adversary with an extraction key can extract meaningful information via oracle access to the marked key. In our construction, we circumvent this barrier by using a white-box extraction algorithm, where the algorithm must view the code of the marked key. We refer the reader to [Zha21] for a more detailed discussion on the notion of white-box tracing/extraction.

Open Problems. We initiate the study of public-key watermarkable PRFs in this work. We give mark-embedding constructions from lattice and message-embedding constructions from iO. We also construct a lattice based message-embedding scheme, but it restricts the parameter $\epsilon = 1/2^{poly(\lambda)}$. This is smaller than the parameter ϵ in previous works, which is either a constant number or restricted by any (rather than a concrete) negligible function. The main open problem is therefore to construct a message-embedding public-key watermarkable PRF with larger ϵ from standard lattice assumptions. Besides, in our mark-embedding constructions and iO based constructions, we need additionally assume circular security of the GSW scheme to achieve a constant ϵ. It will be interesting to obtain constant ϵ without such additional assumptions.

Another important security property that is not discussed in this paper is *unforgeability*, which requires that no one could watermark a new program without a mark key. This property is useful for certifying the watermarked objects in the ownership protection scenario. It was believed that watermarking schemes with

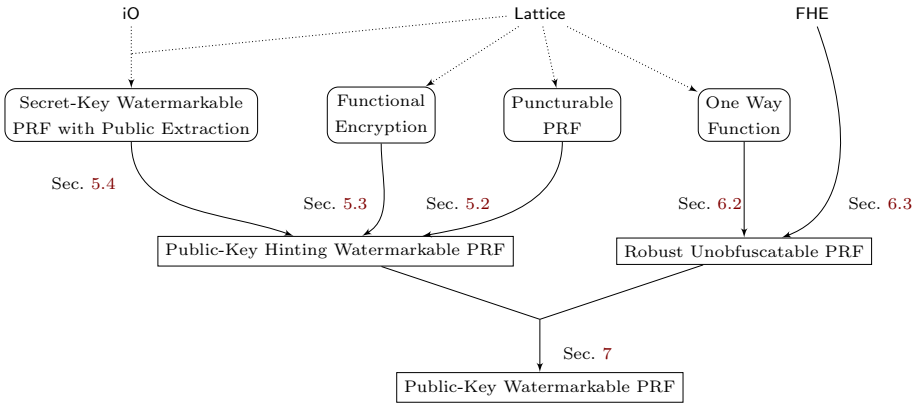

Fig. 1 The roadmap for constructing public-key watermarkable PRFs from concrete assumptions. The dotted lines denote results from previous work.

public marking contradicts with unforgeability, since there is no secret mark key in the scheme. However, as shown in [YAYX20], the conflict can be overcome via defining security in a hybrid model, where the unremovability and pseudorandomness are defined against an adversary with the mark key (i.e., the mark key can be made public when considering these two security properties), and the unforgeability is defined against an adversary without the mark key. They also construct watermarkable PRFs secure in this hybrid model, but their techniques cannot be applied to our constructions here. It is an interesting open problem to construct a public-key watermarkable PRF with unforgeability in the hybrid model.

2 Technical Overview

In this section, we provide a technical overview of our constructions of public-key watermarkable PRFs. We first consider a relaxed notion of watermarking, where each PRF key is associated with a "hint" that can be used to help extract messages. We call it hinting watermarking and in Sec. 2.1, we explain our main ideas for constructing public-key hinting watermarkable PRFs. Then in Sec. 2.2, we show how to upgrade a public-key hinting watermarkable PRF to a standard public-key watermarkable PRF by using an unobfuscatable PRF with "robust learnability". Existing constructions of unobfuscatable PRFs [BGI+01] do not have robust learnability and in Sec. 2.3, we describe how to achieve it.

2.1 Constructing Public-Key Hinting Watermarkable PRFs

The syntax of a hinting watermarkable PRF is identical to a standard watermarkable PRF except that each of its PRF keys is associated with a hint and the hint is used in the extraction algorithm to help extract messages. We assume that the extraction algorithm always uses the correct hint when defining the security of a hinting watermarking scheme, i.e., given a (modified) watermarked

PRF key, the hint associated with the PRF key will be employed in the extraction algorithm. Besides, we require its security to hold against an adversary that has the hint associated with the challenge key, yet we only need its pseudorandomness to hold against an adversary without the hint. Next, we describe how to construct public-key hinting watermarkable PRFs.

Construction from Indistinguishability Obfuscation. We first present a general construction of public-key hinting watermarkable PRF from a watermarkable PRF F with secret marking and public extraction. Our main strategy is to generate a fresh mark key/extraction key pair for each PRF key. In this way, there are no global mark keys that should be kept secret. In addition, we set the hint for a PRF key as its extraction key and this allows the extraction key to be used in the extraction algorithm.

In more detail, the PRF key of the public-key hinting watermarkable PRF is $K = (mk, k)$ and the associated hint is $\mathtt{hint} = ek$, where (mk, ek) is a mark key/extraction key pair of F and k is a PRF key of F. Given the PRF key $K = (mk, k)$ and an input x, the evaluation algorithm of the new scheme runs the evaluation algorithm of F on input (k, x), and given $K = (mk, k)$ and a message msg, the marking algorithm of the new scheme runs the marking algorithm of F on input (mk, k, msg). Besides, given a circuit $\mathtt{C}$ and a hint $\mathtt{hint} = ek$, the extraction algorithm runs the extraction algorithm of F on input $(ek, \mathtt{C})$. Security of the constructed public-key hinting watermarkable PRF comes from the assumption that the correct hint is always used and the fact that F is unremovable even if ek is public.

Now, if we instantiate this general construction from previous watermarkable PRFs with public extraction [CHN+16], we obtain public-key hinting watermarkable PRFs from iO. Next, we propose constructions from cryptographic primitives that can be instantiated from standard lattice assumptions, including puncturable PRF, functional encryption, etc.

Mark-Embedding Public-Key Hinting Watermarking from Lattices. First, we consider mark-embedding public-key hinting watermarkable PRFs.

The Starting Point. The starting point of our construction is a watermarking scheme with public marking and *secret extraction* presented in [QWZ18]. The scheme is built on a puncturable PRF [SW14] and a public key encryption (PKE). A puncturable PRF F is a family of PRF that allows one to derive a punctured key k_{x^*} from a PRF key k, where $\mathsf{F}_{k_{x^*}}(\cdot)$ and $\mathsf{F}_k(\cdot)$ evaluate identically on almost all inputs except at the "punctured" point x^*. Its security requires that given the punctured key k_{x^*}, $\mathsf{F}_k(x^*)$ is still pseudorandom.

Here, we slightly modify the scheme and describe it as a hinting watermarking scheme. Its extraction key is a secret key of the PKE scheme. Also, the PRF key $K = k$ is a key of the puncturable PRF F, and the hint is $\mathtt{hint} = (x^*, ct^*)$, where x^* is a random input of F and ct^* is an encryption of $y^* = \mathsf{F}_k(x^*)$. Given a PRF key $K = k$ and an input x, the evaluation algorithm outputs $\mathsf{F}_k(x)$. Also, on input a PRF key $K = k$, the marking algorithm punctures k on x^* and generates a circuit $\mathtt{C}$ s.t. $\mathtt{C}(x) = \mathsf{F}_{k_{x^*}}(x)$. To test if a circuit $\mathtt{C}$ is watermarked, the

extraction algorithm first recovers y^* by decrypting ct^* in the hint and outputs "marked" iff $\mathtt{C}$ is punctured (i.e., $\mathtt{C}(x^*) \neq y^*$).

By security of the puncturable PRF and the PKE scheme, y^* is hidden from an adversary given a watermarked circuit and the hint. Thus, the adversary cannot create a circuit that outputs y^* on input x^* and security of the scheme follows. However, when the extraction key, which is the secret key of the underlying PKE scheme, is made public, the adversary will be able to recover y^* from ct^* and thus compromise security of the scheme.

On Achieving Public Extraction. We solve the problem by designing an extraction algorithm that tests if output of a circuit equals to a given value without knowing the target value. This is achieved by using an injective one way function f. More precisely, in our new scheme, there are no extraction keys and the ciphertext ct^* in the hint is replaced with $z^* = f(y^*)$ (i.e., $\mathtt{hint} = (x^*, z^*)$), where $y^* = \mathsf{F}_k(x^*)$. For a PRF key $K = k$, the evaluation algorithm still outputs $\mathsf{F}_k(x)$ on input x and the marked version of K is still a circuit $\mathtt{C}$ s.t. $\mathtt{C}(x) = \mathsf{F}_{k_{x^*}}(x)$. Besides, to test if a circuit $\mathtt{C}$ is watermarked, the extraction algorithm outputs "marked" iff $z^* \neq f(\mathtt{C}(x^*))$.

The new extraction algorithm actually tests if $\mathtt{C}(x^*)$ is not equal to y^*. Also, security of the one way function plus security of the puncturable PRF guarantee that the adversary cannot learn y^* from a watermarked circuit and the hint. Thus, it is not able to produce a circuit that outputs y^* on input x^*. Therefore, our new construction achieves security in the public extraction setting and thus is a secure public-key hinting watermarkable PRF.

Message-Embedding Public-Key Hinting Watermarking from Lattices. Next, we show how to construct public-key hinting watermarkable PRFs with message embedding from lattices. The construction relies on a functional encryption (FE) scheme [BSW11, O'N10] and is inspired by the construction of watermarkable PKE scheme presented in [GKM+19]. In a nutshell, an FE scheme is a PKE scheme that associates each secret key sk_f with a function f, where the secret keys can be derived from a master secret key. Besides, by using the secret key sk_f to decrypt a ciphertext that encrypts a plaintext m, one can obtain $f(m)$, but nothing else.

From FE to Publicly Verifiable Puncturing. We can use the FE scheme to realize a puncturable "PRF" that supports public verifiability of punctured keys. More precisely, we set the normal PRF key as a secret key sk_{f_ε} of FE, where $f_\varepsilon(t\|\mu) = \mu$. Also, we puncture the key on (inputs that encrypts) plaintexts with prefix t^* by generating a key $sk_{f_{t^*}}$, where $f_{t^*}(t\|\mu) = \mu$ if $t \neq t^*$ and $f_{t^*}(t\|\mu) = 0$ if $t = t^*$. To evaluate the PRF (with either a normal PRF key or a punctured key), the evaluation algorithm just decrypts the input with the secret key. Note that the normal PRF key and the punctured key function identically on an input if it encrypts a plaintext with prefix $t \neq t^*$. In addition, given a punctured key, one could not learn any information about μ from punctured inputs that encrypt $t^*\|\mu$, due to security of the FE scheme. Finally, given the master public key, one can publicly check if a key is punctured on plaintexts with prefix t^* by sampling a random μ, encrypting $t^*\|\mu$, and checking if its decryption is not equal to μ.

From Publicly Verifiable Puncturing to Public-Key Hinting Watermarking. The FE-based puncturable "PRF" with public verifiability implies a public-key hinting watermarkable "PRF" with mark embedding immediately. In particular, the PRF key of the scheme is $K = (msk, sk_{f_\varepsilon})$, where msk is a master secret key of FE and sk_{f_ε} is a secret key derived from msk. The hint for K is the master public key mpk for msk. Given an input x, the evaluation algorithm decrypts x with sk_{f_ε} and outputs the decryption result. The marking algorithm punctures sk_{f_ε} on a public random string t^* and outputs a circuit that decrypts inputs with the punctured key. Given a circuit $\mathtt{C}$, the extraction algorithm outputs "marked" iff the circuit is punctured on plaintexts with prefix t^*. The extraction algorithm can be run publicly with the hint mpk since the underlying puncturable PRF is publicly verifiable. Also, security of the hinting watermarking scheme follows from security of the puncturable PRF directly.

On Supporting Message Embedding. Based on this, we construct hinting watermarking scheme with message embedding by employing the message embedding technique introduced in [GKM+19, YAL+19]. To support this, we define $g_\varepsilon(ind\|t\|\mu) = \mu$ and define

$$g_{msg,t^*}(ind\|t\|\mu) = \begin{cases} 0 & \text{If } t = t^* \wedge\ ind \geq msg \\ \mu & \text{Otherwise} \end{cases}$$

In the message-embedding construction, the PRF key is $K = (msk, sk_{g_\varepsilon})$ and the hint is still the corresponding master public key. The evaluation algorithm decrypts the input with sk_{g_ε}, and to embed a message msg into a PRF key, the marking algorithm generates a circuit that decrypts with the secret key $sk_{g_{msg,t^*}}$. Then, to extract the embedded message from a circuit $\mathtt{C}$, the extraction algorithm will test if the circuit is punctured on prefix $ind\|t^*$ for all possible[3] ind and output msg if it is not punctured on prefix $(msg-1)\|t^*$, but is punctured on prefix $msg\|t^*$.

Now, given a watermarked circuit embedded with a message msg^*, the adversary cannot modify the embedded message since by security of the FE scheme:

1. The adversary cannot distinguish a ciphertext encrypting $ind\|t^*\|\mu$ from a ciphertext that encrypts a random plaintext if $ind < msg^*$. As the adversary is not allowed to change the functionality of the watermarked circuit too much, it cannot puncture on these ciphertexts.
2. The adversary cannot learn μ from a ciphertext encrypting $ind\|t^*\|\mu$ if $ind \geq msg^*$, thus it cannot "unpuncture" the watermarked circuit on these punctured points.

Similarly, we can show that the construction is collusion resistant if the underlying FE is collusion resistant.

On Achieving Pseudorandomness. The above construction actually does not have pseudorandomness. We solve the problem by using a PKE scheme with pseudorandom ciphertexts and a PRF F. In more detail, we add a secret key k of F in

[3] Here, we assume that the message space of the hinting watermarking scheme is of polynomial-size, and this restriction can be removed if we use the jump finding technique introduced in [BCP14, NWZ16].

both the normal PRF key and the marked keys. Then the evaluation algorithm (resp. the marked circuit) will encrypt the output of the evaluation algorithm (resp. the marked circuit) of previous construction with the PKE scheme, where the encryption randomness is $\mathsf{F}_k(x)$. Note that we can put the secret key of the PKE scheme into the hint and thus the extraction algorithm can still test if a given circuit is punctured on plaintexts with a specific prefix. Thus, security of the scheme still holds. In addition, its pseudorandomness is guaranteed by the (ciphertext) pseudorandomness of the underlying PRF and PKE scheme.

On Instantiating the FE Scheme. In above discussion, we implicitly assume that all ciphertexts in the ciphertext space of the FE scheme (i.e., the input space of the hinting watermarkable PRF) can be output by the encryption algorithm. However, to the best of our knowledge, existing FE schemes from standard assumptions [GVW12, GKP+13, AR17, AV19] do not satisfy this property. Even worse, in all of these schemes, the ratio between the number of honestly encrypted ciphertexts and the size of the ciphertext space is exponentially-small. Thus, we have to carefully deal with those "invalid" ciphertexts, which are not output by the encryption algorithm, in the ciphertext space.

First, to ensure that the functionality of a PRF key will not change significantly after watermarking, we need to guarantee that both the normal PRF key and the watermarked PRF key, which use different secret keys of the FE scheme, evaluate identically on input an invalid ciphertext. We achieve this by requiring the FE scheme to have a special correctness, namely, given any secret key and any invalid ciphertext, the decryption result is always a decryption failure symbol $\perp$. We construct FE scheme with this correctness property from any FE scheme with perfect correctness and statistically sound non-interactive zero-knowledge (NIZK) proofs.

Besides, since the ratio ρ between the number of valid ciphertexts and the number of all possible ciphertexts is exponentially-small, the adversary can damage the evaluation on all valid ciphertexts and thus remove the embedded message even if it can only modify the watermarked circuit on a negligible fraction of inputs. We circumvent this problem by requiring that the adversary has to submit a circuit that agrees with the watermarked circuit on a $(1-\rho\cdot(1-1/poly(\lambda)))$ fraction of inputs. Note that even with this restriction, the adversary can still modify the watermarked circuit on exponentially-many inputs.

Remark 2.1. Our FE based construction only allows the adversary to modify the watermarked circuit on an exponentially-small fraction of inputs. Actually, a simple construction from any PRF also satisfies this weak security requirement. In particular, the marking algorithm replaces the PRF outputs with the embedded message if the input has prefix 0^λ, and the extraction algorithm runs the watermarked circuit on random inputs with prefix 0^λ and outputs the majority of the evaluation results. In this construction, the marking algorithm changes the PRF on $1/2^\lambda$ fraction of inputs, and an adversary can remove the watermark only if it changes the watermarked circuit on about $1/2^{\lambda+1}$ fraction of inputs. However, the scheme is less preferable for the following two reasons:

- In this construction, the adversary can remove the watermark by merely changing the circuit on half of the points modified by the marking algorithm. In contrast, the marking algorithm in our construction only changes the output on a negligible fraction of valid ciphertext, and the adversary has to change the outputs on nearly all valid ciphertexts to remove the watermark.
- Our construction will have a good parameter if we use an FE scheme with dense valid ciphertexts in its ciphertext space, but it seems impossible to improve the parameter of the simple construction described above.

We also would like to stress that our goal is to explore the possibility of building full-fledged public-key watermarkable PRFs from standard assumptions rather than constructing a watermarking scheme with weak security guarantee. We demonstrate that the goal is achievable, but our solution has some restrictions. The restrictions can be removed via either using a better FE scheme or improving the proposed construction. We believe our result would inspire future works that completely solve the problem.

2.2 From Public-Key Hinting Watermarkable PRFs to Public-Key Watermarkable PRFs

Next, we discuss how to transform a public-key hinting watermarkable PRF to a public-key watermarkable PRF. Note that a hinting watermarking scheme is already a standard watermarking scheme except that its extraction algorithm needs the correct hint for the given watermarked key. Thus, the main problem here is how to send the correct hint to the extraction algorithm.

To complete this task, we use an unobfuscatable PRF with robust learnability. In a nutshell, in an unobfuscatable PRF UF, each secret key uk_s is embedded with a secret information s. The function $\mathsf{UF}_{uk_s}(\cdot)$ is still pseudorandom if the adversary is only given oracle accesses to it. In addition, one can learn the secret information s given any circuit that implements the function. An unobfuscatable PRF has robust learnability if the secret information s can be learned from any circuit that *approximately* implements $\mathsf{UF}_{uk_s}(\cdot)$, i.e., the circuit may differ from the function on a small fraction of inputs.

Given a public-key hinting watermarkable PRF HF and an unobfuscatable PRF UF, we can construct a public-key watermarkable PRF as follows. The PRF key of the new scheme includes the PRF key k of HF and the PRF key $uk_{\mathtt{hint}}$ of UF, where $\mathtt{hint}$ is the hint for k and is embedded into $uk_{\mathtt{hint}}$ as the secret information. Given an input x, the evaluation algorithm outputs $(\mathsf{HF}_k(x), \mathsf{UF}_{uk_{\mathtt{hint}}}(x))$. To embed a message msg into the PRF key, the marking algorithm first generates k_{msg} by embedding msg to k and then outputs a circuit C s.t. $\mathsf{C}(x) = (\mathsf{HF}_{k_{msg}}(x), \mathsf{UF}_{uk_{\mathtt{hint}}}(x))$. Finally, given a circuit C, the extraction algorithm first recovers $\mathtt{hint}$ from the second part of the circuit and then extracts the message from the first part of the circuit with $\mathtt{hint}$.

Robust learnability of UF ensures that the extracted hint is correct, thus, security of the new scheme comes from the security of the underlying hinting watermarking scheme directly. The above construction also has pseudorandomness for unmarked keys due to the pseudorandomness of the underlying schemes, but

it would not have pseudorandomness for marked keys if the underlying hinting watermarkable PRF does not have this property (recall that we do not require it when defining hinting watermarkable PRFs).

On Achieving Pseudorandomness for Marked Keys. We solve this issue by additionally using a PRF F to mask outputs of HF in both the evaluation algorithm and the marked circuit. The key k' of F is also embedded into the PRF key of UF and this allows the extraction algorithm to obtain k' and use it to unmask outputs of HF. In this way, security of the scheme is preserved. Besides, pseudorandomness of UF guarantees that k' is hidden to an adversary that can only access the marked key in a black-box manner. Then by the pseudorandomness of F and UF, the outputs of the marked key are also pseudorandom.

2.3 Constructing Robust Unobfuscatable PRFs

It remains to show how to construct an unobfuscatable PRF with robust learnability, which is a PRF family UF that allows one to learn the secret information s embedded in a PRF key k_s from any circuit that agrees with $\mathsf{UF}_{k_s}(\cdot)$ on a large fraction of inputs. We first review existing constructions of unobfuscatable functions and explain why they do not lead to a robust unobfuscatable PRF.

The first constructions of unobfuscatable (pseudorandom) functions are presented by Barak et al. in [BGI+01]. Their unobfuscatable PRF also supports learnability from a circuit that approximates the PRF, but it does not allow the circuit to modify the PRF evaluation on particular inputs with a high probability. In contrast, we require that the secret information can be learned from a circuit that may modify the PRF evaluation on any input with probability 1 as long as the fraction of modified inputs is small. Then, in [BP13], Bitansky and Paneth construct an unobfuscatable function with robust learnability. However, the extraction algorithm of the scheme needs a verification key and it should be included in all outputs of the function. Therefore, the scheme cannot be pseudorandom. Recently, Zhandry [Zha21] constructs a robust unobfuscatable function for decryption functionality from an unobfuscatable function without robust learnability and a public-key traitor tracing scheme. It seems that the idea also works for the PRF setting, but this needs a public-key watermarkable PRF, which does not have a candidate construction yet[4].

Next, we describe our constructions of unobfuscatable PRFs with robust learnability. The constructions are inspired by techniques provided in [BGI+01, BP13]. In particular, both our construction and the construction of robust unobfuscatable function given in [BP13] can be viewed as random-self-reducible versions of the non-robust unobfuscatable functions constructed in [BGI+01]. However, as discussed above, the main techniques in [BP13] contradict the requirement of pseudorandomness, and we introduce some new ideas to overcome the difficulties.

[4] Recall that the main goal of this work is to construct the first public-key watermarkable PRF.

Construction from Fully Homomorphic Encryption. The construction needs two PRFs F and F'. Besides, it relies on a special fully homomorphic encryption (FHE) scheme with the following properties[5]:

1. One can homomorphically evaluate a circuit over a ciphertext and rerandomize a ciphertext, without using the public key of the FHE scheme.
2. The ciphertext of the FHE scheme should be pseudorandom.
3. Even given the secret key of the FHE scheme, no one could distinguish a rerandomized ciphertext that encrypts a random plaintext from a random string in the ciphertext space.

The PRF key of the constructed robust unobfuscatable PRF UF is $K = (\alpha, \beta, k, k', pk, sk, s)$, where α, β are random strings, k and k' are PRF keys of F and F' respectively, (pk, sk) is a key pair of the FHE scheme, and s is the secret information. Then, given an input $X = (ind, x, ct)$, the PRF is defined as follows:

$$\mathsf{UF}_K(X) = \begin{cases} \mathtt{Enc}(pk, \alpha; \mathsf{F}'_{k'}(X)) & \textbf{If } \text{ind}=0; \\ \mathsf{F}_k(x \| ct) & \textbf{If } \text{ind}=1; \\ \mathsf{F}_k(x \oplus \alpha \| ct) \oplus \beta & \textbf{If } \text{ind}=2; \\ \mathsf{F}_k(x \oplus \mathtt{Dec}(sk, ct) \oplus \beta \| ct) \oplus s & \textbf{If } \text{ind}=3. \end{cases}$$

where $\mathtt{Enc}$ and $\mathtt{Dec}$ are the encryption algorithm and the decryption algorithm of the underlying FHE scheme respectively.

Robust Learnability of the Construction. We first explain why the above construction has robust learnability. For simplicity, we assume that the extractor is given a circuit $\mathtt{C}$ that agrees with $\mathsf{UF}_K(\cdot)$ on all but negligible fraction of inputs.

The extractor first gets an encryption of α via computing $ct^*_\alpha = \mathtt{C}(0\|x_1\|ct_1)$, where x_1 and ct_1 are random strings. As $\mathtt{C}$ and $\mathsf{UF}_K(\cdot)$ agree on all but negligible fraction of inputs, we have $ct^*_\alpha = \mathsf{UF}_K(0\|x_1\|ct_1) = \mathtt{Enc}(pk, \alpha; \mathsf{F}'_{k'}(0\|x_1\|ct_1))$ with all but negligible probability, i.e., ct^*_α should be an encryption of α.

Then, the extractor obtains an encryption of β as follows. It first computes $y_2 = \mathtt{C}(1\|x_2\|ct_2)$, where x_2 and ct_2 are random strings. Similar, we have $y_2 = \mathsf{F}_k(x_2\|ct_2)$ with all but negligible probability. Next, it runs a circuit $\mathtt{P}(\cdot)$ on ct^*_α to obtain ct^*_β, where for any string a, $\mathtt{P}(a) = \mathtt{C}(2\|x_2 \oplus a\|ct_2) \oplus y_2$. Again, with all but negligible probability, we have $\mathtt{C}(2\|x_2 \oplus \alpha\|ct_2) = \mathsf{UF}_K(2\|x_2 \oplus \alpha\|ct_2) = \mathsf{F}_k(x_2\|ct_2) \oplus \beta$, which implies $\mathtt{P}(\alpha) = \beta$, i.e., ct^*_β is an encryption of β.

Now, with ct^*_α and ct^*_β, the extractor is ready to learn the secret information. It first samples a random γ and computes ct^*_3 as a rerandomized encryption of $\beta \oplus \gamma$. Then it computes $y_3 = \mathtt{C}(3\|x_3\|ct^*_3)$, where x_3 is a random string. Note that ct^*_3 is also random due to Property 3 of the special FHE scheme and the fact that γ is a random string. Thus we have $y_3 = \mathsf{UF}_K(3\|x_3\|ct^*_3) = \mathsf{F}_k(x_3 \oplus \gamma\|ct^*_3) \oplus s$ with all but negligible probability. Next, the extractor computes $y'_3 = \mathtt{C}(1\|x_3 \oplus \gamma\|ct^*_3)$ and recovers $\bar{s} = y_3 \oplus y'_3$. As x_3 is a random string, γ is still hidden given $x_3 \oplus \gamma$, thus $x_3 \oplus \gamma\|ct^*_3$ is indistinguishable from a random string and with all but negligible probability, we have $y'_3 = \mathsf{UF}_K(1\|x_3 \oplus \gamma\|ct^*_3) = \mathsf{F}_k(x_3 \oplus \gamma\|ct^*_3)$,

[5] We show how to construct the desired FHE scheme later in this section.

which implies that $\bar{s} = s$. Therefore, the extractor can succeed in recovering s from the circuit $\mathtt{C}$ with all but negligible probability.

Remark 2.2. The above construction also supports learnability from a circuit that deviates from $\mathsf{UF}_K(\cdot)$ on a constant fraction of inputs. To achieve this, the extractor needs to produce multiple test points in each step and choose the majorities. In more detail, let N be a suitable polynomial. The extractor first produces N ciphertexts ct^*_α via running the circuit $\mathtt{C}$ on N independent inputs (x_1, ct_1). Then for each ct^*_α, it produces N ciphertexts ct^*_β and for each pair $(ct^*_\alpha, ct^*_\beta)$, it computes N results $\bar{s}$. The extractor sets the extraction outputs as the majority-of-majorities-of-majorities. More precisely, for each pair $(ct^*_\alpha, ct^*_\beta)$, it chooses the extracted result for this pair as the majority of all N results $\bar{s}$ produced for this pair. It also chooses the extracted result for each ct^*_α as the majority of all N results for the N pairs $(ct^*_\alpha, ct^*_\beta)$. Finally, it outputs the majority of all N results for the N ciphertexts ct^*_α.

In above extraction procedure, inputs (excluding the index ind) to the circuit $\mathtt{C}$ are all random since they are composed of either random strings or rerandomized ciphertexts encrypting random plaintexts, which are random due to Property 3 of the special FHE scheme. Thus, if the fraction of inputs that $\mathtt{C}$ differs with $\mathsf{UF}_K(\cdot)$ is a small constant δ, the majority result at each step should be the correct secret information. In particular, the extraction result will be correct if

$$\Pr[\mathtt{C}(0\|x_1\|ct_1) = \mathsf{UF}_K(0\|x_1\|ct_1)] > 1/2$$
$$\Pr[\mathtt{C}(1\|x_2\|ct_2) = \mathsf{UF}_K(1\|x_2\|ct_2) \wedge \mathtt{C}(2\|x_2 \oplus \alpha\|ct_2) = \mathsf{UF}_K(2\|x_2 \oplus \alpha\|ct_2)] > 1/2$$
$$\Pr[\mathtt{C}(1\|x_3 \oplus \gamma\|ct^*_3) = \mathsf{UF}_K(1\|x_3 \oplus \gamma\|ct^*_3) \wedge \mathtt{C}(3\|x_3\|ct^*_3) = \mathsf{UF}_K(3\|x_3\|ct^*_3)] > 1/2$$

for random $x_1\|ct_1$, $x_2\|ct_2$, and $x_3\|ct^*_3$, and all three inequalities can be satisfied if $\delta < 1/8$. Besides, the constant δ can be improved to be about $\frac{1}{6}$ if we slightly modify the above construction. Please see Sec. 6.3 for more details.

Pseudorandomness of the Construction. Next, we explain why UF is pseudorandom. We assume w.l.o.g. that all queries submitted by the adversary are distinct.

First, suppose that there are no collisions in the inputs to $\mathsf{F}_k(\cdot)$ when answering queries from the adversary, then outputs of $\mathsf{F}_k(\cdot)$ would be indistinguishable from strings sampled uniformly and independently from its output space, i.e., outputs of $\mathsf{UF}_K(ind\|x\|ct)$ will be pseudorandom if $ind \in \{1, 2, 3\}$. This also implies that the adversary cannot learn any information about sk. Then by ciphertext pseudorandomness of the FHE scheme, outputs of $\mathsf{UF}_K(ind\|x\|ct)$ will also be pseudorandom if $ind = 0$. To summarize, the adversary cannot distinguish UF from a random function if there are no collisions in the inputs to $\mathsf{F}_k(\cdot)$.

Next, we show why the collisions do not occur. In a nutshell, this is because to make a collision, the adversary must have the knowledge of α, β, or encryption of β, and none of them can be obtained via black-box accesses to $\mathsf{UF}_K(\cdot)$. In more detail, assume that there are no collisions in the first q queries to the oracle, then responses of these q queries would be indistinguishable from random strings, which contain no information. Thus, the adversary also cannot make a collision in the $(q+1)$-th query. There is no collision if the adversary only makes

one oracle query, then by the above statement, the adversary cannot make any collision when querying $\mathsf{UF}_K(\cdot)$. Therefore, the pseudorandomness follows.

Construction from One Way Function. Next, we show how to construct robust unobfuscatable PRFs without using FHE. More precisely, the new construction only relies on a standard secret-key encryption scheme with some specific properties, which can be instantiated from any one way function.

Following [BGI+01, BP13], we remove the dependency on homomorphic encryption via performing the homomorphic operations by $\mathsf{UF}_K(\cdot)$. In particular, given an input $X = (ind, x, ct)$, the new PRF proceeds identically as in the construction from FHE if $ind \in \{0, 1, 2, 3\}$. In addition, if $ind = 4$, it decrypts the ciphertext ct, performs the specified homomorphic operation over the decrypted bits and outputs an encryption of the evaluation result, where the randomness is derived from $\mathsf{F}'_{k'}(X)$.

The extractor can use this additional functionality of UF to evaluate $\mathtt{P}$ gate by gate. Thus, it can still succeed in extracting the secret information from a circuit $\mathtt{C}$ that approximates $\mathsf{UF}_K(\cdot)$ even if the underlying encryption scheme does not support homomorphic evaluation over encrypted data.[6]

Constructing Special Fully Homomorphic Encryption. We finally show how to construct the special FHE needed. Our starting point is the GSW homomorphic encryption scheme presented in [GSW13]. In a nutshell, the secret key of the scheme is a random vector $\boldsymbol{s} \in \mathbb{Z}_q^n$. Its public key contains a matrix

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{s}^\intercal \boldsymbol{B} + \boldsymbol{e}^\intercal \end{pmatrix} \mod q$$

and a ciphertext ct^*_{sk}, where $\boldsymbol{B}$ is a random matrix in $\mathbb{Z}_q^{n \times m}$, $\boldsymbol{e}$ is a "short" vector in $\mathbb{Z}^m$ and ct^*_{sk} is an encryption of the secret key $\boldsymbol{s}$. The ciphertext that encrypts a bit μ is defined as[7]

$$\boldsymbol{C} = \mu \cdot \boldsymbol{G} + \boldsymbol{A} \cdot \boldsymbol{R} \mod q$$

where $\boldsymbol{R}$ is a random binary matrix and $\boldsymbol{G}$ is the standard powers-of-two gadget matrix [MP12]. Besides, to rerandomize a ciphertext $\boldsymbol{C}$, the rerandomization algorithm adds the ciphertext with an encryption of 0. Next, we describe how to adapt the construction to achieve the three properties needed.

Achieving Property 1 and Property 2. In the evaluation algorithm of the GSW scheme, the ciphertext ct^*_{sk} should be used to perform the bootstrapping procedure. Also, the rerandomization algorithm needs the matrix $\boldsymbol{A}$ to generate an encryption of 0. Both variables are contained in the public key and thus the first property, which requires that the evaluation algorithm and the rerandomization algorithm can be performed without using the public key, is not satisfied.

[6] We notice that however, the trick presented in Remark 2.2 does not work in this setting as it will require the extractor to produce $N^{O(|\mathtt{C}|)}$ test points, which is exponential in the size of the circuit $\mathtt{C}$. Thus, the construction only supports learnability from a circuit that deviates from $\mathsf{UF}_K(\cdot)$ on a negligible fraction of inputs.

[7] Here, we change the format of the ciphertext of the original GSW scheme slightly.

We solve the problem by putting randomized versions of both variables into the ciphertext of the scheme. In particular, the new ciphertext is (ct_μ, ct_{sk}, ct_0), where ct_μ is an encryption of the message, ct_{sk} is generated by reramdomizing ct^*_{sk} and ct_0 is a fresh encryption of 0. Then we can use ct_{sk} and ct_0 instead of ct^*_{sk} and $\boldsymbol{A}$ when running the evaluation algorithm and the rerandomization algorithm, and Property 1 follows. In addition, as the new ciphertext consists of ciphertexts of the original scheme, the ciphertext pseudorandomness of the modified scheme (i.e., Property 2) comes from that of the original scheme, which can be guaranteed by the circular-secure learning with errors (LWE) assumption.

There is one subtle issue when employing this scheme in the construction of unobfuscatable PRF. That is the extractor can obtain ct_0 only from output of the circuit, which may deviate the PRF evaluation on a 1/6 fraction of inputs, and the obtained ct_0 may not be pseudorandom (e.g., the circuit could rejects to output ct_0 if its first 3 bits are 000). As a result, the output distribution of the rerandomization algorithm may also be changed. We fix the issue by including multiple ct_0 in each ciphertext and use a random subset sum of them in the rerandomization algorithm. The selection of the subset provides additional entropy and we can show that the result is pseudorandom by using the leftover hash lemma and the fact that $\boldsymbol{A}$ is pseudorandom.

Achieving Property 3. The third property of the special FHE scheme requires that a rerandomized ciphertext of a random plaintext should look uniform even given the secret key of the FHE scheme. Here we relax this property and only require that one can transform a rerandomized ciphertext of the FHE scheme into a ciphertext with this strong uniformity. The transformed ciphertext is still decryptable, but does not have to support homomorphic evaluation over it. Note that this relaxed property is sufficient in our construction of unobfuscatable PRF.

Given a ciphertext $CT = (\boldsymbol{C}, ct_{sk}, ct_0)$, where $\boldsymbol{C}$ encrypts a bit μ, we first transform it as:

$$\boldsymbol{c} = \begin{pmatrix} \boldsymbol{0} \\ \mu \cdot \frac{q+1}{2} \end{pmatrix} + \boldsymbol{A}\boldsymbol{r} \mod q$$

where $\boldsymbol{r}$ is a short vector in $\mathbb{Z}_q^m$. We can obtain $\boldsymbol{c}$ via summing some columns of $\boldsymbol{C}$. In addition, to decrypt the ciphertext, one can first compute

$$(-\boldsymbol{s}^\intercal, 1) \cdot \boldsymbol{c} = \mu \cdot \frac{q+1}{2} + \boldsymbol{e}^\intercal \cdot \boldsymbol{r} \mod q \tag{1}$$

where $\boldsymbol{e}$ is the short error term in $\boldsymbol{A}$. Then the decryption result will be 1 if Equation (1) is close to $\frac{q+1}{2}$ and it will be 0 if Equation (1) is close to 0.

However, the above transformed ciphertext can be distinguished from a random vector given $\boldsymbol{s}$ due to the following decryption attack. Given a ciphertext $\boldsymbol{c}$, which is either a transformed ciphertext or a random vector, the distinguisher with the secret key $\boldsymbol{s}$ first computes Equation (1). It will get a number that is close to $\frac{q+1}{2}$ or 0 if $\boldsymbol{c}$ is a transformed ciphertext and it will get a random number in $\mathbb{Z}_q$ if $\boldsymbol{c}$ is a random vector. Thus, it could distinguish these two cases.

We prevent the attack via adding a number $z \xleftarrow{\$} [0, \frac{q-1}{2}]$ to the last element of the transformed ciphertext and require that q is much larger than the error

term $\boldsymbol{e}^\intercal \cdot \boldsymbol{r}$. One will get $\mu \cdot \frac{q+1}{2} + \boldsymbol{e}^\intercal \cdot \boldsymbol{r} + z$ via computing Equation (1) on a transformed ciphertext that encrypts μ, and this will be a random number in $[\mu \cdot \frac{q+1}{2}, \mu \cdot \frac{q+1}{2} + \frac{q-1}{2}]$ due to the smudging lemma [AJLA+12], which states that a small error (i.e., $\boldsymbol{e}^\intercal \cdot \boldsymbol{r}$) can be smudged out by a large error (i.e., z). The encrypted message can still be recovered from $\boldsymbol{c}$ via computing Equation (1) and checking if the result exceeds $\frac{q-1}{2}$. Besides, if $\boldsymbol{c}$ is a transformed ciphertext that encrypts a random bit, then Equation (1) would also be a random number in $\mathbb{Z}_q$ and thus the distinguisher cannot distinguish it from a random vector.

3 Notations

We write $negl(\cdot)$ to denote a negligible function and write $poly(\cdot)$ to denote a polynomial. For integers $a \leq b$, we write $[a, b]$ to denote all integers from a to b. Let s be a string, we use $|s|$ to denote the length of s. For integers $a \leq |s|$, $s[a]$ denotes the a-th character of s and for integers $a \leq b \leq |s|$, $s[a : b]$ denotes the substring $(s[a], s[a + 1], \ldots, s[b])$. Let $\mathcal{S}$ be a finite set, we use $|\mathcal{S}|$ to denote the size of $\mathcal{S}$, and use $s \xleftarrow{\$} \mathcal{S}$ to denote sampling an element s uniformly from set $\mathcal{S}$. Let $\mathcal{D}$ be a distribution, we use $d \leftarrow \mathcal{D}$ to denote sampling d according to $\mathcal{D}$. Following the syntax in [BLW17], for a circuit family $\mathtt{C}$ indexed by a few, say m, constants, we write $\mathtt{C}[c_1, \ldots, c_m]$ to denote a circuit with constants $c_1, \ldots, c_m$. We use $\overline{\wedge}$ to denote the NAND gate and suppose that all circuits are composed exclusively by NAND gates unless otherwise specified. We use bold lower-case letters to denote vectors, and use bold upper-case letters to denote matrices. All elements in vectors and matrices are integers unless otherwise specified. Let $\boldsymbol{v}$ be a vector of length n, $\boldsymbol{v}[i]$ denotes the i-th element of $\boldsymbol{v}$ for $i \in [1, n]$ and $\boldsymbol{v}[i : j]$ denotes the vector $(\boldsymbol{v}[i], \boldsymbol{v}[i + 1], \ldots, \boldsymbol{v}[j])^\intercal$ for $1 \leq i < j \leq n$. For an m-by-n matrix $\boldsymbol{A}$, $\boldsymbol{A}[i, j]$ denotes the element on the i-th row and the j-th columon of $\boldsymbol{A}$ for $i \in [1, m]$ and $j \in [1, n]$. We provide more background knowledge and recall definitions of cryptographic primitives employed in this work in Appendix B.

4 Definition of Public-Key Watermarkable PRFs

In this section, we provide the definition of public-key watermarkable PRFs, which is adapted from definitions of watermarkable PRFs in previous works [CHN+16, BLW17, KW17, QWZ18, KW19, YAL+19, YAYX20]. More precisely, a public-key watermarkable PRF with key space $\mathcal{K}$, input space $\mathcal{X}$, output space $\mathcal{Y}$, and message space $\mathcal{M}$ consists of the following algorithms:

- $\mathtt{Setup}(1^\lambda) \to PP$: On input the security parameter 1^λ, the setup algorithm outputs the public parameter PP.
- $\mathtt{KeyGen}(PP) \to k$: On input the public parameter PP, the key generation algorithm outputs a PRF key $k \in \mathcal{K}$.
- $\mathtt{Eval}(PP, k, x) \to y$: On input the public parameter PP, a PRF key $k \in \mathcal{K}$, and an input $x \in \mathcal{X}$, the evaluation algorithm outputs an output $y \in \mathcal{Y}$.

- $\texttt{Mark}(PP, k, msg) \rightarrow \texttt{C}$: On input the public parameter PP, a PRF key $k \in \mathcal{K}$, and a message $msg \in \mathcal{M}$, the marking algorithm outputs a marked circuit $\texttt{C} : \mathcal{X} \rightarrow \mathcal{Y}$.
- $\texttt{Extract}(PP, \texttt{C}) \rightarrow msg$: On input the public parameter PP and a circuit $\texttt{C}$, the extraction algorithm outputs a message $msg \in \mathcal{M} \cup \{\perp\}$, where $\perp$ denotes that the circuit is unmarked.

Correctness. The correctness of a watermarking scheme includes three properties. The functionality preserving property requires that the watermarked key can roughly preserve the functionality of the original key.

Definition 4.1 (Functionality Preserving). *For any $msg \in \mathcal{M}$, let $PP \leftarrow \texttt{Setup}(1^\lambda)$, $k \leftarrow \texttt{KeyGen}(PP)$, $\texttt{C} \leftarrow \texttt{Mark}(PP, k, msg)$, $x \xleftarrow{\$} \mathcal{X}$, then we have $\Pr[\texttt{C}(x) \neq \texttt{Eval}(PP, k, x)] \leq negl(\lambda)$.*

The extraction correctness requires that the extraction algorithm can extract the correct message from an honestly-watermarked key and will obtain the "unmarked" symbol when extracting an unmarked key.

Definition 4.2 (Extraction Correctness). *For any $msg \in \mathcal{M}$, let $PP \leftarrow \texttt{Setup}(1^\lambda)$, $k \leftarrow \texttt{KeyGen}(PP)$, and $\texttt{C} \leftarrow \texttt{Mark}(PP, k, msg)$, then we have*

$$\Pr[\texttt{Extract}(PP, \texttt{C}) \neq msg] \leq negl(\lambda)$$

$$\Pr[\texttt{Extract}(PP, \texttt{Eval}(PP, k, \cdot)) \neq \perp] \leq negl(\lambda)$$

The meaningfulness property requires that most circuits are unmarked, which rules out the trivial construction that regards all circuits as marked.

Definition 4.3 (Watermarking Meaningfulness). *For any circuit $\texttt{C} : \mathcal{X} \rightarrow \mathcal{Y}$, let $PP \leftarrow \texttt{Setup}(1^\lambda)$, then we have $\Pr[\texttt{Extract}(PP, \texttt{C}) \neq \perp] \leq negl(\lambda)$.*

Pseudorandomness. Our definition of pseudorandomness is twofold, including pseudorandomness for unmarked keys and that for marked keys. The properties require that given oracle access to an unmarked PRF key (or a marked key), the adversary cannot distinguish it from a random function.

Definition 4.4 (Pseudorandomness for Unmarked Keys). *Let $PP \leftarrow \texttt{Setup}(1^\lambda)$, $k \leftarrow \texttt{KeyGen}(PP)$, and f be a random function from $\mathcal{X}$ to $\mathcal{Y}$. Also, let $\mathcal{O}_0(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $\texttt{Eval}(PP, k, x)$, and let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $f(x)$. Then for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, we have:*

$$|\Pr[\mathcal{A}^{\mathcal{O}_0(\cdot)}(PP) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(PP) = 1]| \leq negl(\lambda)$$

Definition 4.5 (Pseudorandomness for Marked Keys). *For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, let $PP \leftarrow \texttt{Setup}(1^\lambda)$ and $k \leftarrow \texttt{KeyGen}(PP)$. Also, let $(msg, state) \leftarrow \mathcal{A}_1(PP)$, $\texttt{C} \leftarrow \texttt{Mark}(PP, k, msg)$, and f be a random function from $\mathcal{X}$ to $\mathcal{Y}$. Let $\mathcal{O}_0(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $\texttt{C}(x)$, and let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $f(x)$. Then we have:*

$$|\Pr[\mathcal{A}_2^{\mathcal{O}_0(\cdot)}(state) = 1] - \Pr[\mathcal{A}_2^{\mathcal{O}_1(\cdot)}(state) = 1]| \leq negl(\lambda)$$

Unremovability. This is the main security requirement for a watermarking scheme, which requires that the adversary cannot remove or modify the messages embedded in a random PRF key without significantly changing its functionality.

Definition 4.6 (Q-Bounded ϵ-Unremovability). *A watermarkable PRF is Q-bounded ϵ-unremovable if for all PPT and ϵ-unremoving-admissible adversaries $\mathcal{A}$, we have $\Pr[\mathtt{ExptUR}_{\mathcal{A},Q}(\lambda) = 1] \leq negl(\lambda)$, where we define the experiment $\mathtt{ExptUR}$ as follows:*

1. *The challenger samples $PP \leftarrow \mathtt{Setup}(1^\lambda)$ and $k^* \leftarrow \mathtt{KeyGen}(PP)$.*
2. *Then, it returns PP to $\mathcal{A}$ and answers $\mathcal{A}$'s challenge oracle queries. Here, $\mathcal{A}$ is only allowed to query the challenge oracle for at most Q times.*
   - ***Challenge Oracle.** On input a message $msg \in \mathcal{M}$, the challenge oracle returns a circuit $\mathtt{C}^* \leftarrow \mathtt{Mark}(PP, k^*, msg)$ to the adversary.*
3. *Finally, $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$ and the experiment outputs 1 iff $\mathtt{Extract}(PP, \tilde{\mathtt{C}}) \notin \mathcal{Q}^*$. Here, we use $\mathcal{Q}^*$ to denote all messages submitted to the challenge oracle and use $\mathcal{R}^*$ to denote all circuits returned by the challenge oracle.*

We say that an adversary $\mathcal{A}$ is ϵ-unremoving-admissible *if there exists circuit $\mathtt{C}^* \in \mathcal{R}^*$ that $|\{x \in \mathcal{X} : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq \epsilon \cdot |\mathcal{X}|$.*

Remark 4.1. We can also define $negl(\lambda)$-unremovability for a watermarkable PRF, which is identical to the definition of ϵ-unremovability for concrete ϵ, except that $\mathcal{A}$ should be $negl(\lambda)$-unremoving-admissible, i.e., there exists circuit $\mathtt{C}^* \in \mathcal{R}^*$ that $|\{x \in \mathcal{X} : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq negl(\lambda) \cdot |\mathcal{X}|$.

5 Public-Key Hinting Watermarkable PRFs

We define and construct public-key hinting watermarkable PRFs in this section. We provide its formal definition in Sec. 5.1. Then in Sec. 5.2, Sec. 5.3, and Sec. 5.4, we give constructions with different properties.

5.1 The Definition

The definition of public-key hinting watermarkable PRF is similar to the definition of standard public-key watermarkable PRFs given in Sec. 4 except that its key generation algorithm will generate a *"hint"* together with the PRF key, which can be used later in the extraction algorithm. More precisely, a public-key hinting watermarkable PRF with key space $\mathcal{K}$, input space $\mathcal{X}$, output space $\mathcal{Y}$, and message space $\mathcal{M}$ consists of the following algorithms:

- $\mathtt{Setup}(1^\lambda) \to PP$: On input the security parameter 1^λ, the setup algorithm outputs the public parameter PP.
- $\mathtt{KeyGen}(PP) \to (k, \mathtt{hint})$: On input the public parameter PP, the key generation algorithm outputs a PRF key $k \in \mathcal{K}$ and a hint $\mathtt{hint}$.
- $\mathtt{Eval}(PP, k, x) \to y$: On input the public parameter PP, a PRF key $k \in \mathcal{K}$, and an input $x \in \mathcal{X}$, the evaluation algorithm outputs an output $y \in \mathcal{Y}$.

- $\texttt{Mark}(PP, k, msg) \rightarrow \texttt{C}$: On input the public parameter PP, a PRF key $k \in \mathcal{K}$, and a message $msg \in \mathcal{M}$, the marking algorithm outputs a marked circuit $\texttt{C} : \mathcal{X} \rightarrow \mathcal{Y}$.
- $\texttt{Extract}(PP, \texttt{C}, \texttt{hint}) \rightarrow msg$: On input the public parameter PP, a circuit $\texttt{C}$, and a hint $\texttt{hint}$, the extraction algorithm outputs a message $msg \in \mathcal{M} \cup \{\perp\}$, where $\perp$ denotes that the circuit is unmarked.

Correctness. The correctness of a public-key hinting watermarkable PRF also requires the following three properties. Here for the extraction correctness, we require that the *correct hint* is used.

- **Functionality Preserving.** For any $msg \in \mathcal{M}$, let $PP \leftarrow \texttt{Setup}(1^\lambda)$, $(k, \texttt{hint}) \leftarrow \texttt{KeyGen}(PP)$, $\texttt{C} \leftarrow \texttt{Mark}(PP, k, msg)$, $x \xleftarrow{\$} \mathcal{X}$, then we have $\Pr[\texttt{C}(x) \neq \texttt{Eval}(PP, k, x)] \leq negl(\lambda)$.
- **Extraction Correctness.** For any $msg \in \mathcal{M}$, let $PP \leftarrow \texttt{Setup}(1^\lambda)$, $(k, \texttt{hint}) \leftarrow \texttt{KeyGen}(PP)$, and $\texttt{C} \leftarrow \texttt{Mark}(PP, k, msg)$, then we have

$$\Pr[\texttt{Extract}(PP, \texttt{C}, \texttt{hint}) \neq msg] \leq negl(\lambda)$$

$$\Pr[\texttt{Extract}(PP, \texttt{Eval}(PP, k, \cdot), \texttt{hint}) \neq \perp] \leq negl(\lambda)$$

- **Watermarking Meaningfulness.** For any circuit $\texttt{C} : \mathcal{X} \rightarrow \mathcal{Y}$ and any $\texttt{hint}$, let $PP \leftarrow \texttt{Setup}(1^\lambda)$, then we have $\Pr[\texttt{Extract}(PP, \texttt{C}, \texttt{hint}) \neq \perp] \leq negl(\lambda)$.

Pseudorandomness. The pseudorandomness property requires that the evaluation of the PRF with an unmarked key should be pseudorandom. Here, the adversary is *not* allowed to access the hint associated with the PRF key.

Definition 5.1 (Pseudorandomness). *Let* $PP \leftarrow \texttt{Setup}(1^\lambda)$, $(k, \texttt{hint}) \leftarrow \texttt{KeyGen}(PP)$, *and* f *be a random function from* $\mathcal{X}$ *to* $\mathcal{Y}$. *Also, let* $\mathcal{O}_0(\cdot)$ *be an oracle that takes as input a string* $x \in \mathcal{X}$ *and returns* $\texttt{Eval}(PP, k, x)$, *and let* $\mathcal{O}_1(\cdot)$ *be an oracle that takes as input a string* $x \in \mathcal{X}$ *and returns* $f(x)$. *Then for all PPT adversary* $\mathcal{A}$, *we have:*

$$|\Pr[\mathcal{A}^{\mathcal{O}_0(\cdot)}(PP) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(PP) = 1]| \leq negl(\lambda)$$

Unremovability. The unremovability property also requires that an adversary cannot remove or modify the message embedded in a watermarked PRF key while keeping its functionality. Here, we allow the adversary to learn the *hint* associated with the PRF key. Also, we require that the *correct hint* should be used when extracting the circuit submitted by the adversary.

Definition 5.2 (Q-Bounded ϵ-Unremovability). *A hinting watermarkable PRF is* Q*-bounded* ϵ*-unremovable if for all PPT and* ϵ*-unremoving-admissible adversaries* $\mathcal{A}$, *we have* $\Pr[\texttt{ExptUR}_{\mathcal{A},Q}(\lambda) = 1] \leq negl(\lambda)$, *where we define the experiment* $\texttt{ExptUR}$ *as follows:*

1. *The challenger samples* $PP \leftarrow \texttt{Setup}(1^\lambda)$ *and* $(k^*, \texttt{hint}^*) \leftarrow \texttt{KeyGen}(PP)$.
2. *Then, it returns* $(PP, \texttt{hint}^*)$ *to* $\mathcal{A}$ *and answers* $\mathcal{A}$*'s challenge oracle queries. Here,* $\mathcal{A}$ *is only allowed to query the challenge oracle for at most* Q *times.*

- ***Challenge Oracle.*** *On input a message $msg \in \mathcal{M}$, the challenge oracle returns a circuit $\mathtt{C}^* \leftarrow \mathtt{Mark}(PP, k^*, msg)$ to the adversary.*

3. *Finally, $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$ and the experiment outputs 1 iff $\mathtt{Extract}(PP, \tilde{\mathtt{C}}, \mathtt{hint}^*) \notin \mathcal{Q}^*$. Here, we use $\mathcal{Q}^*$ to denote all messages submitted to the challenge oracle and use $\mathcal{R}^*$ to denote all circuits returned by the challenge oracle.*

We say that an adversary $\mathcal{A}$ is ϵ-unremoving-admissible *if there exists circuit $\mathtt{C}^* \in \mathcal{R}^*$ that $|\{x \in \mathcal{X} : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq \epsilon \cdot |\mathcal{X}|$.*[8]

5.2 Public-Key Hinting Watermarkable PRFs from Puncturable PRFs

In this section, we present our construction of public-key hinting watermarkable PRFs from puncturable PRFs and injective one way functions.

Let λ be the security parameter. Let n, m, l be positive integers that are polynomial in λ.

Our construction is built on the following building blocks:

- A puncturable PRF $\mathsf{PPRF} = (\mathsf{PPRF}.\mathtt{KeyGen}, \mathsf{PPRF}.\mathtt{Eval}, \mathsf{PPRF}.\mathtt{Constrain}, \mathsf{PPRF}.\mathtt{ConstrainEval})$ with input space $\{0,1\}^n$ and output space $\{0,1\}^m$.
- An injective one way function $\mathtt{F} : \{0,1\}^m \to \{0,1\}^l$.

We construct the public-key hinting watermarkable PRF $\mathsf{HWF} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Eval}, \mathtt{Mark}, \mathtt{Extract})$, which has input space $\{0,1\}^n$, output space $\{0,1\}^m$ and message space $\{\text{“}\mathtt{marked}\text{”}\}$ as follows:

- $\mathtt{Setup}$. On input the security parameter 1^λ, the setup algorithm samples $w \xleftarrow{\$} \{0,1\}^\lambda$ and outputs the public parameter $PP = w$.
- $\mathtt{KeyGen}$. On input the public parameter $PP = w$, the key generation algorithm computes:
  1. $k \leftarrow \mathsf{PPRF}.\mathtt{KeyGen}(1^\lambda)$.
  2. $x^* \xleftarrow{\$} \{0,1\}^n$.
  3. $y^* = \mathsf{PPRF}.\mathtt{Eval}(k, x^*)$.
  4. $z^* = \mathtt{F}(y^*)$.

  and outputs the PRF key $K = (k, x^*)$ and the hint $\mathtt{hint} = (x^*, z^*, w)$.
- $\mathtt{Eval}$. On input the public parameter $PP = w$, the PRF key $K = (k, x^*)$ and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs $y = \mathsf{PPRF}.\mathtt{Eval}(k, x)$.
- $\mathtt{Mark}$. On input the public parameter $PP = w$ and the PRF key $K = (k, x^*)$, the marking algorithm computes $ck \leftarrow \mathsf{PPRF}.\mathtt{Constrain}(k, x^*)$ and outputs a circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ s.t. for any $x \in \{0,1\}^n$:

$$\mathtt{C}(x) = \mathsf{PPRF}.\mathtt{ConstrainEval}(ck, x)$$

[8] Similar to a standard watermarkable PRF, we can define $negl(\lambda)$-unremovability, which requires that $\exists \mathtt{C}^* \in \mathcal{R}^*$ s.t. $|\{x \in \mathcal{X} : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq negl(\lambda) \cdot |\mathcal{X}|$.

- $\mathtt{Extract}$. On input the public parameter $PP = w$, a circuit $\mathtt{C}$, and a hint $\mathtt{hint} = (\bar{x}^*, \bar{z}^*, \bar{w})$, the extraction algorithm outputs "$\mathtt{marked}$" if

$$\bar{w} = w \ \wedge \ \bar{z}^* \neq \mathtt{F}(\mathtt{C}(\bar{x}^*))$$

  Otherwise, it outputs $\perp$.

Theorem 5.1. *If* PPRF *is a secure puncturable PRF and* $\mathtt{F}$ *is an injective one way function, then* HWF *is a secure public-key hinting watermarkable PRF with* 1*-bounded* 1*-unremovability.*

We present proof of Theorem 5.1 in Appendix E.1.

5.3 Public-Key Hinting Watermarkable PRFs from Functional Encryption

In this section, we construct public-key hinting watermarkable PRFs from functional encryption. Our construction relies on an FE scheme with additional properties, which is defined and constructed in Appendix C.

Let λ be the security parameter. Let n, m, κ, Q be positive integers that are polynomial in λ. Let ρ, θ be real values in $(0, 1)$ s.t. $1/\theta$ is polynomial in λ. Let $\varphi = \theta/(5 + (\kappa - 1)Q)$. Let $T = \lambda/\varphi^2$. Let $\epsilon = \rho \cdot (1 - \theta)$.

Also, for any $(msg, t^*) \in [0, 2^\kappa - 1] \times \{0,1\}^\lambda$, we define the following functions from $[0, 2^\kappa - 1] \times \{0,1\}^\lambda \times \{0,1\}^\lambda$ to $\{0,1\}^\lambda$:

$$f_\perp(ind \| t \| \mu) = \mu$$

$$f_{msg,t^*}(ind \| t \| \mu) = \begin{cases} 0^\lambda & \text{If } t = t^* \wedge \ ind \geq msg \\ \mu & \text{Otherwise} \end{cases}$$

Our construction is built on the following building blocks:

- An FE scheme $\mathsf{FE} = (\mathsf{FE}.\mathtt{Setup}, \mathsf{FE}.\mathtt{KeyGen}, \mathsf{FE}.\mathtt{Enc}, \mathsf{FE}.\mathtt{Dec})$ with message space $\{0,1\}^{\kappa+2\lambda}$, ciphertext space $\{0,1\}^n$ and density ρ.[9] In addition, we assume w.l.o.g. that $\mathsf{FE}.\mathtt{Dec}$ is a deterministic algorithm (See Remark C.2).
- A PKE scheme $\mathsf{PKE} = (\mathsf{PKE}.\mathtt{KeyGen}, \mathsf{PKE}.\mathtt{Enc}, \mathsf{PKE}.\mathtt{Dec})$ with message space $\{0,1\}^\lambda$ and ciphertext space $\{0,1\}^m$. Also, we use $\mathcal{R}_{\mathsf{PKE}.\mathtt{Enc}}$ to denote the randomness space for the algorithm $\mathsf{PKE}.\mathtt{Enc}$.
- A PRF $\mathsf{F} = (\mathsf{F}.\mathtt{KeyGen}, \mathsf{F}.\mathtt{Eval})$ with input space $\{0,1\}^n$ and output space $\mathcal{R}_{\mathsf{PKE}.\mathtt{Enc}}$.

We construct the public-key hinting watermarkable PRF $\mathsf{HWF} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Eval}, \mathtt{Mark}, \mathtt{Extract})$, which has input space $\{0,1\}^n$, output space $\{0,1\}^m$ and message space $\{0,1\}^\kappa \backslash \{0^\kappa\} = [1, 2^\kappa - 1]$ as follows:

[9] We use density to denote the fraction of honestly generated ciphertexts in the ciphertext space. Its formal definition is given in Remark C.1.

- $\texttt{Setup}$. On input the security parameter 1^λ, the setup algorithm samples $w \xleftarrow{\$} \{0,1\}^\lambda$, $t^* \xleftarrow{\$} \{0,1\}^\lambda$, and outputs the public parameter $PP = (w, t^*)$.
- $\texttt{KeyGen}$. On input the public parameter $PP = (w, t^*)$, the key generation algorithm computes
  1. $(mpk, msk) \leftarrow \mathsf{FE}.\texttt{Setup}(1^\lambda)$.
  2. $(pk, sk) \leftarrow \mathsf{PKE}.\texttt{KeyGen}(1^\lambda)$.
  3. $k \leftarrow \mathsf{F}.\texttt{KeyGen}(1^\lambda)$.
  4. $fsk \leftarrow \mathsf{FE}.\texttt{KeyGen}(mpk, msk, f_\perp)$.

  and outputs the PRF key $K = (mpk, msk, pk, k, fsk)$ and the hint $\texttt{hint} = (mpk, sk, w)$.
- $\texttt{Eval}$. On input the public parameter $PP = (w, t^*)$, the PRF key $K = (mpk, msk, pk, k, fsk)$, and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs $\texttt{M}[mpk, fsk, pk, k](x)$, where $\texttt{M}$ is defined in Figure 2.
- $\texttt{Mark}$. On input the public parameter $PP = (w, t^*)$, the PRF key $K = (mpk, msk, pk, k, fsk)$, and a message $msg \in [1, 2^\kappa - 1]$, the marking algorithm computes $fsk_{msg} \leftarrow \mathsf{FE}.\texttt{KeyGen}(mpk, msk, f_{msg,t^*})$ and outputs the circuit $\texttt{M}[mpk, fsk_{msg}, pk, k]$, where $\texttt{M}$ is defined in Figure 2.
- $\texttt{Extract}$. On input the public parameter $PP = (w, t^*)$, a circuit $\texttt{C}$, and a hint $\texttt{hint} = (\overline{mpk}, \overline{sk}, \bar{w})$, the extraction algorithm **output** $\perp$ if $\bar{w} \neq w$. Otherwise, it runs the jump finding algorithm $\texttt{Trace}$ (recalled in Appendix B.5 and also described in Figure 3) to extract messages from $\texttt{C}$, where the oracle $\texttt{P}$ used in the algorithm is simulated by the $\texttt{Test}$ algorithm defined in Figure 3. More precisely, it proceeds as follow:
  1. Set the constant for the algorithm $\texttt{Test}$ as $(\texttt{C}, \overline{mpk}, \overline{sk}, t^*)$.
  2. $p_0 = \texttt{Test}(0)$.
  3. $p_{2^\kappa - 1} = \texttt{Test}(2^\kappa - 1)$.
  4. $\mathcal{M} \leftarrow \texttt{Trace}(0, 2^\kappa - 1, p_0, p_{2^\kappa - 1})$. Here, the extraction algorithm will abort and **output** $\perp$ if the $\texttt{Test}$ algorithm has been invoked for more than $Q \cdot (\kappa + 1)$ times in the $\texttt{Trace}$ algorithm.
  5. If $\mathcal{M} = \emptyset$, **output** $\perp$.
  6. $msg \xleftarrow{\$} \mathcal{M}$.
  7. **Output** msg.

$\texttt{M}$
Constant: mpk, fsk, pk, k
Input: x
1. $\mu = \mathsf{FE}.\texttt{Dec}(mpk, fsk, x)$.
2. If $\mu = \perp$, set $\mu = 0^\lambda$.
3. Output $y = \mathsf{PKE}.\texttt{Enc}(pk, \mu; \mathsf{F}.\texttt{Eval}(k, x))$.

Fig. 2 The circuit $\texttt{M}$.

Theorem 5.2. *If* FE *is a secure functional encryption scheme with* Q*-adaptive indistinguishability and strong correctness (as defined in Appendix C),* PKE *is a secure PKE scheme with ciphertext pseudorandomness, and* F *is a secure PRF, then* HWF *is a secure public-key hinting watermarkable PRF with* Q*-bounded* ϵ*-unremovability.*

Trace	Test
Input: ind_1, ind_2, p_1, p_2 1. $\Delta = \lvert p_1 - p_2 \rvert$. 2. If $\Delta \leq \varphi$: **Return** $\emptyset$. 3. If $ind_2 - ind_1 = 1$: **Return** $\{ind_2\}$. 4. $ind_3 = \lfloor \frac{ind_1 + ind_2}{2} \rfloor$. 5. $p_3 = \mathtt{Test}(ind_3)$. 6. **Return** $\mathtt{Trace}(ind_1, ind_3, p_1, p_3) \cup \mathtt{Trace}(ind_3, ind_2, p_3, p_2)$.	**Constant:** $\mathtt{E}, mpk, sk, t^*$ **Input:** ind 1. $Acc = 0$ 2. For $i \in [1, T]$: (a) Sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$. (b) $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind \Vert t^* \Vert \mu)$. (c) $y = \mathtt{E}(x)$. (d) $\bar{\mu} = \mathsf{PKE}.\mathtt{Dec}(sk, y)$. (e) If $\mu = \bar{\mu}$: $Acc = Acc + 1$. 3. Return $\frac{Acc}{T}$.

Fig. 3 The algorithms `Trace` and `Test`.

We present proof of Theorem 5.2 in Appendix E.2.

5.4 Public-Key Hinting Watermarkable PRFs from Secret-Key Watermarkable PRFs

In this section, we show how to construct public-key hinting watermarkable PRFs from any secret-key watermarkable PRF with public extraction.

Let λ be the security parameter. Let m, n, κ be positive integers that are polynomial in λ. Let $\mathsf{SK\text{-}WPRF} = (\mathsf{SK\text{-}WPRF}.\mathtt{Setup}, \mathsf{SK\text{-}WPRF}.\mathtt{KeyGen}, \mathsf{SK\text{-}WPRF}.\mathtt{Eval}, \mathsf{SK\text{-}WPRF}.\mathtt{Mark}, \mathsf{SK\text{-}WPRF}.\mathtt{Extract})$ be a secret-key watermarkable PRF with *public extraction* (see Appendix B.6 for its definition) that has input space $\{0,1\}^n$, output space $\{0,1\}^m$, and message space $\{0,1\}^\kappa$.

We construct the public-key hinting watermarkable PRF $\mathsf{HWF} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Eval}, \mathtt{Mark}, \mathtt{Extract})$, which has input space $\{0,1\}^n$, output space $\{0,1\}^m$, and message space $\{0,1\}^\kappa$ as follows:

- $\mathtt{Setup}$. On input the security parameter 1^λ, the setup algorithm samples $w \xleftarrow{\$} \{0,1\}^\lambda$ and outputs the public parameter $PP = w$.
- $\mathtt{KeyGen}$. On input the public parameter $PP = w$, the key generation algorithm first generates $(pp, mk, ek) \leftarrow \mathsf{SK\text{-}WPRF}.\mathtt{Setup}(1^\lambda)$ and $k \leftarrow \mathsf{SK\text{-}WPRF}.\mathtt{KeyGen}(pp)$. Then, it outputs the PRF key $K = (pp, mk, ek, k)$ and the hint $\mathtt{hint} = (pp, ek, w)$.
- $\mathtt{Eval}$. On input the public parameter $PP = w$, the PRF key $K = (pp, mk, ek, k)$ and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs $y = \mathsf{SK\text{-}WPRF}.\mathtt{Eval}(pp, k, x)$.
- $\mathtt{Mark}$. On input the public parameter $PP = w$, the PRF key $K = (pp, mk, ek, k)$ and a message $msg \in \{0,1\}^\kappa$, the marking algorithm outputs $\mathtt{C} \leftarrow \mathsf{SK\text{-}WPRF}.\mathtt{Mark}(pp, mk, k, msg)$.
- $\mathtt{Extract}$. On input the public parameter $PP = w$, a circuit $\mathtt{C}$ and a hint $\mathtt{hint} = (pp, ek, w')$, the extraction algorithm outputs $\perp$ if $w \neq w'$. Otherwise, it outputs $msg \leftarrow \mathsf{SK\text{-}WPRF}.\mathtt{Extract}(pp, ek, \mathtt{C})$.

Theorem 5.3. *If* $\mathsf{SK\text{-}WPRF}$ *is a secure secret-key watermarkable PRF with public extraction and* Q*-bounded* ϵ*-unremovability, then* HWF *is a secure public-key*

hinting watermarkable PRF with Q-bounded ϵ-unremovability, where ϵ is a real value in $(0,1)$ (or the negligible function $negl(\lambda)$) and Q is a positive integer that is polynomial in λ.

We present proof of Theorem 5.3 in Appendix E.3.

6 Robust Unobfuscatable PRFs

In this section, we define and construct robust unobfuscatable PRFs. We first give its formal definition in Sec. 6.1. Then in Sec. 6.2 and Sec. 6.3, we present constructions with different properties.

6.1 The Definition

We give definition of robust unobfuscatable PRFs in this section, which follows definitions in previous works [BGI+01, BP13, Zha21] with slight modifications. More precisely, an unobfuscatable PRF with input space $\mathcal{X}$, output space $\mathcal{Y}$, and message space $\mathcal{M}$ consists of the following algorithms:

- $\mathtt{Setup}(1^\lambda) \to PP$: On input the security parameter 1^λ, the setup algorithm outputs the public parameter PP.
- $\mathtt{KeyGen}(PP, msg) \to K$: On input the public parameter PP and a message $msg \in \mathcal{M}$, the key generation algorithm outputs a PRF key K.
- $\mathtt{Eval}(PP, K, x) \to y$: On input the public parameter PP, a PRF key K, and an input $x \in \mathcal{X}$, the evaluation algorithm outputs an output $y \in \mathcal{Y}$.
- $\mathtt{Extract}(PP, \mathtt{C}) \to msg$: On input the public parameter PP and a circuit $\mathtt{C}$, the extraction algorithm outputs a message $msg \in \mathcal{M} \cup \{\bot\}$, where $\bot$ denotes that the extraction fails.

Correctness. Its correctness requires that the extraction algorithms can always output the correct message given an honestly generated secret key.

Definition 6.1 (Correctness). *For any $msg \in \mathcal{M}$, let $PP \leftarrow \mathtt{Setup}(1^\lambda)$ and $K \leftarrow \mathtt{KeyGen}(PP, msg)$, then we have $\Pr[\mathtt{Extract}(PP, \mathtt{Eval}(PP, K, \cdot)) \neq msg] = 0$.*

Pseudorandomness. Its black-box pseudorandomness requires that outputs of the evaluation algorithm are indistinguishable from outputs of a random function if the adversary is only given oracle access to the evaluation algorithm.

Definition 6.2 (Black-Box Pseudorandomness). *For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, let $PP \leftarrow \mathtt{Setup}(1^\lambda)$. Also, let $(msg, state) \leftarrow \mathcal{A}_1(PP)$, $K \leftarrow \mathtt{KeyGen}(PP, msg)$, and f be a random function from $\mathcal{X}$ to $\mathcal{Y}$. Let $\mathcal{O}_0(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $\mathtt{Eval}(PP, K, x)$, and let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $f(x)$. We have*

$$|\Pr[\mathcal{A}_2^{\mathcal{O}_0(\cdot)}(state) = 1] - \Pr[\mathcal{A}_2^{\mathcal{O}_1(\cdot)}(state) = 1]| \leq negl(\lambda)$$

Learnability. Its robust non-black-box learnability requires that one can learn the message from a circuit that approximately evaluates the PRF. In particular, for any function $\epsilon \in [0,1]$, we define ϵ-robust learnability as follows.

Definition 6.3 (ϵ-Robust Learnability). *For all PPT and ϵ-admissible adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\Pr\left[\begin{array}{l} PP \leftarrow \mathtt{Setup}(1^\lambda); \\ (msg, state) \leftarrow \mathcal{A}_1(PP); \\ K \leftarrow \mathtt{KeyGen}(PP, msg); \ : \ \ \overline{msg} \neq msg \\ \mathtt{C} \leftarrow \mathcal{A}_2(K, state); \\ \overline{msg} \leftarrow \mathtt{Extract}(PP, \mathtt{C}); \end{array}\right] \leq negl(\lambda)$$

Here, we say that an adversary $\mathcal{A}$ is ϵ-admissible *if $|\{x \in \mathcal{X} : \mathtt{C}(x) \neq \mathtt{Eval}(PP, K, x)\}| \leq \epsilon \cdot |\mathcal{X}|$.*[10]

6.2 Robust Unobfuscatable PRFs from One Way Functions

In this section, we provide our construction of robust unobfuscatable PRFs from one way functions.

Let λ be the security parameter. Let $n_0 = 3$, $n_1 = (\lambda+2)\cdot\lambda$, $n_2 = 2(\lambda+1)$, $n_3 = \lambda+1$, $n_4 = \lambda$, $n_5 = \lambda\cdot(\lambda+1)$. Let $n = n_0 + n_1 + n_2 + n_3 + n_4 + n_5$. Let m be a positive integer that is polynomial in λ s.t. $m \geq \lambda\cdot(\lambda+1)$.

Our construction is built on the following building blocks, all of which can be constructed from one way functions:

- A PRF $\mathsf{F}_{enc} = (\mathsf{F}_{enc}.\mathtt{KeyGen}, \mathsf{F}_{enc}.\mathtt{Eval})$ with input space $\{0,1\}^\lambda$ and output space $\{0,1\}$.
- An invoker randomizable PRF $\mathsf{F}_{IR} = (\mathsf{F}_{IR}.\mathtt{KeyGen}, \mathsf{F}_{IR}.\mathtt{Eval})$ with input space $\{0,1\}^{n-n_1} \times \{0,1\}^{n_1}$ and output space $\{0,1\}^{n_1}$.
- A PRF $\mathsf{F}_{mask} = (\mathsf{F}_{mask}.\mathtt{KeyGen}, \mathsf{F}_{mask}.\mathtt{Eval})$ with input space $\{0,1\}^{n-n_0}$ and output space $\{0,1\}^m$.
- A PRF $\mathsf{F}_{pad} = (\mathsf{F}_{pad}.\mathtt{KeyGen}, \mathsf{F}_{pad}.\mathtt{Eval})$ with input space $\{0,1\}^n$ and output space $\{0,1\}^m$.

We construct the robust unobfuscatable PRF $\mathsf{UOF} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Eval}, \mathtt{Extract})$, which has input space $\{0,1\}^n$, output space $\{0,1\}^m$, and message space $\{0,1\}^m$ as follows:

- $\mathtt{Setup}$. There is no need to set the public parameter in this construction and the setup algorithm outputs $PP = 1^\lambda$ on input the security parameter 1^λ.
- $\mathtt{KeyGen}$. On input the security parameter 1^λ and the message msg, the key generation algorithm samples $\alpha \xleftarrow{\$} \{0,1\}^\lambda$ and $\beta \xleftarrow{\$} \{0,1\}^\lambda$. Then it generates PRF keys $k_{enc} \leftarrow \mathsf{F}_{enc}.\mathtt{KeyGen}(1^\lambda)$, $k_{IR} \leftarrow \mathsf{F}_{IR}.\mathtt{KeyGen}(1^\lambda)$,

[10] Similar to a (hinting) watermarkable PRF, we can define $negl(\lambda)$-robust learnability, which requires that $|\{x \in \mathcal{X} : \mathtt{C}(x) \neq \mathtt{Eval}(PP, K, x)\}| \leq negl(\lambda)\cdot|\mathcal{X}|$.

$k_{mask} \leftarrow \mathsf{F}_{mask}.\mathtt{KeyGen}(1^\lambda)$, and $k_{pad} \leftarrow \mathsf{F}_{pad}.\mathtt{KeyGen}(1^\lambda)$. Finally, it outputs the PRF key

$$K = (\alpha, \beta, k_{enc}, k_{IR}, k_{mask}, k_{pad}, msg)$$

- `Eval`. On input the secret key $K = (\alpha, \beta, k_{enc}, k_{IR}, k_{mask}, k_{pad}, msg)$ and an input $x \in \{0,1\}^n$, the evaluation algorithm first parses $x = (u_0, u_1, u_2, u_3, u_4, u_5) \in \{0,1\}^{n_0} \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} \times \{0,1\}^{n_4} \times \{0,1\}^{n_5}$. Then it sets $w = (u_0, u_2, u_3, u_4, u_5)$ and computes $(r_1, r_2, \ldots, r_{\lambda+2}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, w, u_1)$, where $r_i \in \{0,1\}^\lambda$ for $i \in [1, \lambda+2]$. Next, it deals with the following cases:
  - If $u_0 = 0$:
    1. For $i \in [1, \lambda]$:
       (a) $ct_i = r_i \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_i) \oplus \alpha[i]$.
    2. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)[1 : m - \lambda \cdot (\lambda+1)]$.
    3. Output $ct_1 \| \ldots \| ct_\lambda \| y_{pad}$.
  - If $u_0 = 1$:
    1. Parse $u_2 = (\bar{r}_1, \bar{c}_1, \bar{r}_2, \bar{c}_2) \in \{0,1\}^\lambda \times \{0,1\} \times \{0,1\}^\lambda \times \{0,1\}$.
    2. $\mu_1 = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_1) \oplus \bar{c}_1$.
    3. $\mu_2 = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_2) \oplus \bar{c}_2$.
    4. $\mu = \mu_1 \barwedge \mu_2$.
    5. $ct = r_{\lambda+1} \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_{\lambda+1}) \oplus \mu$.
    6. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)[1 : m - (\lambda+1)]$.
    7. Output $ct \| y_{pad}$.
  - If $u_0 = 2$:
    1. Parse $u_3 = (\bar{r}, \bar{c}) \in \{0,1\}^\lambda \times \{0,1\}$.
    2. $\mu = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}) \oplus \bar{c}$.
    3. $ct = r_{\lambda+2} \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_{\lambda+2}) \oplus \mu$.
    4. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)[1 : m - (\lambda+1)]$.
    5. Output $ct \| y_{pad}$.
  - If $u_0 = 3$:
    1. $z = (u_1, u_2, u_3, u_4, u_5)$.
    2. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
    3. Output y_{mask}.
  - If $u_0 = 4$:
    1. $u'_4 = u_4 \oplus \alpha$.
    2. $z = (u_1, u_2, u_3, u'_4, u_5)$.
    3. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
    4. Output $(\beta \| 0^{m-\lambda}) \oplus y_{mask}$.
  - If $u_0 = 5$:
    1. Parse $u_5 = (\bar{r}_i, \bar{c}_i)_{i \in [1,\lambda]} \in (\{0,1\}^\lambda \times \{0,1\})^\lambda$.
    2. For $i \in [1, \lambda]$:
       (a) $\mu_i = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_i) \oplus \bar{c}_i$.
    3. $\nu = \mu_1 \| \ldots \| \mu_\lambda$.
    4. $u'_4 = u_4 \oplus \nu \oplus \beta$.
    5. $z = (u_1, u_2, u_3, u'_4, u_5)$.

6. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
7. Output $msg \oplus y_{mask}$.

– If $u_0 = 6$ or $u_0 = 7$:
1. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)$.
2. Output y_{pad}.

- `Extract`. On input a circuit $\mathtt{C}$, the extraction algorithm first obtains $ct_1, \ldots, ct_\lambda$ as follows:

$$x'_0 \xleftarrow{\$} \{0,1\}^{n-n_0},\ x_0 = 000 \| x'_0,\ y_0 = \mathtt{C}(x_0)$$
$$(ct_1, \ldots, ct_\lambda) = y_0[1 : \lambda \cdot (\lambda + 1)]$$

Then it computes:

$$x'_1 \xleftarrow{\$} \{0,1\}^{n-n_0},\ x_1 = 011 \| x'_1,\ y_1 = \mathtt{C}(x_1)$$

and samples $\gamma \xleftarrow{\$} \{0,1\}^\lambda$. Let $\mathtt{P} = \bar{\mathtt{P}}[x'_1, \mathtt{C}, y_1, \gamma]$, where $\bar{\mathtt{P}}$ is defined in Figure 4. Let $|\mathtt{P}|$ be the number of wires for the circuit $\mathtt{P}$ and label each wire of $\mathtt{P}$ with a number in $[1, |\mathtt{P}|]$, where each wire has a larger label than its children. We can label the input wires as $1, \ldots, \lambda$. Also, we can label the output wires as $|\mathtt{P}| - \lambda + 1, \ldots, |\mathtt{P}|$, where the i-th output wire is labeled with $|\mathtt{P}| - \lambda + i$. Next, the extraction algorithm proceeds as follows for $j \in [\lambda + 1, |\mathtt{P}|]$, where j_L and j_R are the labels of the children of the wire labelled with j:

1. $u_1^{(j)} \xleftarrow{\$} \{0,1\}^{n_1}$.
2. $(u_3^{(j)}, u_4^{(j)}, u_5^{(j)}) \xleftarrow{\$} \{0,1\}^{n_3} \times \{0,1\}^{n_4} \times \{0,1\}^{n_5}$.
3. If $j_L \neq j_R$:
   (a) $u_2^{(j)} = (ct_{j_L}, ct_{j_R})$.
4. If $j_L = j_R$:
   (a) $(\bar{u}_1^{(j)}, \bar{u}_2^{(j)}) \xleftarrow{\$} \{0,1\}^{n_1} \times \{0,1\}^{n_2}$.
   (b) $(\bar{u}_4^{(j)}, \bar{u}_5^{(j)}) \xleftarrow{\$} \{0,1\}^{n_4} \times \{0,1\}^{n_5}$.
   (c) $\bar{u}_3^{(j)} = ct_{j_L}$.
   (d) $\bar{u}_0^{(j)} = 010$.
   (e) $\bar{x}_{2,j} = \bar{u}_0^{(j)} \| \bar{u}_1^{(j)} \| \bar{u}_2^{(j)} \| \bar{u}_3^{(j)} \| \bar{u}_4^{(j)} \| \bar{u}_5^{(j)}$.
   (f) $\bar{y}_{2,j} = \mathtt{C}(\bar{x}_{2,j})$.
   (g) $\bar{ct}_{j_L} = \bar{y}_{2,j}[1 : \lambda + 1]$.
   (h) $u_2^{(j)} = (ct_{j_L}, \bar{ct}_{j_L})$.
5. $u_0^{(j)} = 001$.
6. $x_{2,j} = u_0^{(j)} \| u_1^{(j)} \| u_2^{(j)} \| u_3^{(j)} \| u_4^{(j)} \| u_5^{(j)}$.
7. $y_{2,j} = \mathtt{C}(x_{2,j})$.
8. $ct_j = y_{2,j}[1 : \lambda + 1]$.

After obtaining $ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|}$, the extraction algorithm finally extracts the message as follows:

1. $(u_1, u_2, u_3, u_4) \xleftarrow{\$} \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} \times \{0,1\}^{n_4}$.
2. $u_5 = (ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|})$.

3. $u_0 = 101$.
4. $x_3 = u_0 \| u_1 \| u_2 \| u_3 \| u_4 \| u_5$.
5. $y_3 = \mathtt{C}(x_3)$.
6. $\tilde{u}_0 = 011$.
7. $\tilde{u}_4 = u_4 \oplus \gamma$.
8. $\tilde{x}_3 = \tilde{u}_0 \| u_1 \| u_2 \| u_3 \| \tilde{u}_4 \| u_5$.
9. $\tilde{y}_3 = \mathtt{C}(\tilde{x}_3)$.
10. $msg = \tilde{y}_3 \oplus y_3$.

Finally, it outputs msg.

$\bar{\mathtt{P}}$

Constant: $x'_1, \mathtt{C}, y_1, \gamma$

Input: a

1. Parse $x'_1 = (u_1, u_2, u_3, u_4, u_5) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} \times \{0,1\}^{n_4} \times \{0,1\}^{n_5}$.
2. $\tilde{u}_0 = 100$.
3. $\tilde{u}_4 = u_4 \oplus a$.
4. $\tilde{x}_1 = \tilde{u}_0 \| u_1 \| u_2 \| u_3 \| \tilde{u}_4 \| u_5$.
5. $\tilde{y}_1 = \mathtt{C}(\tilde{x}_1)$.
6. $b = (\tilde{y}_1 \oplus y_1)[1 : \lambda]$.
7. Output $b \oplus \gamma$.

Fig. 4 The circuit $\bar{\mathtt{P}}$.

Theorem 6.1. *If* $\mathsf{F}_{enc}, \mathsf{F}_{mask}, \mathsf{F}_{pad}$ *are secure PRFs and* F_{IR} *is a secure invoker randomizable PRF, then* UOF *is a secure robust unobfuscatable PRF family with* $negl(\lambda)$*-robust learnability.*

We present proof of Theorem 6.1 in Appendix E.4.

6.3 Robust Unobfuscatable PRFs from Fully Homomorphic Encryption

In this section, we construct robust unobfuscatable PRFs from fully homomorphic encryption. Our construction relies on a special FHE scheme, which is defined and constructed in Appendix D.

Let λ be the security parameter. Let $L_c, L_t, L_p, L_e, l_{enc}, l_{rpk}, l_{rek}$ be positive integers that are polynomial in λ. Let $n_0 = \lambda$, $n_1 = \lambda \cdot l_{enc} + l_{rpk} + l_{rek}$, $n_2 = \lambda$, and $n_3 = \lambda \cdot L_t$. Let $m_1 = \lambda \cdot L_c + L_p + L_e$ and let m_2 be a positive integer that is polynomial in λ. Let $n = n_0 + n_1 + n_2 + n_3$ and $m = m_1 + m_2$. Let ε be a real value in $(0, 1)$ s.t. $1/\varepsilon$ is polynomial in λ and $N = \lambda/\varepsilon^2$.

Our construction is built on the following building blocks:

- A special FHE $\mathsf{SFHE} = (\mathsf{SFHE}.\mathtt{KeyGen}, \mathsf{SFHE}.\mathtt{Enc}, \mathsf{SFHE}.\mathtt{Dec}, \mathsf{SFHE}.\mathtt{Eval}, \mathsf{SFHE}.\mathtt{RandPK}, \mathsf{SFHE}.\mathtt{RandEK}, \mathsf{SFHE}.\mathtt{TranCT}, \mathsf{SFHE}.\mathtt{TCTDec})$. Here, the message space is $\{0,1\}$, the output space of algorithms $\mathsf{SFHE}.\mathtt{Enc}$, $\mathsf{SFHE}.\mathtt{RandPK}$,

$\mathsf{SFHE}.\mathtt{RandEK}$ and $\mathsf{SFHE}.\mathtt{TranCT}$ are $\{0,1\}^{L_c}$, $\{0,1\}^{L_p}$, $\{0,1\}^{L_e}$ and $\{0,1\}^{L_t}$ respectively. In addition, the randomness space of the algorithms $\mathsf{SFHE}.\mathtt{Enc}$, $\mathsf{SFHE}.\mathtt{RandPK}$, and $\mathsf{SFHE}.\mathtt{RandEK}$ are $\{0,1\}^{l_{enc}}$, $\{0,1\}^{l_{rpk}}$, and $\{0,1\}^{l_{rek}}$ respectively.

- An invoker randomizable PRF $\mathsf{F}_{IR} = (\mathsf{F}_{IR}.\mathtt{KeyGen}, \mathsf{F}_{IR}.\mathtt{Eval})$ with input space $\{0,1\}^{n-n_1} \times \{0,1\}^{n_1}$ and output space $\{0,1\}^{n_1}$.
- A PRF $\mathsf{F}_{mask} = (\mathsf{F}_{mask}.\mathtt{KeyGen}, \mathsf{F}_{mask}.\mathtt{Eval})$ with input space $\{0,1\}^n$ and output space $\{0,1\}^{m_2}$.

We construct the robust unobfuscatable PRF $\mathsf{UOF} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Eval}, \mathtt{Extract})$, which has input space $\{0,1\}^n$, output space $\{0,1\}^m$, and message space $\{0,1\}^{m_2}$ as follows:

- $\mathtt{Setup}$. There is no need to set the public parameter in this construction and the setup algorithm outputs $PP = 1^\lambda$ on input the security parameter 1^λ.
- $\mathtt{KeyGen}$. On input the security parameter 1^λ and the message msg, the key generation algorithm samples $\alpha \xleftarrow{\$} \{0,1\}^\lambda$ and $\beta \xleftarrow{\$} \{0,1\}^\lambda$. Then it generates $k_{IR} \leftarrow \mathsf{F}_{IR}.\mathtt{KeyGen}(1^\lambda)$, $k_{mask} \leftarrow \mathsf{F}_{mask}.\mathtt{KeyGen}(1^\lambda)$, and $(pk, sk, ek) \leftarrow \mathsf{SFHE}.\mathtt{KeyGen}(1^\lambda)$. Finally, it outputs the PRF key

$$K = (\alpha, \beta, k_{IR}, k_{mask}, pk, sk, ek, msg)$$

- $\mathtt{Eval}$. On input the secret key $K = (\alpha, \beta, k_{IR}, k_{mask}, pk, sk, ek, msg)$, and an input $x \in \{0,1\}^n$, the evaluation algorithm first parses $x = (u_0, u_1, u_2, u_3) \in \{0,1\}^{n_0} \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3}$. Then it generates the first part of the output, which contains ciphertexts for α, the randomized public key, and the randomized evaluation key as follows:
  1. $w = (u_0, u_2, u_3)$.
  2. $(r_1, \ldots, r_\lambda, r_{rpk}, r_{rek}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, w, u_1)$ (Here, $r_i \in \{0,1\}^{l_{enc}}$ for $i \in [1, \lambda]$, $r_{rpk} \in \{0,1\}^{l_{rpk}}$ and $r_{rek} \in \{0,1\}^{l_{rek}}$).
  3. For $i \in [1, \lambda]$:
     (a) $ct_i = \mathsf{SFHE}.\mathtt{Enc}(pk, \alpha[i]; r_i)$.
  4. $rp\mathcal{k} = \mathsf{SFHE}.\mathtt{RandPK}(pk; r_{rpk})$.
  5. $re\mathcal{k} = \mathsf{SFHE}.\mathtt{RandEK}(pk, ek; r_{rek})$.
  6. $y_{ct} = ct_1 \| ct_2 \| \ldots \| ct_\lambda \| rp\mathcal{k} \| re\mathcal{k}$

  Let $d \in \{0,1,2\}$ s.t. $d = u_0 \mod 3$. Then the evaluation algorithm deals with the following cases:
  - If $d = 0$:
    1. $z = (u_0, u_1, u_2, u_3)$.
    2. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
    3. Output $y_{ct} \| y_{mask}$.
  - If $d = 1$:
    1. $u_0' = u_0 - 1$.
    2. $u_2' = u_2 \oplus \alpha$.
    3. $z = (u_0', u_1, u_2', u_3)$.
    4. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
    5. Output $y_{ct} \| ((\beta \| 0^{m_2 - \lambda}) \oplus y_{mask})$.

– If $d = 2$:
  1. Parse $u_3 = (\mathit{tct}_i)_{i\in[1,\lambda]} \in (\{0,1\}^{L_t})^\lambda$.
  2. For $i \in [1, \lambda]$:
     (a) $\mu_i = \mathsf{SFHE}.\mathtt{TCTDec}(sk, \mathit{tct}_i)$.
  3. $\nu = \mu_1 \| \ldots \| \mu_\lambda$.
  4. $u'_2 = u_2 \oplus \nu \oplus \beta$.
  5. $u'_0 = u_0 - 2$.
  6. $z = (u'_0, u_1, u'_2, u_3)$.
  7. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
  8. Output $y_{ct} \| (msg \oplus y_{mask})$.

- `Extract`. On input a circuit `C`, the extraction algorithm **outputs** $\mathtt{ExtractI}(\mathtt{C})$, where we define the algorithms `ExtractI`, `ExtractII`, and `ExtractIII` as follows:
  - `ExtractI`. On input a circuit `C`, the algorithm proceeds as follows for $i \in [1, N]$:
    1. $x_1^{(i)} \xleftarrow{\$} \{0,1\}^n$.
    2. $y_1^{(i)} = \mathtt{C}(x_1^{(i)})$.
    3. Let $(\boldsymbol{ct}_\alpha^{(i)}, \mathit{rp\mathcal{K}}^{(i)}, \mathit{re\mathcal{K}}^{(i)}) = y_1^{(i)}[1 : m_1]$, where $\boldsymbol{ct}_\alpha^{(i)}$ is a vector of λ ciphertexts in $\{0,1\}^{L_c}$, $\mathit{rp\mathcal{K}} \in \{0,1\}^{L_p}$, and $\mathit{re\mathcal{K}} \in \{0,1\}^{L_e}$.
    4. $msg^{(i)} \leftarrow \mathtt{ExtractII}(\mathtt{C}, \boldsymbol{ct}_\alpha^{(i)}, \mathit{rp\mathcal{K}}^{(i)}, \mathit{re\mathcal{K}}^{(i)})$.

    Finally, after obtaining $msg^{(i)}$ for $i \in [1, N]$, the algorithm **outputs** msg if there exists $msg \in \{0,1\}^{m_2}$ s.t. $|\{i : msg^{(i)} = msg\}| > \frac{N}{2}$; otherwise, it **outputs** $\perp$.
  - `ExtractII`. On input a circuit `C`, a vector of ciphertexts $\boldsymbol{ct}_\alpha$, a randomized public key $\mathit{rp\mathcal{K}}$, and a randomized evaluation key $\mathit{re\mathcal{K}}$, the algorithm proceeds as follows for $i \in [1, N]$:
    1. $\psi^{(i)} \xleftarrow{\$} [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1]$.
    2. $u_0^{(i)} = 3 \cdot \psi^{(i)}$.
    3. $\tilde{u}_0^{(i)} = u_0^{(i)} + 1$. (As $u_0^{(i)} < \tilde{u}_0^{(i)} < 2^\lambda - 2$, $u_0^{(i)}, \tilde{u}_0^{(i)} \in \{0,1\}^\lambda$)
    4. $(u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \leftarrow \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3}$.
    5. $x_2^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$.
    6. $y_2^{(i)} = \mathtt{C}(x_2^{(i)})$.
    7. Let $\mathtt{P}^{(i)} = \mathtt{P}[\tilde{u}_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}, \mathtt{C}, y_2^{(i)}]$, where `P` is defined in Figure 5.
    8. $\boldsymbol{ct}_\beta^{(i)} \leftarrow \mathsf{SFHE}.\mathtt{Eval}(\mathit{re\mathcal{K}}, \boldsymbol{ct}_\alpha, \mathtt{P}^{(i)})$.
    9. $msg^{(i)} \leftarrow \mathtt{ExtractIII}(\mathtt{C}, \boldsymbol{ct}_\beta^{(i)}, \mathit{rp\mathcal{K}}, \mathit{re\mathcal{K}})$.

    Finally, after obtaining $msg^{(i)}$ for $i \in [1, N]$, the algorithm outputs msg if there exists $msg \in \{0,1\}^{m_2}$ s.t. $|\{i : msg^{(i)} = msg\}| > \frac{N}{2}$; otherwise, it outputs $\perp$.
  - `ExtractIII`. On input a circuit `C`, a vector of ciphertexts $\boldsymbol{ct}_\beta$, a randomized public key $\mathit{rp\mathcal{K}}$, and a randomized evaluation key $\mathit{re\mathcal{K}}$, the algorithm proceeds as follows for $i \in [1, N]$:
    1. $\psi^{(i)} \xleftarrow{\$} [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1]$.
    2. $\tilde{u}_0^{(i)} = 3 \cdot \psi^{(i)}$.

3. $u_0^{(i)} = \tilde{u}_0^{(i)} + 2$. (As $\tilde{u}_0^{(i)} < u_0^{(i)} < 2^\lambda - 1$, $u_0^{(i)}, \tilde{u}_0^{(i)} \in \{0,1\}^\lambda$)
4. $(u_1^{(i)}, u_2^{(i)}) \leftarrow \{0,1\}^{n_1} \times \{0,1\}^{n_2}$.
5. $\gamma^{(i)} \xleftarrow{\$} \{0,1\}^\lambda$.
6. Let $\mathtt{P}'^{(i)} = \mathtt{P}'[\gamma^{(i)}]$, where $\mathtt{P}'$ is defined in Figure 5.
7. $\boldsymbol{ct}_{\beta,\gamma}^{(i)} \leftarrow \mathsf{SFHE}.\mathtt{Eval}(\mathit{rek}, \boldsymbol{ct}_\beta, \mathtt{P}'^{(i)})$.
8. For $j \in [1, \lambda]$:
   (a) $\mathit{tct}_j^{(i)} \leftarrow \mathsf{SFHE}.\mathtt{TranCT}(\mathit{rpk}, \boldsymbol{ct}_{\beta,\gamma}^{(i)}[j])$
9. $u_3^{(i)} = \mathit{tct}_1^{(i)} \| \ldots \| \mathit{tct}_\lambda^{(i)}$.
10. $x_3^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$.
11. $y_3^{(i)} = \mathtt{C}(x_3^{(i)})$.
12. $\tilde{u}_2^{(i)} = u_2^{(i)} \oplus \gamma^{(i)}$.
13. $\tilde{x}_3^{(i)} = (\tilde{u}_0^{(i)}, u_1^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)})$.
14. $\tilde{y}_3^{(i)} = \mathtt{C}(\tilde{x}_3^{(i)})$.
15. $msg^{(i)} = (y_3^{(i)} \oplus \tilde{y}_3^{(i)})[m_1 + 1 : m]$.

Finally, after obtaining $msg^{(i)}$ for $i \in [1, N]$, the algorithm outputs msg if there exists $msg \in \{0,1\}^{m_2}$ s.t. $|\{i : msg^{(i)} = msg\}| > \frac{N}{2}$; otherwise, it outputs $\perp$.

$\mathtt{P}$	$\mathtt{P}'$
Constant: $\tilde{u}_0, u_1, u_2, u_3, \mathtt{C}, y_2$ **Input:** a 1. $\tilde{u}_2 = u_2 \oplus a$. 2. $\tilde{x}_2 = (\tilde{u}_0, u_1, \tilde{u}_2, u_3)$. 3. $\tilde{y}_2 = \mathtt{C}(\tilde{x}_2)$. 4. Output $b = (\tilde{y}_2 \oplus y_2)[m_1 + 1 : m_1 + \lambda]$.	**Constant**: γ **Input:** b 1. Output $b \oplus \gamma$.

Fig. 5 The circuit $\mathtt{P}$ and $\mathtt{P}'$.

Theorem 6.2. *If* SFHE *is a secure special FHE scheme defined in Appendix D,* F_{mask} *is a secure PRF and* F_{IR} *is a secure invoker randomizable PRF, then* UOF *is a secure robust unobfuscatable PRF family with* $(\frac{1}{6} - \varepsilon)$*-robust learnability.*

We present proof of Theorem 6.2 in Appendix E.6.

7 Construction of Public-Key Watermarkable PRFs

Now, we present the general construction of public-key watermarkable PRFs from public-key hinting watermarkable PRFs and robust unobfuscatable PRFs.

Let λ be the security parameter. Let $n, m, m_1, m_2, \kappa, l, l_1, l_2, Q$ be positive integers that are polynomial in λ and satisfy $m = m_1 + m_2$ and $l = l_1 + l_2$. Let $\epsilon_1, \epsilon_2, \epsilon$ be real values in $(0, 1)$ s.t. $\epsilon = min(\epsilon_1, \epsilon_2)$.[11]

Our construction is built on the following building blocks:

[11] Here, $\epsilon_1, \epsilon_2, \epsilon$ can be the negligible function $negl(\lambda)$ instead of a concrete value.

- A public-key hinting watermarkable PRF $\mathsf{HWF} = (\mathsf{HWF.Setup}, \mathsf{HWF.KeyGen}, \mathsf{HWF.Eval}, \mathsf{HWF.Mark}, \mathsf{HWF.Extract})$ with input space $\{0,1\}^n$, output space $\{0,1\}^{m_1}$, and message space $\{0,1\}^\kappa$. Also, we use l_1 to denote the length of the hint of HWF (i.e., each hint can be represented by a string in $\{0,1\}^{l_1}$).
- A robust unobfuscatable PRF $\mathsf{UOF} = (\mathsf{UOF.Setup}, \mathsf{UOF.KeyGen}, \mathsf{UOF.Eval}, \mathsf{UOF.Extract})$ with input space $\{0,1\}^n$, output space $\{0,1\}^{m_2}$, and message space $\{0,1\}^l$.
- A PRF $\mathsf{F} = (\mathsf{F.KeyGen}, \mathsf{F.Eval})$ with key space $\{0,1\}^{l_2}$, input space $\{0,1\}^n$ and output space $\{0,1\}^{m_1}$.

We construct the public-key watermarkable PRF $\mathsf{WPRF} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Eval}, \mathtt{Mark}, \mathtt{Extract})$, which has input space $\{0,1\}^n$, output space $\{0,1\}^m$ and message space $\{0,1\}^\kappa$ as follows:

- $\mathtt{Setup}$. On input the security parameter 1^λ, the setup algorithm samples $pp_{hw} \leftarrow \mathsf{HWF.Setup}(1^\lambda)$ and $pp_{uo} \leftarrow \mathsf{UOF.Setup}(1^\lambda)$. Then it outputs the public parameter $PP = (pp_{hw}, pp_{uo})$.
- $\mathtt{KeyGen}$. On input the public parameter $PP = (pp_{hw}, pp_{uo})$, the key generation algorithm first generates $(k_{hw}, \mathtt{hint}) \leftarrow \mathsf{HWF.KeyGen}(pp_{hw})$, $k_f \leftarrow \mathsf{F.KeyGen}(1^\lambda)$, and $k_{uo} \leftarrow \mathsf{UOF.KeyGen}(pp_{uo}, \mathtt{hint} \| k_f)$. Then, it outputs the PRF key $K = (k_{hw}, k_f, k_{uo})$.
- $\mathtt{Eval}$. On input the public parameter $PP = (pp_{hw}, pp_{uo})$, the PRF key $K = (k_{hw}, k_f, k_{uo})$, and an input $x \in \{0,1\}^n$, the evaluation algorithm proceeds as follows:
  1. $y_{hw} = \mathsf{HWF.Eval}(pp_{hw}, k_{hw}, x)$.
  2. $y_f = \mathsf{F.Eval}(k_f, x)$.
  3. $y_{uo} = \mathsf{UOF.Eval}(pp_{uo}, k_{uo}, x)$.
  4. Outputs $y = (y_{hw} \oplus y_f, y_{uo})$.
- $\mathtt{Mark}$. On input the public parameter $PP = (pp_{hw}, pp_{uo})$, the PRF key $K = (k_{hw}, k_f, k_{uo})$, and a message $msg \in \{0,1\}^\kappa$, the marking algorithm computes $\mathtt{C}_{hw} \leftarrow \mathsf{HWF.Mark}(pp_{hw}, k_{hw}, msg)$. Then it outputs a circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ s.t. for any $x \in \{0,1\}^n$:

$$\mathtt{C}(x) = (\mathtt{C}_{hw}(x) \oplus \mathsf{F.Eval}(k_f, x), \mathsf{UOF.Eval}(pp_{uo}, k_{uo}, x))$$

- $\mathtt{Extract}$. On input the public parameter $PP = (pp_{hw}, pp_{uo})$, and a circuit $\mathtt{C}$, the extraction algorithm proceeds as follow:
  1. Set $\mathtt{C}_{uo}$ as a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}_{uo}(x) = \mathtt{C}(x)[m_1 + 1 : m]$.
  2. $(\mathtt{hint}, k_f) \leftarrow \mathsf{UOF.Extract}(pp_{uo}, \mathtt{C}_{uo})$.
  3. If $(\mathtt{hint}, k_f) = \perp$: output $\perp$.
  4. Set $\mathtt{C}_{hw}$ as a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}_{hw}(x) = \mathtt{C}(x)[1 : m_1] \oplus \mathsf{F.Eval}(k_f, x)$.
  5. Output $msg \leftarrow \mathsf{HWF.Extract}(pp_{hw}, \mathtt{C}_{hw}, \mathtt{hint})$.

Theorem 7.1. *If* HWF *is a secure public-key hinting watermarkable PRF with* Q*-bounded* ϵ_1*-unremovability,* UOF *is a secure robust unobfuscatable PRF with* ϵ_2*-robust learnability, and* F *is a secure PRF, then* WPRF *is a secure public-key watermarkable PRF with* Q*-bounded* ϵ*-unremovability.*

We present proof of Theorem 7.1 in Appendix E.7.

On Integrating the Building Blocks. The general construction is built on a public-key hinting watermarkable PRF HWF, a robust unobfuscatable PRF UOF, and a standard PRF F. We require that the input spaces of HWF, UOF, and F are all $\{0,1\}^n$. We also require that the output spaces of HWF and F are both $\{0,1\}^{m_1}$. In addition, we require that the message space of UOF is large enough. We explain how these requirements can be satisfied.

First, it is easy to construct a PRF with any given input/output space via the GGM framework [GGM84], where the key space can be fixed as $\{0,1\}^\lambda$. Also, as in both constructions of robust unobfuscatable PRFs proposed in this work, the message space is determined by the output space of a standard PRF, we can set it to be large enough without affecting other parameters. Besides, for each of the public-key hinting watermarkable PRFs and robust unobfuscatable PRFs constructed in this work, we can extend its input length to be of a given (large enough) number as follows. Therefore, all three requirements can be satisfied.

It remains to show how to adjust the input lengths. First, the input space of the public-key hinting watermarkable PRF constructed in Sec. 5.2 is exactly the input space of a puncturable PRF and we can construct a puncturable PRF with any given input space via the GGM framework. Also, the input spaces of the public-key hinting watermarkable PRFs constructed in Sec. 5.3 and Sec. 5.4 (after instantiating with existing watermarkable PRFs with public extraction) are ciphertext space of a functional encryption scheme and that of a puncturable encryption scheme respectively, and we can extend the ciphertext spaces of both schemes (without compromising their security) via appending a random string of suitable length to the original ciphertext. Moreover, for both robust unobfuscatable PRFs constructed in this work (given in Sec. 6.2 and Sec. 6.3), we can increase the input length n by increasing the parameter n_1 and it is easy to see security of the schemes still holds after this modification.

Instantiations from Concrete Assumptions. We next indicate how to instantiate the general construction from concrete assumptions.

First, in Sec. 5.2, we construct a mark-embedding public-key hinting watermarkable PRF that achieves 1-bounded 1-unremovability from a puncturable PRF and an injective one way function, both of which can be instantiated from standard assumptions such as the LWE assumption. In addition, the robust unobfuscatable PRF from Sec. 6.2 is constructed from one way function and achieves $negl(\lambda)$-robust learnability. Therefore, we obtain a mark-embedding public-key watermarkable PRF with 1-bounded $negl(\lambda)$-unremovability from standard lattice assumptions:

Corollary 7.1. *Assuming the worst-case hardness of appropriately parameterized GapSVP problem, there exist public-key watermarkable PRFs with 1-bounded $negl(\lambda)$-unremovability and message space* $\{$"`marked`"$\}$.

Alternatively, we can use the robust unobfuscatable PRF provided in Sec. 6.3, which has $(\frac{1}{6} - 1/poly(\lambda))$-robust learnability and is constructed from a special FHE scheme and PRFs. In Appendix D, We construct the special FHE

scheme via modifying the GSW FHE scheme [GSW13] and its security relies on the circular security of the GSW scheme. Formally, we have:

Corollary 7.2. *Assuming the circular security of the GSW homomorphic encryption scheme*[12] *and the worst-case hardness of appropriately parameterized GapSVP problem, there exist public-key watermarkable PRFs with 1-bounded* $(\frac{1}{6} - 1/poly(\lambda))$*-unremovability and message space* $\{$"`marked`"$\}$.

The message-embedding and collusion resistant public-key hinting watermarkable PRF given in Sec. 5.3 can be constructed from an FE scheme with strong correctness, a PKE scheme, and a PRF. In Appendix C, we show that the desired FE scheme can be instantiated from standard lattice assumptions. The FE scheme achieves $poly(\lambda)$-adaptive indistinguishability, but it has an exponentially-small ciphertext density. Thus our hinting watermarking scheme has $poly(\lambda)$-bounded ϵ-unremovability for some exponentially-small ϵ.[13] Now, if we employ this scheme and the one way function based robust unobfuscatable PRF in our general construction, we have:

Corollary 7.3. *Assuming the worst-case hardness of appropriately parameterized GapSVP problem, there exist public-key watermarkable PRFs with* $poly(\lambda)$*-bounded* ϵ*-unremovability, input space* $\{0,1\}^n$ *and message space* $\{0,1\}^{poly(\lambda)}$*, where* $\frac{2^{poly(\lambda)}}{2^n} \leq \epsilon \leq \frac{1}{2^{poly(\lambda)}}$.

Besides, the general construction of public-key hinting watermarkable PRF presented in Sec. 5.4 can be instantiated with the watermarkable PRF presented in [CHN+16], which satisfies all properties defined in Appendix B.6 and achieves 1-bounded $(1/2 - 1/poly(\lambda))$-unremovability. The scheme is constructed from iO and injective one way function. Next, combing this instantiation with robust unobfuscatable PRFs constructed in this work, we have:

Corollary 7.4. *Assuming the existence of indistinguishability obfuscation and worst-case hardness of appropriately parameterized GapSVP problem, there exist public-key watermarkable PRFs with* 1*-bounded* $negl(\lambda)$*-unremovability and message space* $\{0,1\}^{poly(\lambda)}$.

Corollary 7.5. *Assuming the existence of indistinguishability obfuscation, the circular security of the GSW homomorphic encryption scheme, and the worst-case hardness of appropriately parameterized GapSVP problem, there exist public-key watermarkable PRFs with* 1*-bounded* $(\frac{1}{6} - 1/poly(\lambda))$*-unremovability and message space* $\{0,1\}^{poly(\lambda)}$.

Remark 7.1. In [YAL+19], a secret key watermarkable PRF that achieves public extraction and collusion resistance is also presented, but the scheme only has a selective version of collusion resistant unremovability, where the adversary

[12] Formal definition for this assumption can be found in Definition E.1, and its relations to the original GSW scheme and the standard LWE assumption are discussed in Remark E.2 and Remark E.3 respectively.

[13] Nonetheless, the watermarked circuit can be modified on exponentially-many inputs.

has to submit all its challenge oracle queries simultaneously. It seems that the restriction can be removed if we modify their construction slightly. However, as our main focus is to construct public-key watermarkable PRFs and achieving collusion resistance would be a secondary concern of this work, we leave the detailed description for the revised construction as future work.

Acknowledgement. We appreciate the anonymous reviewers for their valuable comments. Part of this work is supported by the National Science Foundation China (Project No. 61972332), the Innovation Technology Fund (Project No. ITS/224/20FP), the Hong Kong Research Grant Council (Project No. 17201421), and HKU Seed Fund (grant number: 201909185070). Willy Susilo is partially supported by the Australian Research Council Discovery Projects DP200100144 and DP220100003.

References

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS*, pages 166–175. IEEE, 2004.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *computational complexity*, 15(2):115–162, 2006.

[AJLA+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, pages 483–501. Springer, 2012.

[ALL+21] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In *CRYPTO*, pages 526–555. Springer, 2021.

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, pages 173–205. Springer, 2017.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *TCC*, pages 174–198. Springer, 2019.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73. Springer, 2014.

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

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *CRYPTO*, pages 1–18. Springer, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519. Springer, 2014.

[BKS17] Foteini Baldimtsi, Aggelos Kiayias, and Katerina Samari. Watermarking public-key cryptographic functionalities and implementations. In *ISC*, pages 173–191. Springer, 2017.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.

[BLW17] Dan Boneh, Kevin Lewi, and David J Wu. Constraining pseudorandom functions privately. In *PKC*, pages 494–524. Springer, 2017.

[BP13] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In *STOC*, pages 241–250, 2013.

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

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, pages 280–300. Springer, 2013.

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC*, pages 1115–1127, 2016.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540. Springer, 2004.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct randolli functions. In *FOCS*, pages 464–479. IEEE, 1984.

[GKM+19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J Wu. Watermarking public-key cryptographic primitives. In *CRYPTO*, pages 367–398. Springer, 2019.

[GKP+13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

[GKWW21] Rishab Goyal, Sam Kim, Brent Waters, and David J Wu. Beyond software watermarking: Traitor-tracing for pseudorandom functions. In *ASIACRYPT*, pages 250–280. Springer, 2021.

[Goe15] Michel Goemans. Lecture notes on Chernoff bounds. http://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf, February 2015.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92. Springer, 2013.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179. Springer, 2012.

[HMW07] Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. In *TCC*, pages 362–382. Springer, 2007.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304. IEEE, 2000.

[ILL89] Russell Impagliazzo, Leonid A Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *STOC*, pages 12–24, 1989.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, pages 60–73, 2021.

[KN22] Fuyuki Kitagawa and Ryo Nishimaki. Watermarking PRFs against quantum adversaries. In *EUROCRYPT*, pages 488–518. Springer, 2022.

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In *TCC*, pages 31–61. Springer, 2021.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *CCS*, pages 669–684. ACM, 2013.

[KW17] Sam Kim and David J Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO*, pages 503–536. Springer, 2017.

[KW19] Sam Kim and David J. Wu. Watermarking PRFs from lattices: Stronger security via extractable PRFs. In *CRYPTO*, pages 335–366. Springer, 2019.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In *EUROCRYPT*, pages 239–256. Springer, 2014.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755. Springer, 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. Springer, 2012.

[MW22] Sarasij Maitra and David J Wu. Traceable PRFs: Full collusion resistance and active security. In *PKC*, pages 439–469. Springer, 2022.

[Nis13] Ryo Nishimaki. How to watermark cryptographic functions. In *EUROCRYPT*, pages 111–125. Springer, 2013.

[Nis20] Ryo Nishimaki. Equipping public-key cryptographic primitives with watermarking (or: A hole is to watermark). In *TCC*, pages 179–209. Springer, 2020.

[NSS99] David Naccache, Adi Shamir, and Julien P Stern. How to copyright a function? In *PKC*, pages 188–196. Springer, 1999.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In *EUROCRYPT*, pages 388–419. Springer, 2016.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. https://ia.cr/2010/556.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.

[PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In *PKC*, pages 675–701. Springer, 2018.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, pages 89–114. Springer, 2019.

[PS20] Chris Peikert and Sina Shiehian. Constraining and watermarking PRFs from milder assumptions. In *PKC*, pages 431–461. Springer, 2020.

[QWZ18] Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking PRFs under standard assumptions: Public marking and security with extraction queries. In *TCC*, pages 669–698. Springer, 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

[Rén61] Alfréd Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[Rey11] Leo Reyzin. Extractors and the leftover hash lemma. https://www.cs.bu.edu/~reyzin/teaching/s11cs937/notes-leo-1.pdf, 2011.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

[VEH14] Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

[YAL+19] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Collusion resistant watermarking schemes for cryptographic functionalities. In *ASIACRYPT*, pages 371–398. Springer, 2019.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167. IEEE, 1986.

[YAYX20] Rupeng Yang, Man Ho Au, Zuoxia Yu, and Qiuliang Xu. Collusion resistant watermarkable PRFs from standard assumptions. In *CRYPTO*, pages 590–620. Springer, 2020.

[YF11] Maki Yoshida and Toru Fujiwara. Toward digital watermarking for cryptographic data. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 94(1):270–272, 2011.

[Zha21] Mark Zhandry. White box traitor tracing. In *CRYPTO*, pages 303–333. Springer, 2021.

A Related Work

Watermarking Public-Key Cryptographic Primitives. In [CHN+16], Cohen et al. also construct watermarking schemes for PKE and signature. The constructions rely on their watermarkable PRF and achieves secret marking and public extraction. Then in [BKS17], Baldimtsi et al. construct (stateful) watermarkable PKE with secret marking and public extraction from any identity-based encryption scheme. Subsequently, in [GKM+19], Goyal et al. present constructions of public-key watermarking schemes for various public key cryptographic primitives from standard assumptions. Recently, in [Nis20], Nishimaki gives a general framework to watermark any public key primitive that uses some specific techniques in its security proof.

Note that while public-key watermarking schemes for public key primitives are achieved in [GKM+19,Nis20], it is unknown if those techniques can be applied to construct public-key watermarkable PRFs.

Traceable PRFs. In recent works [GKWW21,MW22], a variant of watermarkable PRF called traceable PRF, which has a stronger security requirement, is presented. Specially, in their security definition, the extraction algorithm should still be able to extract the embedded message from a circuit that changes all outputs of the watermarked circuit, as long as the circuit can be used to break pseudorandomness of the PRF. Traceable PRFs that have secret extraction and public extraction are constructed from standard lattice assumptions and iO respectively, but the constructions only support secret marking.

Watermarkable PRFs against Quantum Adversaries. In a concurrent work [KN22], Kitagawa and Nishimaki present watermarkable/traceable PRFs

that are secure against a quantum adversary, which can produce a quantum circuit as its challenge. They construct a scheme with public marking and secret extraction from LWE and a scheme with public marking and public extraction from iO. However, in their schemes, the extraction algorithm needs to take as input a tag associated with the watermarked PRF key. This is similar to the syntax of the hinting watermarkable PRF defined in this work, and it is interesting to see if this restriction can be removed using our techniques.

B Preliminaries

We first recall some useful facts used in this work.

Chernoff Bound. We make use of the Chernoff bound in the security analysis of our schemes. There are various forms of the Chernoff bound, here we use the one from [Goe15].

Lemma B.1 (Chernoff Bounds). *Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}(X) = \sum_{i=1}^{n} p_i$. Then*

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2}{2+\delta}\mu} \text{ for all } \delta > 0;$$

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2}{2}\mu} \text{ for all } 0 < \delta < 1.$$

Statistical Distance and Entropy. For any two random variables X, Y, we use

$$\mathtt{SD}(X, Y) = \frac{1}{2} \sum_{u} |\Pr[X = u] - \Pr[Y = u]|$$

to denote the statistical distance between X and Y. We say that X is statistically indistinguishable from Y (denoted as $X \approx_s Y$) if $\mathtt{SD}(X, Y)$ is negligible. We use the following properties of statistical distance in this paper. Proofs of these properties can be found in many previous works, e.g., this lecture [Rey11].

Lemma B.2. *Let X, Y, Z be random variables with range $\mathcal{U}$.*

- *Let $\mathcal{T} = \{u \in \mathcal{U} : \Pr[X = u] - \Pr[Y = u] > 0\}$, then*

$$\mathtt{SD}(X, Y) = \sum_{u \in \mathcal{T}} (\Pr[X = u] - \Pr[Y = u])$$

- *For any* deterministic *function F,* $\mathtt{SD}(F(X), F(Y)) \leq \mathtt{SD}(X, Y)$.
- $\mathtt{SD}(X, Z) \leq \mathtt{SD}(X, Y) + \mathtt{SD}(Y, Z)$.

For any random variable X, we use $H_\infty(X)$ to denote its min-entropy, which is defined as $H_\infty(X) = -\log(max_x \Pr[X = x])$. Given two random variables X, Y, the average min-entropy ([DRS04]) of X conditioned on Y (denoted as $H_\infty(X \mid Y)$) is defined as

$$H_\infty(X \mid Y) = -\log(\mathbb{E}_{y \leftarrow Y}[max_x \Pr[X = x \mid Y = y]])$$

Lemma B.3 ([DRS04]). *Let X, Y, Z be random variables and Z has at most 2^l possible values, then $H_\infty(X \mid (Y, Z)) \geq H_\infty(X \mid Y) - l$.*

The leftover hash lemma allows one to map a random variable with high (average) min-entropy to a uniform string. It relies on a universal hash family:

Definition B.1. *A family $\mathcal{H}$ of hash functions from $\{0,1\}^n$ to $\{0,1\}^m$ is called universal if for any distinct $x, y \in \{0,1\}^n$,*

$$\Pr[h \xleftarrow{\$} \mathcal{H} : h(x) = h(y)] \leq 2^{-m}$$

Lemma B.4 (Leftover Hash Lemma, [ILL89,DRS04]). *Let $\mathcal{H}$ be a universal hash function family from $\mathcal{X}$ to $\mathcal{Y}$. Let X, Z be random variables that satisfy $H_\infty(X \mid Z) \geq \log|\mathcal{Y}| + \omega(\log\lambda)$ and X is over $\mathcal{X}$. Let $H \xleftarrow{\$} \mathcal{H}$ and $Y \xleftarrow{\$} \mathcal{Y}$, then*

$$\mathtt{SD}((H, H(X), Z), (H, Y, Z)) \leq negl(\lambda)$$

In this work, we consider the concrete universal hash family that maps a vector in $\mathbb{Z}_q^m$ to a vector in $\mathbb{Z}_q^n$ by multiplying a matrix in $\mathbb{Z}_q^{n\times m}$, where q is a prime. Formally, we will use the following lemma, which is implied by the leftover hash lemma directly.

Lemma B.5. *Let q be a prime. Let $\boldsymbol{r} \in \mathbb{Z}_q^m$ that satisfies $H_\infty(\boldsymbol{r}) \geq n \cdot \log q + \omega(\log\lambda)$. Let $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_q^n$. Then we have*

$$\mathtt{SD}((\boldsymbol{A}, \boldsymbol{A}\cdot\boldsymbol{r}), (\boldsymbol{A}, \boldsymbol{b})) \leq negl(\lambda)$$

Discrete Gaussian Distribution. The continuous Gaussian distribution over $\mathbb{R}$ with standard deviation σ is defined by the function

$$\rho_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{\frac{-x^2}{2\sigma^2}}$$

The discrete Gaussian distribution over $\mathbb{Z}$ with standard deviation σ is defined as

$$D_\sigma(x) = \frac{\rho_\sigma(x)}{\rho_\sigma(\mathbb{Z})}$$

where

$$\rho_\sigma(\mathbb{Z}) = \sum_{x\in\mathbb{Z}} \rho_\sigma(x)$$

Lemma B.6 ([Lyu12]). *For any $k > 0$, $\Pr[|z| > k\sigma : z \leftarrow D_\sigma] \leq 2e^{\frac{-k^2}{2}}$.*

We also use the following truncated discrete Gaussian distribution over $\mathbb{Z}$:

$$\tilde{D}_\sigma(x) = \begin{cases} D_\sigma(0) + \sum_{x\in\mathbb{Z}\wedge|x|>\lambda\cdot\sigma} D_\sigma(x) & \text{If } x = 0; \\ D_\sigma(x) & \text{If } 0 < |x| \leq \lambda\cdot\sigma; \\ 0 & \text{If } |x| > \lambda\cdot\sigma. \end{cases}$$

where λ is a security parameter. By Lemma B.6, the statistical distance between D_σ and $\tilde{D}_\sigma$ is negligible in λ. Also, to sample an element from D_σ, one can first sample $x \leftarrow D_\sigma$ and outputs x if $|x| \leq \lambda \cdot \sigma$, and outputs 0 otherwise.

The LWE Assumption. We will use the LWE assumption in this paper.

Definition B.2 (Decision-LWE$_{n,m,q,\chi}$). *Given a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$, and a vector $\boldsymbol{b} \in \mathbb{Z}_q^m$, where $\boldsymbol{b}$ is generated according to either of the following two cases:*

1. $\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod q$, *where* $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$ *and* $\boldsymbol{e} \leftarrow \chi^m$
2. $\boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_q^m$

distinguish which is the case with non-negligible advantage.

Let $m = poly(n)$, $poly(n) \leq q \leq 2^{poly(n)}$, and $\chi = D_\sigma$ (or $\tilde{D}_\sigma$) be a (truncated) discrete Gaussian error distribution with standard deviation $\sigma \geq O(\sqrt{n})$, then solving the decision-LWE$_{n,m,q,\chi}$ problem is as hard as solving the GapSVP$_\gamma$ problem on arbitrary n-dimensional lattices by a quantum algorithm [Reg05], where $\gamma = \tilde{O}(nq/\sigma)$. In subsequent works [Pei09, BLP+13], classical reductions from LWE to GapSVP are also presented for different parameterizations.

B.1 Injective One Way Function

An injective function $\mathtt{F} : \{0,1\}^n \to \{0,1\}^m$ is an injective one way function if it can be evaluated in polynomial time and satisfies the following condition:

- **One-wayness.** For all PPT adversary $\mathcal{A}$, we have:

$$\Pr[x \xleftarrow{\$} \{0,1\}^n, y = \mathtt{F}(x), x' \leftarrow \mathcal{A}(1^n, y) : \mathtt{F}(x') = y] \leq negl(n)$$

Injective One way functions can be constructed from standard assumptions such as the LWE assumption.

B.2 Pseudorandom Objects

We will use pseudorandom functions, puncturable pseudorandom functions, and invoker randomizable pseudorandom functions in our constructions. All of them can be constructed from the (standard) one way function.

Pseudorandom Functions. A PRF with input space $\{0,1\}^n$ and output space $\{0,1\}^m$ consists of two PPT algorithms:

- `KeyGen`. On input the security parameter 1^λ, the key generation algorithm outputs a PRF key k.
- `Eval`. On input the PRF key k and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs an output $y \in \{0,1\}^m$.

Also, it satisfies the following conditions:

- **Pseudorandomness.** Let $k \leftarrow \texttt{KeyGen}(1^\lambda)$, and f be a random function from $\{0,1\}^n$ to $\{0,1\}^m$. Also, let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $\texttt{Eval}(k,x)$, and let $\mathcal{O}_2(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $f(x)$. Then for all PPT adversary $\mathcal{A}$, we have:

$$|\Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)=1] - \Pr[\mathcal{A}^{\mathcal{O}_2(\cdot)}(1^\lambda)=1]| \leq negl(\lambda)$$

Puncturable Pseudorandom Function. A puncturable pseudorandom function [SW14] with key space $\mathcal{K}$, input space $\{0,1\}^n$ and output space $\{0,1\}^m$ consists of four PPT algorithms:

- $\texttt{KeyGen}$. On input the security parameter 1^λ, the key generation algorithm outputs the secret key $k \in \mathcal{K}$.
- $\texttt{Eval}$. On input the secret key $k \in \mathcal{K}$ and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs a string $y \in \{0,1\}^m$.
- $\texttt{Constrain}$. On input the secret keys $k \in \mathcal{K}$ and an input $x \in \{0,1\}^n$, the constrain algorithm outputs a punctured key ck.
- $\texttt{ConstrainEval}$. On input the punctured key ck and an input $x \in \{0,1\}^n$, the constrained evaluation algorithm outputs a string $y \in \{0,1\}^m$.

Also, it satisfies the following conditions:

- **Correctness.** For any $k \in \mathcal{K}$, any $x^* \in \{0,1\}^n$, and any $x \in \{0,1\}^n \backslash \{x^*\}$, let $ck \leftarrow \texttt{Constrain}(k,x^*)$, then we have $\texttt{ConstrainEval}(ck,x) = \texttt{Eval}(k, x)$.
- **Pseudorandomness.** Let $k \leftarrow \texttt{KeyGen}(1^\lambda)$, and f be a random function from $\{0,1\}^n$ to $\{0,1\}^m$. Also, let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $\texttt{Eval}(k,x)$, and let $\mathcal{O}_2(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $f(x)$. Then for all PPT adversary $\mathcal{A}$, we have:

$$|\Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda)=1] - \Pr[\mathcal{A}^{\mathcal{O}_2(\cdot)}(1^\lambda)=1]| \leq negl(\lambda)$$

- **Constrained Pseudorandomness.** For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have

$$\Pr\left[\begin{array}{l} (x^*, state) \leftarrow \mathcal{A}_1(1^\lambda); \\ k \leftarrow \texttt{KeyGen}(1^\lambda); \\ ck \leftarrow \texttt{Constrain}(k, x^*); \\ b \xleftarrow{\$} \{0,1\}; \\ y_0 = \texttt{Eval}(k, x^*); \\ y_1 \xleftarrow{\$} \{0,1\}^m; \end{array} : \mathcal{A}_2(ck, y_b, state) = b \right] \leq 1/2 + negl(\lambda)$$

Invoker Randomizable Pseudorandom Function. An invoker randomizable PRF [BGI+01] with input space $\{0,1\}^n \times \{0,1\}^m$ and output space $\{0,1\}^m$ consists of two PPT algorithms:

- `KeyGen`. On input the security parameter 1^λ, the key generation algorithm outputs a PRF key k.
- `Eval`. On input the PRF key k and an input pair $(x, r) \in \{0,1\}^n \times \{0,1\}^m$, the evaluation algorithm outputs an output $y \in \{0,1\}^m$.

Also, it satisfies the following conditions:

- **Pseudorandomness.** Let $k \leftarrow \texttt{KeyGen}(1^\lambda)$, and f be a random function from $\{0,1\}^{n+m}$ to $\{0,1\}^m$. Also, let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input $(x, r) \in \{0,1\}^n \times \{0,1\}^m$ and returns $\texttt{Eval}(k, x, r)$, and let $\mathcal{O}_2(\cdot)$ be an oracle that takes as input $(x, r) \in \{0,1\}^n \times \{0,1\}^m$ and returns $f(x\|r)$. Then for all PPT adversary $\mathcal{A}$, we have:

$$| \Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_2(\cdot)}(1^\lambda) = 1] | \leq negl(\lambda)$$

- **Invoker Randomization.** For any $k \leftarrow \texttt{KeyGen}(1^\lambda)$ and any $x \in \{0,1\}^n$, the mapping $r \mapsto \texttt{Eval}(k, x, r)$ is a permutation over $\{0,1\}^m$. This implies that for any $k \leftarrow \texttt{KeyGen}(1^\lambda)$ and any $x \in \{0,1\}^n$, if r is sampled uniformly at random from $\{0,1\}^m$, then the value $\texttt{Eval}(k, x, r)$ is distributed uniformly in $\{0,1\}^m$.

As shown in [BGI+01], we can construct invoker randomizable function from any secure pseudorandom function.

B.3 Public Key Encryption

A public Key Encryption scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{CT}$ consists of three PPT algorithms:

- `KeyGen`. On input the security parameter 1^λ, the key generation algorithm outputs a public key pk and a secret key sk.
- `Enc`. On input the public key pk and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext $ct \in \mathcal{CT}$.
- `Dec`. On input the secret key sk and a ciphertext $ct \in \mathcal{CT}$, the decryption algorithm outputs a message $m \in \mathcal{M} \cup \{\perp\}$.

Also, it satisfies the following conditions:

- **Perfect Correctness.** For any message $m \in \mathcal{M}$, we have

$$\Pr[(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda), ct \leftarrow \texttt{Enc}(pk, m) : \texttt{Dec}(sk, ct) \neq m] = 0$$

- **Security.** For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have:

$$\Pr\left[\begin{array}{l} (pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda); \\ (m_0^*, m_1^*, state) \leftarrow \mathcal{A}_1(pk); \\ b \xleftarrow{\$} \{0,1\}; \\ ct^* \leftarrow \texttt{Enc}(pk, m_b^*) \end{array} : \mathcal{A}_2(ct^*, state) = b\right] \leq 1/2 + negl(\lambda)$$

Besides, we need the PKE scheme to have pseudorandom ciphertexts, which requires:

- **Ciphertext Pseudorandomness.** For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have:

$$\Pr\left[\begin{array}{ll} (pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda); & \\ (m^*, state) \leftarrow \mathcal{A}_1(pk); & \\ b \xleftarrow{\$} \{0,1\}; & : \mathcal{A}_2(ct_b^*, state) = b \\ ct_0^* \leftarrow \mathtt{Enc}(pk, m^*); & \\ ct_1^* \xleftarrow{\$} \mathcal{CT} & \end{array}\right] \leq 1/2 + negl(\lambda)$$

As shown in [Reg05], we can construct a secure PKE scheme with pseudorandom ciphertexts from the standard LWE assumption. The scheme does not have perfect correctness. Nonetheless, it is easy to achieve the property via truncating the error terms in the scheme to be of bounded size. This will not affect security of the scheme. In addition, we can use the trick proposed in Remark D.1 to transform a pseudorandom ciphertext of the scheme into a pseudorandom binary string. The transformation is needed when we use the PKE scheme in the construction provided in Sec. 5.3.

B.4 Statistically Sound NIZK Proof

A non-interactive zero-knowledge proof [BFM88] for a language $\mathcal{L}$ consists of three PPT algorithms:

- $\mathtt{KeyGen}$. On input the security parameter 1^λ, the common reference string generation algorithm outputs a common reference string crs.
- $\mathtt{Prove}$. On input the common reference string crs, a statement $x \in \mathcal{L}$ and a witness w for x, the proving algorithm outputs a proof π.
- $\mathtt{Verify}$. On input the common reference string crs, a statement x and a proof π, the verification algorithm outputs a bit indicating whether the proof is valid.

Also, it satisfies the following conditions:

- **Completeness.** For any statement $x \in \mathcal{L}$, and any valid witness w for x, let $crs \leftarrow \mathtt{KeyGen}(1^\lambda)$ and $\pi \leftarrow \mathtt{Prove}(crs, x, w)$, then we have $\Pr[\mathtt{Verify}(crs, x, \pi) = 1] = 1$.
- **Statistical Soundness.** We have:

$$\Pr[crs \leftarrow \mathtt{KeyGen}(1^\lambda) : \exists(x, \pi) \text{ s.t. } \mathtt{Verify}(crs, x, \pi) = 1 \wedge x \notin \mathcal{L}] \leq negl(\lambda)$$

- **Adaptively Zero-Knowledge.** There exists a PPT simulator $(\mathtt{S}_1, \mathtt{S}_2)$ that for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have:

$$\Pr\left[\begin{array}{l} b \leftarrow \{0,1\}; \\ crs_0 \leftarrow \mathtt{KeyGen}(1^\lambda); \\ (crs_1, td) \leftarrow \mathtt{S}_1(1^\lambda); \\ (x, w, state) \leftarrow \mathcal{A}_1(crs_b); : b = b' \\ \pi_0 \leftarrow \mathtt{Prove}(crs_0, x, w); \\ \pi_1 \leftarrow \mathtt{S}_2(crs_1, x, td); \\ b' \leftarrow \mathcal{A}_2(\pi_b, state); \end{array}\right] \leq 1/2 + negl(\lambda)$$

  Here $\mathcal{A}_1$ is required to output a valid statement/witness pair.

As shown in [PS19], we can construct statistically sound NIZK proofs for any NP language from the standard LWE assumption.

B.5 The Jump Finding Algorithm

We will use the jump finding algorithm [BCP14, NWZ16] in our construction of message-embedding watermarking scheme. We recall the algorithm and its properties in Lemma B.7.

Lemma B.7 ([NWZ16]). *Let κ, q be positive integers. Let δ, φ be real values in $[0,1]$ s.t. $\delta \geq \varphi(2 + (\kappa - 1)q)$. Let $\mathcal{C} \subset (0, 2^\kappa - 1]$ be a set of q elements and let $\mathtt{P} : [0, 2^\kappa - 1] \to [0,1]_{\mathbb{R}}$[14] be an oracle that:*

1. *$|\mathtt{P}(2^\kappa - 1) - \mathtt{P}(0)| > \delta$.*
2. *For any $a, b \in [0, 2^\kappa - 1]$ s.t. $a < b$ and $(a, b] \cap \mathcal{C} = \emptyset$, $|\mathtt{P}(a) - \mathtt{P}(b)| < \varphi$.*

The algorithm $\mathtt{Trace}^{\mathtt{P}}(0, 2^\kappa - 1)$, which is described in Figure 6, will run in time $poly(\kappa, q)$ (and makes at most $q \cdot (\kappa + 1)$ distinct queries to $\mathtt{P}$) and will output a non-empty subset of $\mathcal{C}$.

$\mathtt{Trace}^{\mathtt{P}}$

Input: ind_1, ind_2

1. $p_1 = \mathtt{P}(ind_1)$.
2. $p_2 = \mathtt{P}(ind_2)$.
3. $\Delta = |p_1 - p_2|$.
4. If $\Delta \leq \varphi$: **Return** $\emptyset$.
5. If $ind_2 - ind_1 = 1$: **Return** $\{ind_2\}$.
6. $ind_3 = \lfloor \frac{ind_1 + ind_2}{2} \rfloor$.
7. **Return** $\mathtt{Trace}(ind_1, ind_3) \cup \mathtt{Trace}(ind_3, ind_2)$.

Fig. 6 The algorithms $\mathtt{Trace}^{\mathtt{P}}$.

[14] We use $[0,1]_{\mathbb{R}}$ to denote all real values r s.t. $0 \leq r \leq 1$.

B.6 Secret-Key Watermarkable PRFs

A secret-key watermarkable PRF family with public extraction [CHN+16], which has key space $\mathcal{K}$, input space $\mathcal{X}$, output space $\mathcal{Y}$, and message space $\mathcal{M}$ consists of the following algorithms:

- $\mathtt{Setup}(1^\lambda) \to (PP, MK, EK)$: On input the security parameter 1^λ, the setup algorithm outputs the public parameter PP, the mark key MK and the extraction key EK.
- $\mathtt{KeyGen}(PP) \to k$: On input the public parameter PP, the key generation algorithm outputs a PRF key $k \in \mathcal{K}$.
- $\mathtt{Eval}(PP, k, x) \to y$: On input the public parameter PP, a PRF key $k \in \mathcal{K}$, and an input $x \in \mathcal{X}$, the evaluation algorithm outputs an output $y \in \mathcal{Y}$.
- $\mathtt{Mark}(PP, MK, k, msg) \to \mathtt{C}$: On input the public parameter PP, the mark key MK, a PRF key $k \in \mathcal{K}$, and a message $msg \in \mathcal{M}$, the marking algorithm outputs a marked circuit $\mathtt{C} : \mathcal{X} \to \mathcal{Y}$.
- $\mathtt{Extract}(PP, EK, \mathtt{C}) \to msg$: On input the public parameter PP, the extraction key EK, and a circuit $\mathtt{C}$, the extraction algorithm outputs a message $msg \in \mathcal{M} \cup \{\perp\}$, where $\perp$ denotes that the circuit is unmarked.

Also, it satisfies the following conditions:

- **Functionality Preserving.** For any $msg \in \mathcal{M}$, let $(PP, MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$, $k \leftarrow \mathtt{KeyGen}(PP)$, $\mathtt{C} \leftarrow \mathtt{Mark}(PP, MK, k, msg)$, and $x \xleftarrow{\$} \mathcal{X}$, then we have
$$\Pr[\mathtt{C}(x) \neq \mathtt{Eval}(PP, k, x)] \leq negl(\lambda)$$
- **Extraction Correctness.** For any $msg \in \mathcal{M}$, let $(PP, MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$, $k \leftarrow \mathtt{KeyGen}(PP)$, and $\mathtt{C} \leftarrow \mathtt{Mark}(PP, MK, k, msg)$, then we have
$$\Pr[\mathtt{Extract}(PP, EK, \mathtt{C}) \neq msg] \leq negl(\lambda)$$
and
$$\Pr[\mathtt{Extract}(PP, EK, \mathtt{Eval}(PP, k, \cdot)) \neq \perp] \leq negl(\lambda)$$
- **Watermarking Meaningfulness.** For any circuit $\mathtt{C} : \mathcal{X} \to \mathcal{Y}$, let $(PP, MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$, then we have
$$\Pr[\mathtt{Extract}(PP, EK, \mathtt{C}) \neq \perp] \leq negl(\lambda)$$
- **Pseudorandomness.** Let $(PP, MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$, $k \leftarrow \mathtt{KeyGen}(PP)$, and f be a random function from $\mathcal{X}$ to $\mathcal{Y}$. Also, let $\mathcal{O}_0(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $\mathtt{Eval}(PP, k, x)$, and let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $f(x)$. Then for all PPT adversary $\mathcal{A}$, we have:
$$|\Pr[\mathcal{A}^{\mathcal{O}_0(\cdot)}(PP, MK, EK) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(PP, MK, EK) = 1]| \leq negl(\lambda)$$

- Q**-Bounded** ϵ**-Unremovability.** For any PPT adversary $\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{l}(PP, MK, EK) \leftarrow \mathtt{Setup}(1^\lambda);\\ k^* \leftarrow \mathtt{KeyGen}(PP);\\ \tilde{\mathtt{C}} \leftarrow \mathcal{A}^{\mathcal{O}_{mark}(\cdot,\cdot),\mathcal{O}_{cha}(\cdot)}(PP, EK);\end{array} : \mathtt{Extract}(PP, EK, \tilde{\mathtt{C}}) \notin \mathcal{Q}^*\right] \leq negl(\lambda)$$

  Here, the marking oracle $\mathcal{O}_{mark}(\cdot,\cdot)$ takes as input a PRF key $k \in \mathcal{K}$ and a message $msg \in \mathcal{M}$ and returns a circuit $\mathtt{C} \leftarrow \mathtt{Mark}(PP, MK, k, msg)$; and the challenge oracle $\mathcal{O}_{cha}(\cdot)$ takes as input a message $msg \in \mathcal{M}$ and returns a circuit $\mathtt{C}^* \leftarrow \mathtt{Mark}(PP, MK, k^*, msg)$. Also, we use $\mathcal{Q}^*$ to denote all messages submitted to the challenge oracle and use $\mathcal{R}^*$ to denote all circuits returned by the challenge oracle. In addition, we require that $\mathcal{A}$ is only allowed to query the challenge oracle for at most Q times, and that there exists circuit $\mathtt{C}^* \in \mathcal{R}^*$ that $|\{x \in \mathcal{X} : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq \epsilon \cdot |\mathcal{X}|$.[15]

C Functional Encryption with Strong Correctness

In this section, we define and construct the FE scheme needed.

The Definition. A functional encryption scheme [BSW11, O'N10] for a function family $\mathcal{F}$ with message space $\mathcal{M}$ and ciphertext space $\mathcal{CT}$ consists of four PPT algorithms:

- `Setup`. On input the security parameter 1^λ, the setup algorithm outputs the master public key/master secret key pair (mpk, msk).
- `KeyGen`. On input the master public key mpk, the master secret key msk and a function $f \in \mathcal{F}$, the key generation algorithm outputs a secret key fsk for f.
- `Enc`. On input the master public key mpk and a message $msg \in \mathcal{M}$, the encryption algorithm outputs a ciphertext $ct \in \mathcal{CT}$.
- `Dec`. On input the master public key mpk, the secret key fsk for a function f and a ciphertext $ct \in \mathcal{CT}$, the decryption algorithm outputs a message msg, where msg is either in the output space of f or is the decryption failure symbol $\perp$.

Also, it satisfies the following conditions:

- **Perfect Correctness.** For any message $msg \in \{0,1\}^m$ and any $f \in \mathcal{F}$, let $(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda)$, $fsk \leftarrow \mathtt{KeyGen}(mpk, msk, f)$, and $ct \leftarrow \mathtt{Enc}(mpk, msg)$, then we have $\Pr[\mathtt{Dec}(mpk, fsk, ct) = f(msg)] = 1$.

[15] We can also define $negl(\lambda)$-unremovability, which requires that there exists circuit $\mathtt{C}^* \in \mathcal{R}^*$ that $|\{x \in \mathcal{X} : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq negl(\lambda) \cdot |\mathcal{X}|$.

- **Q-Adaptive Indistinguishability.** For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have:

$$\Pr\left[\begin{array}{l}(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda);\\ (msg_0, msg_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{msk}(\cdot)}(mpk);\\ b \leftarrow \{0,1\}; \qquad\qquad : b = b'\\ ct \leftarrow \mathtt{Enc}(mpk, msg_b);\\ b' \leftarrow \mathcal{A}_2(ct, state);\end{array}\right] \leq 1/2 + negl(\lambda)$$

  Here, $\mathcal{O}_{msk}$ takes as input a function $f \in \mathcal{F}$ and outputs a secret key $fsk \leftarrow \mathtt{KeyGen}(mpk, msk, f)$. We require that for all f submitted to the oracle $\mathcal{O}_{msk}$, $f(msg_0) = f(msg_1)$ and the oracle $\mathcal{O}_{msk}$ can only be queried for Q times.

Besides, we need the FE scheme to have a stronger correctness, which requires:

- **Strong Correctness.** For each master public key/master secret key pair (mpk, msk), there exists a valid ciphertext set $\mathcal{V}_{mpk,msk}$ s.t.
  1. For any $f \in \mathcal{F}$, we have

$$\begin{aligned}\Pr[(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda), fsk \leftarrow \mathtt{KeyGen}(mpk, msk, f) :\\ \exists ct \in \mathcal{CT} \backslash \mathcal{V}_{mpk,msk}, \mathtt{Dec}(mpk, fsk, ct) \neq \perp] \leq negl(\lambda)\end{aligned}$$

  2. Assume that the identity function $\mathtt{ID} \in \mathcal{F}$ (for any $msg \in \mathcal{M}$, $\mathtt{ID}(msg) = msg$), then for any $f \in \mathcal{F}$, we have

$$\begin{aligned}\Pr[(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda), fsk_f \leftarrow \mathtt{KeyGen}(mpk, msk, f),\\ fsk_{\mathtt{ID}} \leftarrow \mathtt{KeyGen}(mpk, msk, \mathtt{ID}) : \exists ct \in \mathcal{V}_{mpk,msk},\\ \mathtt{Dec}(mpk, fsk_f, ct) \neq f(\mathtt{Dec}(mpk, fsk_{\mathtt{ID}}, ct))] \leq negl(\lambda)\end{aligned}$$

Remark C.1. We use *density* of FE to denote the fraction of honestly generated ciphertexts in the ciphertext space of the FE scheme. For technical reason, we consider the ratio between the distribution of honestly generated ciphertexts and the distribution of random ciphertexts (instead of the ratio between the sizes of the supports of these two distribution).

Formally, let $\mathcal{E}_{mpk}$ be the output distribution of the algorithm $\mathtt{Enc}$ on input mpk and a random message. Let $\mathcal{C}$ be the uniform distribution over $\mathcal{CT}$. We use the Rényi divergence [Rén61, VEH14, LSS14] to measure the ratio between $\mathcal{E}_{mpk}$ and $\mathcal{C}$, which is

$$RD_{mpk} = \max_{x \in \mathcal{CT}} \frac{\mathcal{E}_{mpk}(x)}{\mathcal{C}(x)}$$

We also define

$$\rho = \frac{1}{\max_{mpk} RD_{mpk}}$$

as the *density* of FE.

Remark C.2. We assume that the decryption algorithm of FE is deterministic. Note that for an FE scheme with probabilistic decryption algorithm and perfect correctness, it is safe to fix its decryption randomness without affecting its correctness and security.

Remark C.3. We define Q-adaptive indistinguishability in the single-challenge setting, i.e., the adversary is only allowed to submit one pair of challenge messages and receives one challenge ciphertext. It is easy to see that by a standard hybrid argument, the single-challenge security implies Q-adaptive indistinguishability in the multiple-challenge setting, where the adversary is allowed to submit n pairs of challenge messages and receives n challenge ciphertexts, for any polynomial n.

The Construction. Existing FE schemes does not have strong correctness and next we show how to upgrade an FE scheme with perfect correctness and Q-adaptive indistinguishability (but without strong correctness) into an FE scheme with all three properties listed above. Our construction is inspired by the construction of puncturable functional encryption with similar strong correctness requirement presented in [YAL+19].

Let λ be the security parameter.

Let $\mathsf{FE}_0 = (\mathsf{FE}_0.\mathtt{Setup}, \mathsf{FE}_0.\mathtt{KeyGen}, \mathsf{FE}_0.\mathtt{Enc}, \mathsf{FE}_0.\mathtt{Dec})$ be an FE scheme for a function family $\mathcal{F}$.

Let $\mathsf{NIZK} = (\mathsf{NIZK}.\mathtt{KeyGen}, \mathsf{NIZK}.\mathtt{Prove}, \mathsf{NIZK}.\mathtt{Verify})$ be a statistically sound NIZK proof system for $\mathcal{L}$, where

$$\mathcal{L} = \{(mpk, ct) : \exists (msg, r), ct = \mathsf{FE}_0.\mathtt{Enc}(mpk, msg; r)\}.$$

We construct the FE scheme $\mathsf{FE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ for $\mathcal{F}$ as follows:

- **Setup.** On input the security parameter 1^λ, the setup algorithm generates $(mpk, msk) \leftarrow \mathsf{FE}_0.\mathtt{Setup}(1^\lambda)$ and $crs \leftarrow \mathsf{NIZK}.\mathtt{KeyGen}(1^\lambda)$. Then it outputs the master public key $MPK = (mpk, crs)$ and the master secret key $MSK = msk$.
- **KeyGen.** On input the master public key $MPK = (mpk, crs)$, the master secret key $MSK = msk$, and a function $f \in \mathcal{F}$, the key generation algorithm generates $fsk \leftarrow \mathsf{FE}_0.\mathtt{KeyGen}(mpk, msk, f)$ and outputs $FSK = fsk$.
- **Enc.** On input the master public key $MPK = (mpk, crs)$ and a message msg, the encryption algorithm first samples randomness r for the algorithm $\mathsf{FE}_0.\mathtt{Enc}$. Then, it computes $ct = \mathsf{FE}_0.\mathtt{Enc}(mpk, msg; r)$, and $\pi \leftarrow \mathsf{NIZK}.\mathtt{Prove}(crs, (mpk, ct), (msg, r))$. Finally, it outputs $CT = (ct, \pi)$.
- **Dec.** On input the master public key $MPK = (mpk, crs)$, the secret key $FSK = fsk$ and a ciphertext $CT = (ct, \pi)$, the decryption algorithm first checks the validity of π and outputs $\perp$ if $\mathsf{NIZK}.\mathtt{Verify}(crs, (mpk, ct), \pi) = 0$. Otherwise, it outputs $\mathsf{FE}_0.\mathtt{Dec}(mpk, fsk, ct)$.

Theorem C.1. *If FE_0 is a secure functional encryption for $\mathcal{F}$ with perfect correctness and Q-adaptive indistinguishability and NIZK is a statistically sound*

NIZK proof system for language $\mathcal{L}$, then FE *is a secure functional encryption for $\mathcal{F}$ with perfect correctness, Q-adaptive indistinguishability, and strong correctness.*

Proof. We prove Theorem C.1 by proving the perfect correctness, adaptive indistinguishability, and strong correctness of FE.

Perfect Correctness. The perfect correctness of FE comes from the perfect correctness of FE_0 and the completeness of NIZK directly.

Adaptive Indistinguishability. The Q-adaptive indistinguishability of FE comes from the adaptively zero-knowledge property of NIZK and the Q-adaptive indistinguishability of FE_0 by direct reductions.

Strong Correctness. Finally, to argue the strong correctness of FE, we define

$$\mathcal{V}_{MPK=(mpk,crs),MSK} = \{CT = (ct, \pi) : \mathsf{NIZK}.\mathtt{Verify}(crs, (mpk, ct), \pi) = 1\}$$

for any $(MPK, MSK) \leftarrow \mathtt{Setup}(1^\lambda)$.

Let $(mpk, msk) \leftarrow \mathsf{FE}_0.\mathtt{Setup}(1^\lambda)$, $crs \leftarrow \mathsf{NIZK}.\mathtt{KeyGen}(1^\lambda)$, $MPK = (mpk, crs)$ and $MSK = msk$. Also, for any function $f \in \mathcal{F}$, let $fsk_{\mathtt{ID}} \leftarrow \mathsf{FE}_0.\mathtt{KeyGen}(mpk, msk, \mathtt{ID})$, $fsk_f \leftarrow \mathsf{FE}_0.\mathtt{KeyGen}(mpk, msk, f)$, $FSK_{\mathtt{ID}} = fsk_{\mathtt{ID}}$ and $FSK_f = fsk_f$.

Firstly, for any ciphertext $CT = (ct, \pi) \notin \mathcal{V}_{MPK,MSK}$, the decryption algorithm will always output $\perp$ since $\mathsf{NIZK}.\mathtt{Verify}(crs, (mpk, ct), \pi) = 0$.

Secondly, by the statistical soundness of NIZK, we have $(mpk, ct) \in \mathcal{L}$ for all $CT = (ct, \pi) \in \mathcal{V}_{MPK,MSK}$ with all but negligible probability. This implies that for each $CT = (ct, \pi) \in \mathcal{V}_{MPK,MSK}$, there exists msg, r s.t. $ct = \mathsf{FE}_0.\mathtt{Enc}(mpk, msg; r)$. Then by the perfect correctness of FE_0, we have $\mathsf{FE}_0.\mathtt{Dec}(mpk, fsk_{\mathtt{ID}}, ct) = msg$ and $\mathsf{FE}_0.\mathtt{Dec}(mpk, fsk_f, ct) = f(msg)$. Thus, with all but negligible probability, we have $\mathtt{Dec}(MPK, FSK_f, CT) = f(\mathtt{Dec}(MPK, FSK_{\mathtt{ID}}, CT))$ for all $CT \in \mathcal{V}_{MPK,MSK}$.

□

Remark C.4. Looking ahead, we can use the trick proposed in Remark D.1 to transform ciphertexts of the constructed FE scheme into binary strings. Let ρ and ρ' be the density of the original scheme and the new scheme respectively, then we have $\rho' \geq (1 - \frac{1}{2^\lambda}) \cdot \rho$. Thus, the parameter of the scheme constructed in Sec. 5.3 will not be affected after the transformation.

The Instantiations. As shown in [GVW12], for any polynomial Q, we can construct FE scheme for all polynomial-size circuits with perfect correctness and Q-adaptive indistinguishability from a secure PKE scheme with perfect correctness and a randomized encoding [Yao86, IK00, AIK06] with perfect correctness, small locality and low degree. Such randomized encoding can be instantiated from the SIS assumption as shown in [AIK04, AIK06]. Thus, we can construct such FE schemes from standard lattice assumptions. Also, as shown in [PS19], we can construct statistically sound NIZK proofs for any NP language from the

standard LWE assumption. Thus, we can instantiate FE schemes with perfect correctness, Q-adaptive indistinguishability, and strong correctness from standard lattice assumptions. However, when instantiating our construction with existing FE and NIZK proofs, the scheme has an exponentially-small ciphertext density (although the encryption algorithm could output exponentially-many ciphertexts). To summarize, we have

Corollary C.1. *Assuming the worst-case hardness of appropriately parameterized GapSVP problem, there exist FE schemes for all polynomial-size circuits with perfect correctness, $poly(\lambda)$-adaptive indistinguishability, strong correctness, and ciphertext density $\frac{2^{poly(\lambda)}}{2^n} \leq \rho \leq \frac{1}{2^{poly(\lambda)}}$, where n is the length of the ciphertext.*

D Special Fully Homomorphic Encryption

In this section, we define and construct the special FHE scheme needed. We remark that the syntax and construction details are quite different from that in the construction overview presented in Sec. 2.3. Nonetheless, the high-level ideas are identical.

D.1 The Definition

In this section, we define the special FHE scheme needed in our construction of robust unobfuscatable PRF. Generally, the definition is similar to previous definitions of FHE schemes [Gen09,BV11,GSW13], with some additional algorithms and properties.

In more detail, the scheme additionally has a public key randomization algorithm and an evaluation key randomization algorithm, which rerandomize the public key and the evaluation key respectively. We require that outputs of the two algorithms (as well as ciphertexts of the scheme) should be pseudorandom. Besides, the scheme has an algorithm that transforms its ciphertexts, where the transformed ciphertext can also be decrypted correctly by the secret key (with a modified decryption algorithm). In addition, if the encrypted message is random, the transformed ciphertext should be indistinguishable from a random sting even if the distinguisher has the *secret key*.

Formally, the special FHE scheme SFHE consists of the following PPT algorithms, where we use $\mathcal{C}$, $\mathcal{RPK}$, $\mathcal{REK}$, $\bar{\mathcal{C}}$ to denote the output spaces of algorithms $\mathtt{Enc}$, $\mathtt{RandPK}$, $\mathtt{RandEK}$, and $\mathtt{TranCT}$ respectively.

- $\mathtt{KeyGen}$. On input the security parameter 1^λ, the key generation algorithm outputs a public key pk, a secret key sk, and an evaluation key ek.
- $\mathtt{Enc}$. On input the public key pk and a message $msg \in \{0,1\}$, the encryption algorithm outputs a ciphertext $ct \in \mathcal{C}$.
- $\mathtt{Dec}$. On input the secret key sk and a ciphertext $ct \in \mathcal{C}$, the decryption algorithm outputs a message $msg \in \{0,1\}$.

- $\texttt{Eval}$. For any polynomial ℓ_{in}, ℓ_{out}, on input a (randomized) evaluation key ek, a vector of ℓ_{in} ciphertexts $\boldsymbol{ct}_x$, and a circuit $\texttt{C} : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell_{out}}$, the evaluation algorithm outputs a vector of ℓ_{out} ciphertexts $\boldsymbol{ct}_y \in \mathcal{C}^{\ell_{out}}$.

- $\texttt{RandPK}$. On input the public key pk, the public key randomization algorithm outputs a randomized public key $\mathit{rpk} \in \mathcal{RPK}$.
- $\texttt{RandEK}$. On input the public key pk and the evaluation key ek, the evaluation key randomization algorithm outputs a randomized evaluation key $ek' \in \mathcal{REK}$.
- $\texttt{TranCT}$. On input a randomized public key $\mathit{rpk} \in \mathcal{RPK}$ and a ciphertext $ct \in \mathcal{C}$, the ciphertext transform algorithm outputs a transformed ciphertext $\mathit{tct} \in \bar{\mathcal{C}}$.
- $\texttt{TCTDec}$.On input the secret key sk and a transformed ciphertext $\mathit{tct} \in \bar{\mathcal{C}}$, the decryption algorithm for transformed ciphertexts outputs a message $msg \in \{0,1\}$. Here, we require that the algorithm is *deterministic*.

Also, we define the following notions and properties for SFHE:

- **Valid Ciphertext.** A ciphertext $ct \in \mathcal{C}$ is a "valid ciphertext" for a message $msg \in \{0,1\}$ under keys $(pk, sk, ek) \leftarrow \texttt{KeyGen}(1^\lambda)$ if one of the following two conditions is satisfied:
  1. $ct \leftarrow \texttt{Enc}(pk, msg)$.
  2. $ct = \boldsymbol{ct}_y[i]$, where
     (a) $\boldsymbol{ct}_y \leftarrow \texttt{Eval}(ek', \boldsymbol{ct}_x, \texttt{C})$;
     (b) Either $ek' = ek$ or $ek' \leftarrow \texttt{RandEK}(pk, ek)$.
     (c) $\boldsymbol{ct}_x$ is a valid ciphertexts vector of x (i.e., the j-th ciphertext in the vector $\boldsymbol{ct}_x$ is a valid ciphertext of $x[j]$);
     (d) $\texttt{C}(x)[i] = msg$.
- **Perfect Correctness.** For any message $msg \in \{0,1\}$, for any $(pk, sk, ek) \leftarrow \texttt{KeyGen}(1^\lambda)$, and for any $\mathit{rpk} \leftarrow \texttt{RandPK}(pk)$, let ct be a valid ciphertext of msg under (pk, sk, ek), then we have

$$\Pr[\texttt{Dec}(sk, ct) \neq msg] = 0$$

and

$$\Pr[\texttt{TCTDec}(sk, \texttt{TranCT}(\mathit{rpk}, ct)) \neq msg] = 0$$

- **Ciphertext and Key Pseudorandomness.** Let $(pk, sk, ek) \leftarrow \texttt{KeyGen}(1^\lambda)$, then for all PPT adversary $\mathcal{A}$, we have:

$$\begin{aligned} |\Pr[\mathcal{A}^{\mathcal{O}^0_{enc}(\cdot), \mathcal{O}^0_{rpk}(), \mathcal{O}^0_{rek}()}(pk, ek) = 1] - \\ \Pr[\mathcal{A}^{\mathcal{O}^1_{enc}(\cdot), \mathcal{O}^1_{rpk}(), \mathcal{O}^1_{rek}()}(pk, ek) = 1]| \leq negl(\lambda) \end{aligned}$$

Here we define the oracles that can be queried by the adversary as follows:
- $\mathcal{O}^0_{enc}$ is an oracle that takes as input a message $msg \in \{0,1\}$ and returns $\texttt{Enc}(pk, msg)$.
- $\mathcal{O}^0_{rpk}$ is an oracle that returns $\texttt{RandPK}(pk)$.

- $\mathcal{O}^0_{rek}$ is an oracle that returns $\texttt{RandEK}(pk, ek)$.
- $\mathcal{O}^1_{enc}$ is an oracle that takes as input a message $msg \in \{0,1\}$ and returns $ct \xleftarrow{\$} \mathcal{C}$.
- $\mathcal{O}^1_{rpk}$ is an oracle that returns $\mathit{rpk} \xleftarrow{\$} \mathcal{RPK}$.
- $\mathcal{O}^1_{rek}$ is an oracle that returns $ek' \xleftarrow{\$} \mathcal{REK}$.

- **Bad Randomness for RandPK.** Let $\{0,1\}^{l_{rpk}}$ be the randomness space for the algorithm RandPK. We define $\mathcal{B} \subset \{0,1\}^{l_{rpk}}$ as the set of "bad randomness"[16] and require that

$$\Pr[r_{rpk} \xleftarrow{\$} \{0,1\}^{l_{rpk}} : r_{rpk} \in \mathcal{B}] \leq negl(\lambda) \tag{2}$$

- **Dividing the Ciphertext Space.** For any $(pk, sk, ek) \leftarrow \texttt{KeyGen}(1^\lambda)$, we can divide the output space $\bar{\mathcal{C}}$ of TranCT into two subsets $\mathcal{C}_0$ and $\mathcal{C}_1$[17], s.t.

$$\mathcal{C}_0 \cap \mathcal{C}_1 = \emptyset \tag{3}$$

$$|\frac{|\mathcal{C}_0|}{|\bar{\mathcal{C}}|} - \frac{1}{2}| \leq negl(\lambda) \qquad |\frac{|\mathcal{C}_1|}{|\bar{\mathcal{C}}|} - \frac{1}{2}| \leq negl(\lambda) \tag{4}$$

  where the set C_b for $b \in \{0,1\}$ is actually a set that is close to the set of transformed ciphertexts encrypting b.
- **Uniformity of Transformed Ciphertext.** For any $msg \in \{0,1\}$ and any $r_{rpk} \in \{0,1\}^{l_{rpk}} \backslash \mathcal{B}$, let $(pk, sk, ek) \leftarrow \texttt{KeyGen}(1^\lambda)$ and $\mathit{rpk} = \texttt{RandPK}(pk; r_{rpk})$. Also, let ct be a valid ciphertext of msg, then we have

$$(pk, sk, ek, r_{rpk}, ct, msg, \mathit{tct}_0) \approx_s (pk, sk, ek, r_{rpk}, ct, msg, \mathit{tct}_1)$$

  where

$$\mathit{tct}_0 \leftarrow \texttt{TranCT}(\mathit{rpk}, ct) \quad \text{and} \quad \mathit{tct}_1 \xleftarrow{\$} \mathcal{C}_{msg}$$

  The following fact can be implied by properties of $\mathcal{C}_0$ and $\mathcal{C}_1$ directly.

Corollary D.1. *For any $(pk, sk, ek) \leftarrow \texttt{KeyGen}(1^\lambda)$, let $b \xleftarrow{\$} \{0,1\}$, $\mathit{tct}_1 \xleftarrow{\$} \mathcal{C}_b$, and $\mathit{tct}_2 \xleftarrow{\$} \bar{\mathcal{C}}$, then we have $\mathit{tct}_1 \approx_s \mathit{tct}_2$.*

Proof.

$$\begin{aligned}
&\texttt{SD}(\mathit{tct}_1, \mathit{tct}_2) \\
=&\frac{1}{2} \cdot (|\mathcal{C}_0| \cdot |\frac{1}{2} \cdot \frac{1}{|\mathcal{C}_0|} - \frac{1}{|\bar{\mathcal{C}}|}| + |\mathcal{C}_1| \cdot |\frac{1}{2} \cdot \frac{1}{|\mathcal{C}_1|} - \frac{1}{|\bar{\mathcal{C}}|}| + (|\bar{\mathcal{C}}| - (|\mathcal{C}_0| + |\mathcal{C}_1|)) \cdot \frac{1}{|\bar{\mathcal{C}}|}) \\
=&\frac{1}{2} \cdot (|\frac{1}{2} - \frac{|\mathcal{C}_0|}{|\bar{\mathcal{C}}|}| + |\frac{1}{2} - \frac{|\mathcal{C}_1|}{|\bar{\mathcal{C}}|}| + (1 - \frac{|\mathcal{C}_0| + |\mathcal{C}_1|}{|\bar{\mathcal{C}}|})) \\
\leq&negl(\lambda)
\end{aligned}$$

where the first equality comes from Equation (3) the last inequality comes from Equation (4). □

[16] The set $\mathcal{B}$ is used below in defining the "uniformity of transformed ciphertext" property.

[17] The subsets $\mathcal{C}_0$ and $\mathcal{C}_1$ are used below in defining the "uniformity of transformed ciphertext" property.

D.2 The Construction

Next, we construct the special FHE scheme defined above. The starting point is a variant of the GSW homomorphic encryption scheme presented in [GSW13], where the dimension-modulus reduction technique proposed in [BV11] is applied to reduce the decryption complexity in the bootstrapping procedure. Then we show how to rerandomize the public key and the evaluation key for this scheme. We also design a ciphertext transform algorithm and the corresponding decryption algorithm for the transformed ciphertext.

Let λ be the security parameter. Let $k, p, n, m, \bar{m}, \hat{l}, l, K, N, \hat{\sigma}, \sigma, \widehat{\Sigma}$ be positive integers that are polynomial in λ and Σ', Σ, q be positive integers that are quasi-polynomial in λ, which satisfy: p, q are odd prime numbers, $n > k$, $\hat{l} = \lceil \log p \rceil$, $l = \lceil \log q \rceil$, $K = k\hat{l}$, $\bar{m} = (n+1)l + \lambda$, $m = \bar{m}l + \lambda$, $N = (n+1)l$, $\widehat{\Sigma} = nl((nl+1)(\lambda \cdot \hat{\sigma} + 1) + 1)$, $\Sigma' = 2(N+1)^{\omega(\log k)}$, $\Sigma = m \cdot \lambda \cdot \sigma \cdot (l \cdot \Sigma' + \bar{m}) + 2$, $p > 8\widehat{\Sigma}$, and $q \geq 2^{\omega(\log \lambda)} \cdot p \cdot \Sigma$. Without loss of generality, we use integers in $[0, q-1]$ (resp. $[0, p-1]$) to represent elements in $\mathbb{Z}_q$ (resp. $\mathbb{Z}_p$).

Let

$$\boldsymbol{G} = \begin{bmatrix} 1 & 2 & 4 & \dots & 2^{l-1} & & & & & & & & & & \\ & & & & & 1 & 2 & 4 & \dots & 2^{l-1} & & & & & \\ & & & & & & & & & & \ddots & & & & \\ & & & & & & & & & & & & 1 & 2 & 4 & \dots & 2^{l-1} \end{bmatrix}$$

be a matrix in $\mathbb{Z}^{(n+1)\times N}$. Let $\boldsymbol{G}^{-1} : \mathbb{Z}_q^{n+1} \to \{0,1\}^N$ be a (bijective) function that decomposes each element of a vector into bits. In particular, for any vector $\boldsymbol{v} \in \mathbb{Z}_q^{n+1}$, let $\bar{\boldsymbol{v}} = \boldsymbol{G}^{-1}(\boldsymbol{v}) \in \{0,1\}^N$, then $\boldsymbol{v}[i] = \sum_{j=1}^{l}(2^{j-1} \cdot \bar{\boldsymbol{v}}[(i-1)l + j])$ for $i \in [1, n+1]$. It is easy to see $\boldsymbol{G} \cdot \boldsymbol{G}^{-1}(\boldsymbol{v}) = \boldsymbol{v}$ for any vector $\boldsymbol{v} \in \mathbb{Z}_q^n$. For any positive integer ℓ and for any matrix $\boldsymbol{V} \in \mathbb{Z}_q^{n\times\ell}$, we also write

$$\boldsymbol{G}^{-1}(\boldsymbol{V}) = \begin{pmatrix} \boldsymbol{G}^{-1}(\boldsymbol{v}_1) & \boldsymbol{G}^{-1}(\boldsymbol{v}_2) & \dots & \boldsymbol{G}^{-1}(\boldsymbol{v}_\ell) \end{pmatrix}$$

where $\boldsymbol{v}_i$ is the $i-th$ column of $\boldsymbol{V}$. Note that we still have $\boldsymbol{G} \cdot \boldsymbol{G}^{-1}(\boldsymbol{V}) = \boldsymbol{V}$.

Let $\boldsymbol{h} \in \{0,1\}^l$ be the binary decomposition of $\frac{q+1}{2}$, i.e., $\sum_{j=1}^{l} 2^{j-1} \cdot \boldsymbol{h}[j] = \frac{q+1}{2}$. Let $\boldsymbol{G}'$ be the last l columns of the matrix $\boldsymbol{G}$, then

$$\boldsymbol{G}' \cdot \boldsymbol{h} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \frac{q+1}{2} \end{pmatrix}$$

We construct the special FHE scheme $\mathsf{SFHE} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Eval}, \mathtt{RandPK}, \mathtt{RandEK}, \mathtt{TranCT}, \mathtt{TCTDec})$ as follows:

- $\mathtt{KeyGen}$. On input the security parameter 1^λ, the key generation algorithm first samples $\boldsymbol{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e} \leftarrow \tilde{\mathcal{D}}_\sigma^m$, and computes

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{s}^\intercal \boldsymbol{B} + \boldsymbol{e}^\intercal \end{pmatrix} \mod q$$

Then, the algorithm samples $\hat{\boldsymbol{t}} \xleftarrow{\$} \mathbb{Z}_p^k$. Also, for $i \in [1, n]$, $\iota \in [1, l]$, it samples $\hat{\boldsymbol{a}}_{i,\iota} \xleftarrow{\$} \mathbb{Z}_p^k$, $\hat{e}_{i,\iota} \leftarrow \tilde{\mathcal{D}}_{\hat{\sigma}}$, and computes

$$\hat{\boldsymbol{\psi}}_{i,\iota} = \begin{pmatrix} \hat{\boldsymbol{a}}_{i,\iota} \\ \\ \hat{\boldsymbol{a}}_{i,\iota} \cdot \hat{\boldsymbol{t}} + \hat{e}_{i,\iota} + \lfloor \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \rceil \end{pmatrix} \mod p$$

Besides, the algorithm decomposes the vector $\hat{\boldsymbol{t}}$ into a K-bit string $\bar{\boldsymbol{t}}$ and for $i \in [1, K]$, it samples $\check{\boldsymbol{R}}_i \xleftarrow{\$} \{0,1\}^{m \times N}$ and computes

$$\check{\boldsymbol{\Psi}}_i = \bar{\boldsymbol{t}}[i] \cdot \boldsymbol{G} + \boldsymbol{A}\check{\boldsymbol{R}}_i \mod q$$

Finally, it outputs the public key $pk = \boldsymbol{A}$, the secret key $sk = \boldsymbol{s}$ and the evaluation key $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$.

- $\mathtt{Enc}$. On input a public key $pk = \boldsymbol{A}$ and a message $\mu \in \{0,1\}$, the encryption algorithm samples $\boldsymbol{R} \xleftarrow{\$} \{0,1\}^{m \times N}$ and computes

$$\boldsymbol{C} = \mu \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R} \mod q$$

Then it outputs the ciphertext $ct = \boldsymbol{C}$.

- $\mathtt{Dec}$. On input a secret key $sk = \boldsymbol{s}$ and a ciphertext $ct = \boldsymbol{C} \in \mathbb{Z}_q^{(n+1)\times N}$, the decryption algorithm first sets $\boldsymbol{C}'$ as the last l columns of $\boldsymbol{C}$ and computes $\boldsymbol{c} = \boldsymbol{C}' \cdot \boldsymbol{h}$. Then it sets $w = \boldsymbol{c}[n+1]$ and $\boldsymbol{v} = \boldsymbol{c}[1:n]$ and computes

$$u = w - \boldsymbol{s} \cdot \boldsymbol{v} \mod q$$

It output 1 if $|u - \frac{q+1}{2}| \leq \Sigma$ and outputs 0 otherwise.

- $\mathtt{Eval}$. For any ℓ_{in}, ℓ_{out} that are polynomial in λ, the input of the evaluation algorithm includes a (randomized) evaluation key $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$, a vector of ℓ_{in} ciphertext $\boldsymbol{ct}_x = (\boldsymbol{C}_i)_{i\in[1,\ell_{in}]} \in (\mathbb{Z}_q^{(n+1)\times N})^{\ell_{in}}$, and a circuit $\mathtt{C} : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell_{out}}$.
Let $|\mathtt{C}|$ be the number of wires for the circuit $\mathtt{C}$ and label each wire of $\mathtt{C}$ with a number in $[1, |\mathtt{C}|]$, where each wire has a larger label than its children. We can label the input wires as $1, \ldots, \ell_{in}$. Also, we can label the output wires as $|\mathtt{C}| - \ell_{out} + 1, \ldots, |\mathtt{C}|$, where the i-th output wire is labeled with $|\mathtt{C}| - \ell_{out} + i$. For $i \in [\ell_{in} + 1, |\mathtt{C}|]$, let i_L and i_R be the labels of the children of the wire labelled with i, then the evaluation algorithm computes

$$\boldsymbol{C}_i = \mathtt{DecNAND}(ek, \boldsymbol{C}_{i_L}, \boldsymbol{C}_{i_R})$$

where we define the auxiliary algorithm $\mathtt{DecNAND}$ below. Finally, the evaluation algorithm **outputs**

$$\boldsymbol{ct}_y = (\boldsymbol{C}_i)_{i\in[|\mathtt{C}|-\ell_{out}+1,|\mathtt{C}|]} \in (\mathbb{Z}_q^{(n+1)\times N})^{\ell_{out}}$$

Description of auxiliary algorithms. We next describe a few auxiliary algorithms that are used in the evaluation algorithm. The algorithm $\mathtt{DecNAND}$

takes as input two ciphertexts, which encrypt μ_0, μ_1 respectively, and outputs a ciphertext that encrypts $\mu_0 \barwedge \mu_1$. As the bootstrapping method is used, the error term of the output ciphertext is fixed. The algorithm $\mathtt{NAND}$ also performs the NAND operation over the input ciphertexts. But, the error term in the output ciphertext will increase. The algorithm $\mathtt{RedCT}$ reduce the dimension and modulus of a ciphertext. In particular, it transforms a ciphertext under $\boldsymbol{s}$, which is in $\mathbb{Z}_q^{(n+1)\times N}$ into a ciphertext under $\boldsymbol{t}$, which is in $\mathbb{Z}_p^{k+1}$.

– $\mathtt{DecNAND}$. On input a (randomized) evaluation key $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$, and two ciphertexts $\boldsymbol{C}_0, \boldsymbol{C}_1 \in \mathbb{Z}_q^{(n+1)\times N}$, the algorithm first computes

$$(\hat{\boldsymbol{v}}_0, \hat{w}_0) \leftarrow \mathtt{RedCT}(ek, \boldsymbol{C}_0)$$
$$(\hat{\boldsymbol{v}}_1, \hat{w}_1) \leftarrow \mathtt{RedCT}(ek, \boldsymbol{C}_1)$$

where we define the algorithm $\mathtt{RedCT}$ below. Then it sets $\mathtt{D} = \bar{\mathtt{D}}[\hat{\boldsymbol{v}}_0, \hat{w}_0, \hat{\boldsymbol{v}}_1, \hat{w}_1]$, where $\bar{\mathtt{D}}$ is defined in Figure 7.
Let $|\mathtt{D}|$ be the number of wires for the circuit $\mathtt{D}$ and label each wire of $\mathtt{D}$ with a number in $[1, |\mathtt{D}|]$, where each wire has a larger label than its children. We can label the input wires as $1, \ldots, K$. Also, we can label the output wire as $|\mathtt{D}|$.
For $i \in [1, K]$, let $\boldsymbol{D}_i = \check{\boldsymbol{\Psi}}_i$. For $i \in [K+1, |\mathtt{D}|]$, let i_L and i_R be the labels of the children of the wire labelled with i, then the algorithm computes

$$\boldsymbol{D}_i = \mathtt{NAND}(\boldsymbol{D}_{i_L}, \boldsymbol{D}_{i_R})$$

where the algorithm $\mathtt{NAND}$ is defined below.
Finally, it outputs $\boldsymbol{D}_{|\mathtt{D}|}$.

– $\mathtt{RedCT}$. On input a (randomized) evaluation key $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$, and a ciphertext $\boldsymbol{C} \in \mathbb{Z}_q^{(n+1)\times N}$, the algorithm first sets $\boldsymbol{C}'$ as the last l columns of $\boldsymbol{C}$ and computes $\boldsymbol{c} = \boldsymbol{C}' \cdot \boldsymbol{h}$. Then it sets $w = \boldsymbol{c}[n+1]$ and $\boldsymbol{v} = \boldsymbol{c}[1:n]$.
Let $v_{i,\iota}$ be the ι-th bit of $\boldsymbol{v}[i]$, i.e., $\boldsymbol{v}[i] = \sum_{\iota\in[1,l]} 2^{\iota-1} \cdot v_{i,\iota}$. Also, for $i \in [1, n]$, $\iota \in [1, l]$, let $\hat{\boldsymbol{a}}_{i,\iota} = \hat{\boldsymbol{\psi}}_{i,\iota}[1:k]$ and let $\hat{b}_{i,\iota} = \hat{\boldsymbol{\psi}}_{i,\iota}[k+1]$.
Then, the algorithm outputs

$$\hat{w} = \lfloor \frac{p}{q} w \rceil - \sum_{i=1}^{n} \sum_{\iota=1}^{l} v_{i,\iota} \cdot \hat{b}_{i,\iota} \mod p$$

and

$$\hat{\boldsymbol{v}} = -\sum_{i=1}^{n} \sum_{\iota=1}^{l} v_{i,\iota} \cdot \hat{\boldsymbol{a}}_{i,\iota} \mod p$$

– $\mathtt{NAND}$. On input two ciphertexts $\boldsymbol{C}_0, \boldsymbol{C}_1 \in \mathbb{Z}_q^{(n+1)\times N}$, the algorithm outputs

$$\boldsymbol{G} - \boldsymbol{C}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1) \mod q$$

$\bar{\mathtt{D}}$

Constant: $\hat{\boldsymbol{v}}_0, \hat{w}_0, \hat{\boldsymbol{v}}_1, \hat{w}_1$

Input: $\boldsymbol{t} \in \{0,1\}^K$

1. $u_0 = \widehat{\Sigma} + \hat{w}_0 - \sum_{i=1}^{k} \sum_{\iota=1}^{\hat{l}} 2^{\iota-1} \cdot \hat{\boldsymbol{v}}_0[i] \cdot \boldsymbol{t}[(i-1)\hat{l} + \iota] \mod p.$
2. $u_1 = \widehat{\Sigma} + \hat{w}_1 - \sum_{i=1}^{k} \sum_{\iota=1}^{\hat{l}} 2^{\iota-1} \cdot \hat{\boldsymbol{v}}_1[i] \cdot \boldsymbol{t}[(i-1)\hat{l} + \iota] \mod p.$
3. If $u_0 \geq 2^{\hat{l}-2}$, $\mu_0 = 1$; otherwise, $\mu_0 = 0$.
4. If $u_1 \geq 2^{\hat{l}-2}$, $\mu_1 = 1$; otherwise, $\mu_1 = 0$.
5. Output $\mu_0 \barwedge \mu_1$.

Fig. 7 The circuit $\bar{\mathtt{D}}$.

- `RandPK`. On input a public key $pk = \boldsymbol{A}$, the public key randomization algorithm samples $\boldsymbol{R} \xleftarrow{\$} \{0,1\}^{m \times \bar{m}}$ and computes

$$\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{R} \mod q$$

  Then it outputs the randomized public key $rp\mathcal{k} = \boldsymbol{A}'$.
- `RandEK`. The input of the evaluation key randomization algorithm includes a public key $pk = \boldsymbol{A}$ and an evaluation key $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$. First, for $i \in [1,n]$ and $\iota \in [1,l]$, the algorithm sample $\widehat{\boldsymbol{R}}'_{i,\iota} \xleftarrow{\$} \{0,1\}^{m\times N}$, computes

$$\widehat{\boldsymbol{C}}'_{i,\iota} = \boldsymbol{A}\widehat{\boldsymbol{R}}'_{i,\iota} \mod q$$

  and obtains

$$(\hat{\boldsymbol{v}}_{i,\iota}, \hat{w}_{i,\iota}) \leftarrow \mathtt{RedCT}(ek, \widehat{\boldsymbol{C}}'_{i,\iota})$$

  where `RedCT` is an auxiliary algorithm defined above in the evaluation algorithm. Then it computes

$$\hat{\boldsymbol{\psi}}'_{i,\iota} = \hat{\boldsymbol{\psi}}_{i,\iota} + \begin{pmatrix} \hat{\boldsymbol{v}}_{i,\iota} \\ \\ \hat{w}_{i,\iota} \end{pmatrix} \mod p$$

  Next, for $i \in [1,K]$, the algorithm samples $\check{\boldsymbol{R}}'_i \xleftarrow{\$} \{0,1\}^{m\times N}$ and computes

$$\check{\boldsymbol{\Psi}}'_i = \check{\boldsymbol{\Psi}}_i + \boldsymbol{A}\check{\boldsymbol{R}}'_i \mod q$$

  Finally, it outputs the randomized evaluation key $ek' = ((\hat{\boldsymbol{\psi}}'_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}'_i)_{i\in[1,K]})$.
- `TranCT`. On input a randomized public key $rp\mathcal{k} = \boldsymbol{A}'$ and a ciphertext $ct = \boldsymbol{C}' \in \mathbb{Z}_q^{(n+1)\times N}$, the ciphertext transform algorithm first sets $\boldsymbol{C}''$ as the last l columns of $\boldsymbol{C}'$ and computes $\boldsymbol{c}' = \boldsymbol{C}'' \cdot \boldsymbol{h}$. Next, it samples $\boldsymbol{x} \xleftarrow{\$} \{0,1\}^{\bar{m}}$, $z \xleftarrow{\$} [\Sigma, \frac{q+1}{2} - \Sigma]$, and computes

$$\boldsymbol{c} = \boldsymbol{c}' + \boldsymbol{A}'\boldsymbol{x} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ z \end{pmatrix} \mod q$$

Finally, it outputs the transformed ciphertext $\mathit{tct} = \boldsymbol{c}$.

- $\mathtt{TCTDec}$. On input a secret key $sk = \boldsymbol{s}$ and a transformed ciphertext $\mathit{tct} = \boldsymbol{c} \in \mathbb{Z}_q^{n+1}$, the decryption algorithm first sets $w = \boldsymbol{c}[n+1]$, $\boldsymbol{v} = \boldsymbol{c}[1:n]$ and computes

$$u = w - \boldsymbol{s} \cdot \boldsymbol{v} \mod q$$

It output 0 if $u \in [0, \frac{q+1}{2})$ and outputs 1 otherwise.

Theorem D.1. *Assume the circular security of the GSW homomorphic encryption scheme,*[18] *then* SFHE *satisfies all properties required in Appendix D.1.*

We present Proof of Theorem D.1 in Appendix E.5.

Remark D.1. In above construction, the outputs of the algorithms $\mathtt{Enc}$, $\mathtt{RandPK}$, $\mathtt{RandEK}$, and $\mathtt{TranCT}$ are matrices/vectors in $\mathbb{Z}_q$ or $\mathbb{Z}_p$. However, in the construction of unobfuscatable PRF given in Sec. 6.3, it is required that outputs of these algorithms are binary strings. The trivial solution that decomposes numbers in $\mathbb{Z}_q$ (or $\mathbb{Z}_p$) into bits does not work here. This is because we need the outputs to be (pseudo)random strings over the output spaces of the algorithms, but as p, q are not a power of 2, the decomposed binary string may not uniform even if the original output is a uniform matrix/vector over $\mathbb{Z}_q$ or $\mathbb{Z}_q$. For example, if $\lceil \log q \rceil - \log q \approx 1/2$, then the probability that the most significant bit of a random number in $\mathbb{Z}_q$ is 0 is about 0.7 (instead of 1/2).

We solve the problem in a general way, i.e. we do not impose any restriction on the selection of the parameters in above construction. Our main observation is that for any number Q, there exists a number $\tilde{Q}$ s.t. $\tilde{Q}$ is a multiple of Q and $\tilde{Q}$ is close to a power of 2. Let $l = \lceil \log Q \rceil$ and let $L = l + \lambda$. Then we set

$$\tilde{Q} = Q \cdot \lfloor \frac{2^L}{Q} \rfloor$$

Obviously, $\tilde{Q}$ is a multiple of Q and let $P = \tilde{Q}/Q$. Also, note that

$$\frac{2^L - \tilde{Q}}{2^L} = \frac{Q \cdot (\frac{2^L}{Q} - \lfloor \frac{2^L}{Q} \rfloor)}{2^L} \leq \frac{Q}{2^L} \leq \frac{1}{2^\lambda}$$

Now, let $r_Q \xleftarrow{\$} \mathbb{Z}_Q$ and $r_P \xleftarrow{\$} \mathbb{Z}_P$. Let $r_0 = r_P \cdot Q + r_Q$, and let $r_1 \xleftarrow{\$} \mathbb{Z}_{2^L}$ (i.e., the binary decomposition of r_1 is a uniform string in $\{0,1\}^L$). Then we have

$$\mathtt{SD}(r_0, r_1) = \frac{1}{2} \Big| \sum_{a \in [0, \tilde{Q}-1)} (\frac{1}{\tilde{Q}} - \frac{1}{2^L}) + \sum_{a \in [\tilde{Q}-1, 2^L-1)} \frac{1}{2^L} \Big| = \frac{2^L - \tilde{Q}}{2^L} \leq \frac{1}{2^\lambda} \tag{5}$$

In this way, we can approximately map a random element in $\mathbb{Z}_Q$ into a random L-bit string.

[18] We elaborate the assumption later in the proof of ciphertext and key pseudorandomness property.

Let $\mathcal{C}$, $\mathcal{RPK}$, $\mathcal{REK}$, and $\bar{\mathcal{C}}$ be the output spaces of algorithms $\mathtt{Enc}$, $\mathtt{RandPK}$, $\mathtt{RandEK}$, and $\mathtt{TranCT}$ respectively. Let $Q_c = |\mathcal{C}|$, $Q_p = |\mathcal{RPK}|$, $Q_e = |\mathcal{REK}|$, and $Q_{\bar{c}} = |\bar{\mathcal{C}}|$. Note that one can map between elements in $\mathcal{C}$, $\mathcal{RPK}$, $\mathcal{REK}$, $\bar{\mathcal{C}}$ and numbers in $\mathbb{Z}_{Q_c}$, $\mathbb{Z}_{Q_p}$, $\mathbb{Z}_{Q_e}$, $\mathbb{Z}_{Q_{\bar{c}}}$ efficiently. Thus, we can apply the above trick to map (pseudo)random outputs of the algorithms $\mathtt{Enc}$, $\mathtt{RandPK}$, $\mathtt{RandEK}$, and $\mathtt{TranCT}$ into (pseudo)random binary strings and recovers the original outputs from the binary strings if needed.

E Deferred Proofs

E.1 Security Analysis of Public-Key Hinting Watermarkable PRFs from Puncturable PRFs

We present proof of Theorem 5.1 in this section. More precisely, we will prove the functionality preserving property, extraction correctness, watermarking meaningfulness, pseudorandomness, and 1-bounded 1-unremovability of HWF.

Functionality Preserving. The functionality preserving property of HWF comes from correctness of PPRF directly.

Extraction Correctness. Let $w \xleftarrow{\$} \{0,1\}^\lambda$ and $PP = w$. Also, let $k \leftarrow \mathsf{PPRF}.\mathtt{KeyGen}(1^\lambda)$, $x^* \xleftarrow{\$} \{0,1\}^n$, $y^* = \mathsf{PPRF}.\mathtt{Eval}(k, x^*)$, $z^* = \mathtt{F}(y^*)$, $K = (k, x^*)$, and $\mathtt{hint} = (x^*, z^*, w)$. Besides, let $ck \leftarrow \mathsf{PPRF}.\mathtt{Constrain}(k, x^*)$ and let $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ be a circuit s.t. $\mathtt{C}(x) = \mathsf{PPRF}.\mathtt{ConstrainEval}(ck, x)$ for any $x \in \{0,1\}^n$.

Now, assume the extraction algorithm takes as input the watermarked circuit $\mathtt{C}$ and the hint $\mathtt{hint} = (x^*, z^*, w)$. First, by the constrained pseudorandomness of PPRF, $\mathtt{C}(x^*) \neq y^*$ with all but negligible probability. Next, by the injectivity of $\mathtt{F}$, $\mathtt{F}(C(x^*)) \neq z^*$ if $\mathtt{C}(x^*) \neq y^*$. Thus, with all but negligible probability we have $\mathtt{Extract}(PP, \mathtt{C}, \mathtt{hint}) =$"$\mathtt{marked}$".

On the other hand, assume the input circuit is $\mathtt{Eval}(PP, K, \cdot)$. As $\mathtt{Eval}(PP, K, x^*) = \mathsf{PPRF}.\mathtt{Eval}(k, x^*) = y^*$, we have $\mathtt{F}(\mathtt{Eval}(PP, K, x^*)) = z^*$. Thus, we always have $\mathtt{Extract}(PP, \mathtt{Eval}(PP, K, \cdot), \mathtt{hint}) = \perp$.

This completes the proof of extraction correctness.

Watermarking Meaningfulness. For any fixed circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ and for any fixed hint $\mathtt{hint} = (x, z, w')$, let $w \xleftarrow{\$} \{0,1\}^\lambda$. Then the probability that $w = w'$ is $1/2^\lambda$. Thus, the probability that a fixed pair of circuit and hint can pass the extraction algorithm (with random public parameter w) is negligible.

Pseudorandomness. Pseudorandomness of HWF comes from pseudorandomness of PPRF directly.

Unremovability. Next, we prove the unremovability of HWF. First, we define the following games between a challenger and a PPT 1-unremoving-admissible adversary $\mathcal{A}$:

- **Game 0.** This is the real experiment $\mathtt{ExptUR}$. More precisely, the challenger proceeds as follows.

I. First, the challenger samples $w \xleftarrow{\$} \{0,1\}^\lambda$, $k \leftarrow \mathsf{PPRF}.\mathtt{KeyGen}(1^\lambda)$, $x^* \xleftarrow{\$} \{0,1\}^n$, and computes $y^* = \mathsf{PPRF}.\mathtt{Eval}(k, x^*)$, $z^* = \mathtt{F}(y^*)$.

II. Then, it returns $PP = w$ and $\mathtt{hint}^* = (x^*, z^*, w)$ to the adversary. It also computes $ck \leftarrow \mathsf{PPRF}.\mathtt{Constrain}(k, x^*)$ and sets $\mathtt{C}^* : \{0,1\}^n \to \{0,1\}^m$ as a circuit that for any $x \in \{0,1\}^n$: $\mathtt{C}^*(x) = \mathsf{PPRF}.\mathtt{ConstrainEval}(ck, x)$. It returns $\mathtt{C}^*$ to the adversary as the response to the challenge oracle query[19].

III. Finally, after $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$, the challenger **outputs 1** if

$$z^* = \mathtt{F}(\tilde{\mathtt{C}}(x^*))$$

Otherwise, it **outputs 0**.

- **Game 1.** This is identical to Game 0 except that the challenger samples $y^* \xleftarrow{\$} \{0,1\}^m$.

Let $\mathcal{E}_i$ be the output of Game i and we next show that $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$ via proving the following lemmas.

Lemma E.1. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 0 and Game 1 comes from the constrained pseudorandomness of PPRF directly. □

Lemma E.2. $\Pr[\mathcal{E}_1 = 1] \leq negl(\lambda)$.

Proof. Game 1 outputs 1 iff $\mathtt{F}(\tilde{\mathtt{C}}(x^*)) = z^*$, where $z^* = \mathtt{F}(y^*)$ for a random string y^*. This will occur with a negligible probability due to the one-wayness of $\mathtt{F}$. □

Combining Lemma E.1 to Lemma E.2, we have $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$, i.e., the probability that $\mathcal{A}$ wins in the real experiment $\mathtt{ExptUR}$ is negligible. This completes the proof of unremovability.

E.2 Security Analysis of Public-Key Hinting Watermarkable PRFs from Functional Encryption

We present proof of Theorem 5.2 in this section. More precisely, we will prove the functionality preserving property, extraction correctness, watermarking meaningfulness, pseudorandomness, and unremovability of HWF.

Functionality Preserving. For any message $msg \in [1, 2^\kappa - 1]$, let $w \xleftarrow{\$} \{0,1\}^\lambda$, $t^* \xleftarrow{\$} \{0,1\}^\lambda$, and $PP = (w, t^*)$. Also, let $(mpk, msk) \leftarrow \mathsf{FE}.\mathtt{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{PKE}.\mathtt{KeyGen}(1^\lambda)$, $k \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$, $fsk \leftarrow \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, f_\perp)$, $K = (mpk, msk, pk, k, fsk)$ and $\mathtt{hint} = (mpk, sk, w)$. Besides, let $fsk_{msg} \leftarrow \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, f_{msg,t^*})$ and let $\mathtt{C} = \mathtt{M}[mpk, fsk_{msg}, pk, k]$. Let $fsk_{\mathtt{ID}} = \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, \mathtt{ID})$, where $\mathtt{ID}$ is the identity function.

[19] As there is only one message in the message space, the adversary only needs to query the challenge oracle once and the submitted message must be "$\mathtt{marked}$".

It is easy to see that for any $x \in \{0,1\}^n$, $\mathtt{C}(x) = \mathtt{Eval}(PP, K, x)$ if $\mathsf{FE}.\mathtt{Dec}(mpk, fsk, x) = \mathsf{FE}.\mathtt{Dec}(mpk, fsk_{msg}, x)$. Thus, it is sufficient to bound the probability that fsk_0 and fsk_1 decrypt a random ciphertext to different messages.

First, by the strong correctness of FE, with all but negligible probability over the choice of $mpk, msk, fsk, fsk_{msg}, fsk_{\mathtt{ID}}$, we have:

1. For any $x \notin \mathcal{V}_{mpk,msk}$, $\mathsf{FE}.\mathtt{Dec}(mpk, fsk, x) = \mathsf{FE}.\mathtt{Dec}(mpk, fsk_{msg}, x) = \perp$.
2. For any $x \in \mathcal{V}_{mpk,msk}$, let $(ind \| t \| \mu) = \mathsf{FE}.\mathtt{Dec}(mpk, fsk_{\mathtt{ID}}, x)$, then $\mathsf{FE}.\mathtt{Dec}(mpk, fsk, x) = \mathsf{FE}.\mathtt{Dec}(mpk, fsk_{msg}, x) = \mu$ if $t \neq t^*$.

It remains to show that for a random $x \in \mathcal{V}_{mpk,msk}$, the probability that $\mathsf{FE}.\mathtt{Dec}(mpk, fsk_{\mathtt{ID}}, x)[\kappa+1, \kappa+\lambda] = t^*$ is negligible.

To prove this, we define $\mathcal{V}_t = \{x \in \mathcal{V}_{mpk,msk} : \mathsf{FE}.\mathtt{Dec}(mpk, fsk_{\mathtt{ID}}, x)[\kappa+1, \kappa+\lambda] = t\}$ for any $t \in \{0,1\}^\lambda$ and define $\mathcal{L} = \{t \in \{0,1\}^\lambda : |\mathcal{V}_t|/|\mathcal{V}_{mpk,msk}| > 1/2^{\lambda/2}\}$. Then we have $|\mathcal{L}| \leq 2^{\lambda/2}$ (recall that $\mathsf{FE}.\mathtt{Dec}$ is a deterministic algorithm). Thus, $\Pr[t^* \in \mathcal{L}] \leq 2^{\lambda/2}/2^\lambda = 1/2^{\lambda/2}$, which is negligible. Now, assume that $t^* \notin \mathcal{L}$, which occurs with all but negligible probability, then we have $\Pr[x \xleftarrow{\$} \mathcal{V}_{mpk,msk} : \mathsf{FE}.\mathtt{Dec}(mpk, fsk_{\mathtt{ID}}, x)[\kappa+1, \kappa+\lambda] = t^*] \leq 1/2^{\lambda/2}$, which is negligible.

This completes the proof of functionality preserving.

Extraction Correctness. For any message $msg \in [1, 2^\kappa - 1]$, let $w \xleftarrow{\$} \{0,1\}^\lambda$, $t^* \xleftarrow{\$} \{0,1\}^\lambda$, and $PP = (w, t^*)$. Also, let $(mpk, msk) \leftarrow \mathsf{FE}.\mathtt{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{PKE}.\mathtt{KeyGen}(1^\lambda)$, $k \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$, $fsk \leftarrow \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, f_\perp)$, $K = (mpk, msk, pk, k, fsk)$ and $\mathtt{hint} = (mpk, sk, w)$. Besides, let $fsk_{msg} \leftarrow \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, f_{msg,t^*})$ and let $\mathtt{C} = \mathtt{M}[mpk, fsk_{msg}, pk, k]$.

Now, assume the extraction algorithm takes as input the watermarked circuit $\mathtt{C}$ and the hint $\mathtt{hint} = (mpk, sk, w)$. Then by perfect correctness of FE and perfect correctness of PKE, we have $\mathtt{Test}(ind) = 1$ for all $ind < msg$ tested in the extraction algorithm; also, with all but negligible probability, we have $\mathtt{Test}(ind) = 0$ for all $ind \geq msg$ tested in the extraction algorithm. Next, by Lemma B.7, the $\mathtt{Trace}$ algorithm will output $\{msg\}$. Therefore, the extraction algorithm will output msg with all but negligible probability given $\mathtt{C}$ as input.

On the other hand, assume the input circuit is $\mathtt{Eval}(PP, K, \cdot)$. Then by perfect correctness of FE and perfect correctness of PKE, we have $\mathtt{Test}(0) = \mathtt{Test}(2^\kappa - 1) = 1$. Thus, the $\mathtt{Trace}$ algorithm will output $\emptyset$. Therefore, the extraction algorithm will output $\perp$ on input $\mathtt{Eval}(PP, K, \cdot)$.

This completes the proof of extraction correctness.

Watermarking Meaningfulness. For any fixed circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ and for any fixed hint $\mathtt{hint} = (mpk, sk, w')$, let $w \xleftarrow{\$} \{0,1\}^\lambda$. Then the probability that $w = w'$ is $1/2^\lambda$. Thus, the probability that a fixed pair of circuit and hint can pass the extraction algorithm (with random public parameter w) is negligible.

Pseudorandomness. Pseudorandomness of HWF comes from pseudorandomness of F and ciphertext pseudorandomness of PKE directly.

Unremovability. Next, we prove the unremovability of HWF. First, we define the following games between a challenger and a PPT ϵ-unremoving-admissible adversary $\mathcal{A}$:

- **Game 0.** This is the real experiment $\texttt{ExptUR}$. More precisely, the challenger proceeds as follows.
  - **I.** First, the challenger samples $w \xleftarrow{\$} \{0,1\}^\lambda$, $t^* \xleftarrow{\$} \{0,1\}^\lambda$, and generates $(mpk, msk) \leftarrow \mathsf{FE}.\texttt{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{PKE}.\texttt{KeyGen}(1^\lambda)$, and $k \leftarrow \mathsf{F}.\texttt{KeyGen}(1^\lambda)$.
  - **II.** Then, it returns $PP = (w, t^*)$ and $\texttt{hint}^* = (mpk, sk, w)$ to the adversary and answers the adversary's challenge oracle queries (for at most Q times) as follows:
    - On receiving a message msg, the challenger first computes $fsk_{msg} \leftarrow \mathsf{FE}.\texttt{KeyGen}(mpk, msk, f_{msg,t^*})$ and returns the circuit $\texttt{M}[mpk, fsk_{msg}, pk, k]$, where $\texttt{M}$ is defined in Figure 2.

    Here, we denote the messages submitted by the adversary as $msg_1^*, msg_2^*, \ldots, msg_Q^*$, where $msg_1^* \leq msg_2^* \leq \ldots \leq msg_Q^*$. Also, we set $\mathcal{Q}^* = \{msg_1^*, msg_2^*, \ldots, msg_Q^*\}$. In addition, for $i \in [1, Q]$, assuming msg_i^* is submitted in the ℓ_i-th challenge oracle query, we use $\texttt{C}_i^*$ to denote the response of the ℓ_i-th challenge oracle query and use fsk_i^* to denote the secrete key fsk_{msg} generated when answering this query. [20]
  - **III.** Finally, after $\mathcal{A}$ submits a circuit $\tilde{\texttt{C}}$, the challenger proceeds as follows, where the algorithms $\texttt{Test}, \texttt{Trace}$ are defined in Figure 3 and we also recall the two algorithms (with slight modification on the syntax) in Figure 8.
    1. Set the constant for the algorithm $\texttt{Test}$ as $(\tilde{\texttt{C}}, mpk, sk, t^*)$.
    2. $\mathfrak{p}_0 = \texttt{Test}(0)$.
    3. $\mathfrak{p}_{2^\kappa - 1} = \texttt{Test}(2^\kappa - 1)$.
    4. $\mathcal{M} \leftarrow \texttt{Trace}(0, 2^\kappa - 1, \mathfrak{p}_0, \mathfrak{p}_{2^\kappa - 1})$. Here, the extraction algorithm will abort and **output 1** if the test algorithms (including $\texttt{Test}$, $\texttt{Test}'$ and $\texttt{Test}''$) have been invoked for more than $Q \cdot (\kappa + 1)$ times in the $\texttt{Trace}$ algorithm.
    5. If $\mathcal{M} = \emptyset$, **output 1**.
    6. $msg \xleftarrow{\$} \mathcal{M}$.
    7. If $msg \notin \mathcal{Q}^*$, **output 1**.
    8. **Output 0**.
- **Game 1.** This is identical to Game 0 except that the challenger sets the constant for the algorithm $\texttt{Test}'$ as $(\tilde{\texttt{C}}, mpk, sk, t^*)$ and computes $\mathfrak{p}_0' = \texttt{Test}'()$ (the algorithm $\texttt{Test}'$ is defined in Figure 8) before computing $\mathfrak{p}_0$. Then it aborts and outputs 2 if $\mathfrak{p}_0' < \theta - \varphi$.
- **Game 2.** This is identical to Game 1 except that the challenger changes the way to compute $\mathfrak{p}_{ind}$ (including $\mathfrak{p}_0$ and some $\mathfrak{p}_{ind_3}$ calculated in the $\texttt{Trace}$ algorithm) in Phase III. In particular, it invokes $\texttt{Test}'()$ (with constant

[20] If repeated messages are submitted, e.g., there exists $1 \leq i < j \leq Q$ s.t. $msg_{i-1}^* < msg_i^* = \ldots = msg_j^* < msg_{j+1}^*$, we assume $\ell_i < \ldots < \ell_j$.

$(\tilde{\mathtt{C}}, mpk, sk, t^*)$) instead of $\mathtt{Test}(ind)$ if $ind < msg_1^*$, i.e., for all $\mathfrak{p}_{ind}$ s.t. $0 \leq ind < msg_1^*$, $\mathfrak{p}_{ind} = \mathtt{Test}'()$.

- **Game 3.** This is identical to Game 2 except that the challenger changes the way to compute $\mathfrak{p}_{ind}$ (including some $\mathfrak{p}_{ind_3}$ calculated in the $\mathtt{Trace}$ algorithm) in Phase III. In particular, it invokes $\mathtt{Test}(msg_i^*)$ (with constant $(\tilde{\mathtt{C}}, mpk, sk, t^*)$) instead of $\mathtt{Test}(ind)$ when there exists $i \in [1, Q-1]$ s.t. $msg_i^* \leq ind < msg_{i+1}^*$, i.e., for all $\mathfrak{p}_{ind}$ s.t. $msg_i^* \leq ind < msg_{i+1}^*$, $\mathfrak{p}_{ind} = \mathtt{Test}(msg_i^*)$.
- **Game 4.** This is identical to Game 3 except that the challenger changes the way to compute $\mathfrak{p}_{ind}$ (including $\mathfrak{p}_{2^\kappa-1}$ and some $\mathfrak{p}_{ind_3}$ calculated in the $\mathtt{Trace}$ algorithm) in Phase III. In particular, it sets the constant of $\mathtt{Test}''$ (the algorithm $\mathtt{Test}''$ is defined in Figure 8) as $(\tilde{\mathtt{C}}, mpk, sk, t^*)$ and invokes $\mathtt{Test}''(ind)$ instead of $\mathtt{Test}(ind)$ if $ind \geq msg_Q^*$, i.e., for all $\mathfrak{p}_{ind}$ s.t. $ind \geq msg_Q^*$, $\mathfrak{p}_{ind} = \mathtt{Test}''(ind)$.
- **Game 5.** This is identical to Game 4 except that the challenger aborts and outputs 2 if $\mathfrak{p}_0 < \theta - 2\varphi$.
- **Game 6.** Let $msg_0^* = 0$, then the challenger aborts and outputs 2 if there exists $i \in [0, Q-1]$ and $ind, ind' \in [msg_i^*, msg_{i+1}^*)$ s.t. $|\mathfrak{p}_{ind} - \mathfrak{p}_{ind'}| \geq \varphi$. It proceeds identically as in Game 5 if no such (i, ind, ind') exists.
- **Game 7.** This is identical to Game 6 except that the challenger aborts and outputs 2 if there exists $ind \geq msg_Q^*$ s.t. $\mathfrak{p}_{ind} \neq 0$.

Let $\mathcal{E}_i$ be the output of Game i and we next show that $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$ via proving the following lemmas.

Lemma E.3. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. Let

$$p = \Pr[(ind', t, \mu) \xleftarrow{\$} \{0,1\}^\kappa \times \{0,1\}^\lambda \backslash \{t^*\} \times \{0,1\}^\lambda, \\ x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu), y = \tilde{\mathtt{C}}(x) : \mathsf{PKE}.\mathtt{Dec}(sk, y) = \mu]$$

We first give a lower bound of p.

Claim E.1. $p \geq \theta - 1/(2^\lambda - 1)$.

Proof. For any $(ind', t, \mu) \in \{0,1\}^\kappa \times \{0,1\}^\lambda \backslash \{t^*\} \times \{0,1\}^\lambda$, let $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu)$ and $y = \tilde{\mathtt{C}}(x)$. Then, by perfect correctness of FE and perfect correctness of PKE, we have $\mathsf{PKE}.\mathtt{Dec}(sk, y) = \mu$ if $\tilde{\mathtt{C}}(x) = \mathtt{C}_1^*(x)$ (recall that $\mathtt{C}_1^*(x)$ is the watermarked circuit returned by the challenge oracle on receiving msg_1^*). Thus, we have

$$p \geq p' = \Pr[(ind', t, \mu) \xleftarrow{\$} \{0,1\}^\kappa \times \{0,1\}^\lambda \backslash \{t^*\} \times \{0,1\}^\lambda, \\ x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu) : \tilde{\mathtt{C}}(x) = \mathtt{C}_1^*(x)]$$

Trace

Input: $ind_1, ind_2, \mathfrak{p}_{ind_1}, \mathfrak{p}_{ind_2}$

1. $\Delta = |\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}|$.
2. If $\Delta \leq \varphi$:
   Return $\emptyset$.
3. If $ind_2 - ind_1 = 1$:
   Return $\{ind_2\}$.
4. $ind_3 = \lfloor \frac{ind_1 + ind_2}{2} \rfloor$.
5. $\mathfrak{p}_{ind_3} = \mathtt{Test}(ind_3)$.
6. **Return** $\mathtt{Trace}(ind_1, ind_3, \mathfrak{p}_{ind_1}, \mathfrak{p}_{ind_3}) \cup \mathtt{Trace}(ind_3, ind_2, \mathfrak{p}_{ind_3}, \mathfrak{p}_{ind_2})$.

Test

Constant: $\mathtt{E}, mpk, sk, t^*$

Input: ind

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) Sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$.
   (b) $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind \| t^* \| \mu)$.
   (c) $y = \mathtt{E}(x)$.
   (d) $\bar{\mu} = \mathsf{PKE}.\mathtt{Dec}(sk, y)$.
   (e) If $\mu = \bar{\mu}$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

Test′

Constant: $\mathtt{E}, mpk, sk, t^*$

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) Sample $ind' \xleftarrow{\$} \{0,1\}^\kappa$.
   (b) Sample $t \xleftarrow{\$} \{0,1\}^\lambda \backslash \{t^*\}$.
   (c) Sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$.
   (d) $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu)$.
   (e) $y = \mathtt{E}(x)$.
   (f) $\bar{\mu} = \mathsf{PKE}.\mathtt{Dec}(sk, y)$.
   (g) If $\mu = \bar{\mu}$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

Test″

Constant: $\mathtt{E}, mpk, sk, t^*$

Input: ind

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) Sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$.
   (b) $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind \| t^* \| 0^\lambda)$.
   (c) $y = \mathtt{E}(x)$.
   (d) $\bar{\mu} = \mathsf{PKE}.\mathtt{Dec}(sk, y)$.
   (e) If $\mu = \bar{\mu}$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

Fig. 8 The algorithms `Trace`, `Test`, `Test′`, and `Test″`. We use red underlines to highlight the differences between `Test′` and `Test` and the difference between `Test″` and `Test`.

Let $\mathcal{E}'$ be the distribution of x in the `Test′` algorithm, i.e., for any $x^* \in \{0,1\}^n$:

$$\mathcal{E}'(x^*) = \Pr[(ind', t, \mu) \xleftarrow{\$} \{0,1\}^\kappa \times \{0,1\}^\lambda \backslash \{t^*\} \times \{0,1\}^\lambda, \\ x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu) : x = x^*]$$

Let $\mathcal{R}$ be the randomness space for the algorithm $\mathsf{FE}.\mathtt{Enc}$ and let $\mathcal{E}_{mpk}$ be the output distribution of $\mathsf{FE}.\mathtt{Enc}(mpk, \cdot)$ on input a random message, then for any $x^* \in \{0,1\}^n$, we have:

$$\begin{aligned}
&\mathcal{E}'(x^*) \\
=&\frac{|\{(ind', t, \mu, r) \in \{0,1\}^\kappa \times \{0,1\}^\lambda \backslash \{t^*\} \times \{0,1\}^\lambda \times \mathcal{R} : x^* = \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu; r)\}|}{2^{\kappa+\lambda} \cdot (2^\lambda - 1) \cdot |\mathcal{R}|} \\
=&\frac{|\{(ind', t, \mu, r) \in \{0,1\}^\kappa \times \{0,1\}^\lambda \backslash \{t^*\} \times \{0,1\}^\lambda \times \mathcal{R} : x^* = \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu; r)\}|}{2^{\kappa+2\lambda} \cdot |\mathcal{R}|} \cdot \frac{2^\lambda}{2^\lambda - 1} \\
\leq&\frac{|\{(ind', t, \mu, r) \in \{0,1\}^\kappa \times \{0,1\}^\lambda \times \{0,1\}^\lambda \times \mathcal{R} : x^* = \mathsf{FE}.\mathtt{Enc}(mpk, ind' \| t \| \mu; r)\}|}{2^{\kappa+2\lambda} \cdot |\mathcal{R}|} \cdot \frac{2^\lambda}{2^\lambda - 1} \\
=&\mathcal{E}_{mpk}(x^*) \cdot \frac{2^\lambda}{2^\lambda - 1}
\end{aligned}$$

Let $\mathcal{D} = \{x \leftarrow \mathcal{E}' : \tilde{\mathtt{C}}(x) \neq \mathtt{C}_1^*(x)\}$. Note that for any $x \leftarrow \mathcal{E}'$, any $i \in [1, Q]$, $\mathtt{C}_i^*(x) = \mathtt{C}_1^*(x)$. Also, the adversary is ϵ-unremoving-admissibile, where $\epsilon \leq \rho \cdot (1-\theta)$, thus we have

$$\begin{aligned}
|\mathcal{D}| &\leq 2^n \cdot \rho \cdot (1-\theta) \\
&= 2^n \cdot 1/(2^n \cdot (\max_{mpk'} \max_{x \in \{0,1\}^n} \mathcal{E}_{mpk'}(x))) \cdot (1-\theta) \\
&\leq 2^n \cdot 1/(2^n \cdot \max_{x \in \{0,1\}^n} \mathcal{E}_{mpk}(x)) \cdot (1-\theta) \\
&= \frac{1-\theta}{\max_{x \in \{0,1\}^n} \mathcal{E}_{mpk}(x)}
\end{aligned}$$

Thus, we have

$$\begin{aligned}
1 - p' &= \sum_{x \in \mathcal{D}} \mathcal{E}'(x) \\
&\leq \sum_{x \in \mathcal{D}} \mathcal{E}_{mpk}(x) \cdot \frac{2^\lambda}{2^\lambda - 1} \\
&\leq (\max_{x \in \{0,1\}^n} \mathcal{E}_{mpk}(x) \cdot \frac{2^\lambda}{2^\lambda - 1}) \cdot |\mathcal{D}| \\
&\leq (\max_{x \in \{0,1\}^n} \mathcal{E}_{mpk}(x) \cdot \frac{2^\lambda}{2^\lambda - 1}) \cdot \frac{1-\theta}{\max_{x \in \{0,1\}^n} \mathcal{E}_{mpk}(x)} \\
&= \frac{2^\lambda}{2^\lambda - 1} \cdot (1-\theta) \\
&\leq 1 - \theta + \frac{1}{2^\lambda - 1}
\end{aligned}$$

i.e., $p \geq p' \geq \theta - \frac{1}{2^\lambda - 1}$. □

In each repetition of the $\mathtt{Test}'$ algorithm, *Acc* will increase with probability p. Then, for $i \in [1, T]$, let X_i be a random variable over $\{0, 1\}$ that $\Pr[X_i = 1] = p$ and let $X = \sum_{i=1}^{T} X_i$. Then by the Chernoff bound, we have

$$\Pr[X \leq (1 - \frac{\varphi}{2p})pT] \leq e^{-\frac{\varphi^2}{8p^2}pT} = e^{-\frac{\lambda}{8p}} \leq e^{-\frac{\lambda}{8}}$$

which is negligible. Thus with all but negligible probability, we have

$$\mathfrak{p}_0' > (1 - \frac{\varphi}{2p})p \geq \theta - \frac{1}{2^\lambda - 1} - \frac{\varphi}{2} > \theta - \varphi$$

Therefore, the challenger will output 2 with only negligible probability in Game 1. □

Lemma E.4. $|\Pr[\mathcal{E}_1 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$.

Proof. Game 1 and Game 2 are identical except that in these two games, the challenger uses different ways to generate the test points in the test algorithm when $ind < msg_1^*$.

In particular, in Game 1, the challenger will sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$ and compute $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind\|t^*\|\mu)$. In Game 2, the challenger will sample $ind' \xleftarrow{\$} \{0,1\}^\kappa$, $t \xleftarrow{\$} \{0,1\}^\lambda \backslash \{t^*\}$, $\mu \xleftarrow{\$} \{0,1\}^\lambda$, and compute $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind'\|t\|\mu)$.

Note that the adversary can only view at most Q secret keys of the FE scheme and for $j \in [1, Q]$, the secret key $fsk_j^* \leftarrow \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, f_{msg_j^*, t^*})$, where

$$f_{msg_j^*, t^*}(ind\|t^*\|\mu) = f_{msg_j^*, t^*}(ind'\|t\|\mu) = \mu$$

In addition, in the whole extraction procedure, only polynomially-many test points are calculated (as the test algorithm is only invoked for polynomially-many times). Thus, points generated in these two ways are computationally indistinguishable due to the Q-adaptive indistinguishability (in the multi-challenge setting) of FE. □

Lemma E.5. $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| \leq negl(\lambda)$.

Proof. Game 2 and Game 3 are identical except that in these two games, the challenger uses different ways to generate the test points in the test algorithm when $msg_i^* \leq ind < msg_{i+1}^*$ for some $i \in [1, Q-1]$.

In particular, in Game 2, the challenger will sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$ and compute $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind\|t^*\|\mu)$. In Game 3, the challenger will sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$ and compute $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, msg_i^*\|t^*\|\mu)$.

Note that the adversary can only view at most Q secret keys of the FE scheme and for $j \in [1, Q]$, the secret key $fsk_j^* \leftarrow \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, f_{msg_j^*, t^*})$, where for $j \leq i$

$$f_{msg_j^*, t^*}(ind\|t^*\|\mu) = f_{msg_j^*, t^*}(msg_i^*\|t^*\|\mu) = 0$$

and for $j \geq i+1$

$$f_{msg_j^*, t^*}(ind\|t^*\|\mu) = f_{msg_j^*, t^*}(msg_i^*\|t^*\|\mu) = \mu$$

In addition, in the whole extraction procedure, only polynomially-many test points are calculated. Thus, points generated in these two ways are computationally indistinguishable due to the Q-adaptive indistinguishability (in the multi-challenge setting) of FE. □

Lemma E.6. $|\Pr[\mathcal{E}_3 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. Game 3 and Game 4 are identical except that in these two games, the challenger uses different ways to generate the test points in the test algorithm when $ind \geq msg_Q^*$.

In particular, in Game 3, the challenger will sample $\mu \xleftarrow{\$} \{0,1\}^\lambda$ and compute $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind\|t^*\|\mu)$. In Game 4, the challenger will compute $x \leftarrow \mathsf{FE}.\mathtt{Enc}(mpk, ind\|t^*\|0^\lambda)$.

Note that the adversary can only view at most Q secret keys of the FE scheme and for $j \in [1, Q]$, the secret key $fsk_j^* \leftarrow \mathsf{FE}.\mathtt{KeyGen}(mpk, msk, f_{msg_j^*, t^*})$, where

$$f_{msg_j^*, t^*}(ind \| t^* \| \mu) = f_{msg_j^*, t^*}(ind \| t^* \| 0^\lambda) = 0^\lambda$$

In addition, in the whole extraction procedure, only polynomially-many test points are calculated. Thus, points generated in these two ways are computationally indistinguishable due to the Q-adaptive indistinguishability (in the multi-challenge setting) of FE. □

Lemma E.7. $|\Pr[\mathcal{E}_4 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$.

Proof. Game 4 and Game 5 are identical except that $\mathfrak{p}_0' \geq \theta - \varphi$ but $\mathfrak{p}_0 < \theta - 2\varphi$ in the games. Thus, it is sufficient to prove that $|\mathfrak{p}_0 - \mathfrak{p}_0'| \leq \varphi$ with all but negligible probability. As both $\mathfrak{p}_0 = \mathtt{Test}'()$ and $\mathfrak{p}_0' = \mathtt{Test}'()$, this can be implied by the following general claim :

Claim E.2. *For any $p \in [0, 1]$, for $i \in [1, T]$, let X_i be a random variable over $\{0, 1\}$ that $\Pr[X_i = 1] = p$ and let $X = \sum_{i=1}^{T} X_i$; also, for $i \in [1, T]$, let Y_i be a random variable over $\{0, 1\}$ that $\Pr[Y_i = 1] = p$ and let $Y = \sum_{i=1}^{T} Y_i$. Then we have $\Pr[|X - Y| \geq \varphi T] \leq negl(\lambda)$.*

Proof. First, by the Chernoff bound, we have

$$\Pr[X \geq (1 + \frac{\varphi}{2p})pT] \leq e^{-\frac{\varphi^2}{4p^2 \cdot (2 + \varphi/2p)}pT} = e^{-\frac{\lambda}{8p + 2\varphi}} \leq e^{-\frac{\lambda}{10}}$$

which is negligible. Similar, we have $\Pr[Y \geq (1 + \frac{\varphi}{2p})pT] \leq negl(\lambda)$.

Now, if $p \leq \frac{\varphi}{2}$, then with all but negligible probability, we have

$$0 \leq X < pT + \frac{\varphi}{2}T \leq \varphi T$$

Similar, we have $0 \leq Y < \varphi T$ with all but negligible probability. Thus, we have $|X - Y| < \varphi T$ with all but negligible probability.

Also, if $p > \frac{\varphi}{2}$, then by the Chernoff bound , we have

$$\Pr[X \leq (1 - \frac{\varphi}{2p})pT] \leq e^{-\frac{\varphi^2}{8p^2}pT} = e^{-\frac{\lambda}{8p}} \leq e^{-\frac{\lambda}{8}}$$

which is negligible. Similar, we have $\Pr[Y \leq (1 - \frac{\varphi}{2p})pT] \leq negl(\lambda)$. That is, with all but negligible probability we have

$$pT - \frac{\varphi}{2}T < X, Y < pT + \frac{\varphi}{2}T$$

Thus, we have $|X - Y| < \varphi T$ with all but negligible probability.

This completes proof of the Claim. □

□

Lemma E.8. $|\Pr[\mathcal{E}_5 = 1] - \Pr[\mathcal{E}_6 = 1]| \leq negl(\lambda)$.

Proof. First, for $ind, ind' \in [0, msg_1^*)$, $\mathfrak{p}_{ind} = \mathtt{Test}'()$ and $\mathfrak{p}_{ind'} = \mathtt{Test}'()$, thus, by Claim E.2, we have $\Pr[|\mathfrak{p}_{ind} - \mathfrak{p}_{ind'}| \geq \varphi] \leq negl(\lambda)$.

Also, for $i \in [1, Q-1]$, for $ind, ind' \in [msg_i^*, msg_{i+1}^*)$, $\mathfrak{p}_{ind} = \mathtt{Test}(msg_i^*)$ and $\mathfrak{p}_{ind'} = \mathtt{Test}(msg_i^*)$, thus, by Claim E.2, we have $\Pr[|\mathfrak{p}_{ind} - \mathfrak{p}_{ind'}| \geq \varphi] \leq negl(\lambda)$.

Since in the whole extraction procedure, only polynomially-many $\mathfrak{p}_{ind}$ are calculated (as the test algorithm is only invoked for polynomially-many times), the probability that there exists $i \in [0, Q-1]$ and $ind, ind' \in [msg_i^*, msg_{i+1}^*)$ s.t. $|\mathfrak{p}_{ind} - \mathfrak{p}_{ind'}| \geq \varphi$ is also negligible.

Thus, with all but negligible probability, the new abort condition introduced in Game 6 will not be triggered, and indistinguishability between Game 5 and Game 6 follows. □

Lemma E.9. $|\Pr[\mathcal{E}_6 = 1] - \Pr[\mathcal{E}_7 = 1]| \leq negl(\lambda)$.

Proof. In each repetition of the algorithm $\mathtt{Test}''$, μ is independent of $\bar{\mu}$ and is sampled uniformly at random from $\{0,1\}^\lambda$, thus, the probability that $\bar{\mu} = \mu$ is $1/2^\lambda$. Moreover, as T is polynomial in λ, the probability that $Acc \neq 0$ is also negligible. In addition, only polynomially-many $\mathfrak{p}_{ind}$ are calculated, thus we have $\mathfrak{p}_{ind} = 0$ for all $ind \geq msg_Q^*$ with all but negligible probability.

Thus, with all but negligible probability, the new abort condition introduced in Game 7 will not be triggered, and indistinguishability between Game 6 and Game 7 follows. □

Lemma E.10. $\Pr[\mathcal{E}_7 = 1] = 0$.

Proof. In Game 7, the challenger outputs 1 only if the $\mathtt{Trace}$ algorithm invokes the test algorithms for more than $Q \cdot (\kappa+1)$ times or if the $\mathtt{Trace}$ algorithm fails to output a non-empty subset of $\mathcal{Q}^*$.

Note that in Game 7, the $\mathtt{Trace}$ algorithm interacts with an oracle $\mathtt{P}$ that outputs $\mathfrak{p}_{ind}$ on a query ind. Assume that the game does not output 2, then we have $\mathtt{P}(0) \geq \theta - 2\varphi$ and $\mathtt{P}(2^\kappa - 1) = 0$, which implies that

$$|\mathtt{P}(2^\kappa - 1) - \mathtt{P}(0)| \geq \theta - 2\varphi = (3 + (\kappa - 1)Q)\varphi > (2 + (\kappa - 1)Q)\varphi$$

Also, for any $a, b \in [0, 2^\kappa - 1]$ s.t. $a < b$ and $(a, b] \cap \mathcal{Q}^* = \emptyset$, we have either of the following three cases:

- $0 \leq a < b < msg_1^*$, then we have $|\mathtt{P}(a) - \mathtt{P}(b)| < \varphi$.
- $\exists i \in [1, Q-1], msg_i^* \leq a < b < msg_{i+1}^*$, then we have $|\mathtt{P}(a) - \mathtt{P}(b)| < \varphi$.
- $msg_Q^* \leq a < b \leq 2^\kappa - 1$, then we have $|\mathtt{P}(a) - \mathtt{P}(b)| = 0$.

if the game does not output 2.

Thus, by Lemma B.7, the algorithm $\mathtt{Trace}$ will make at most $Q \cdot (\kappa + 1)$ distinct queries to $\mathtt{P}$ and output a non-empty subset of $\mathcal{Q}^*$ if the game does not output 2. That is, the game will not output 1. □

Combining Lemma E.3 to Lemma E.10, we have $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$, i.e., the probability that $\mathcal{A}$ wins in the real experiment $\texttt{ExptUR}$ is negligible. This completes the proof of unremovability.

Remark E.1. One may note that in our proof, we use the fact that the adversary is $(\rho \cdot (1-\theta))$-unremoving-admissibile in Game 0. The adversary is also $(\rho \cdot (1-\theta) + negl(\lambda))$-unremoving-admissibile in Game i for $i > 0$. However, as in our instantiation, ρ is a concrete negligible function, the parameter may be much larger than $\rho \cdot (1-\theta)$ in other games, which would not imply the fact that $\mathfrak{p}_0$ is large enough.

Note that this subtle issue can be omitted when considering an ϵ-unremoving-admissible adversary for some non-negligible ϵ or if ϵ is only required to be any (rather than a concrete) negligible function.

E.3 Security Analysis of Public-Key Hinting Watermarkable PRFs from Secret-Key Watermarkable PRFs

We present proof of Theorem 5.3 in this section. More precisely, we will prove the functionality preserving property, extraction correctness, watermarking meaningfulness, pseudorandomness, and unremovability of HWF.

Functionality Preserving. The functionality preserving property of HWF comes from the functionality preserving property of $\mathsf{SK\text{-}WPRF}$ directly.

Extraction Correctness. Let $(PP = w) \leftarrow \texttt{Setup}(1^\lambda)$ and $(K, \texttt{hint} = (pp, ek, w')) \leftarrow \texttt{KeyGen}(PP)$, then we always have $w = w'$. Thus, the first check in the extraction algorithm can always be passed given the correct hint, i.e., we always have $\texttt{Extract}(PP, \texttt{C}, \texttt{hint}) = \mathsf{SK\text{-}WPRF}.\texttt{Extract}(pp, ek, \texttt{C})$ for any circuit $\texttt{C}$. Then by the extraction correctness of $\mathsf{SK\text{-}WPRF}$, it is easy to prove the extraction correctness of HWF.

Watermarking Meaningfulness. For any fixed circuit $\texttt{C} : \{0,1\}^n \to \{0,1\}^m$ and for any fixed hint $\texttt{hint} = (pp, ek, w')$, let $w \xleftarrow{\$} \{0,1\}^\lambda$. Then the probability that $w = w'$ is $1/2^\lambda$. Thus, the probability that a fixed pair of circuit and hint can pass the extraction algorithm (with random public parameter w) is negligible.

Pseudorandomness. The pseudorandomness of HWF comes from the pseudorandomness of $\mathsf{SK\text{-}WPRF}$ by a direct reduction.

Unremovability. Finally, we prove the unremovability of HWF. This comes from the unremovability of $\mathsf{SK\text{-}WPRF}$ by a direct reduction. More precisely, assuming there exists an adversary $\mathcal{A}$ that breaks the Q-bounded ϵ-unremovability of HWF, then we can construct an adversary $\mathcal{B}$ that breaks the Q-bounded ϵ-unremovability of $\mathsf{SK\text{-}WPRF}$ as follows.[21]

On receiving the public parameter pp and the extraction key ek from its challenger, the adversary $\mathcal{B}$ first samples $w \xleftarrow{\$} \{0,1\}^\lambda$. Then it sets $PP = w$ and $\texttt{hint}^* = (pp, ek, w)$, and sends $(PP, \texttt{hint}^*)$ to the adversary $\mathcal{A}$. Next, it answers

[21] The reduction proceeds identically for the $negl(\lambda)$-unremovability case.

$\mathcal{A}$'s challenge oracle queries for at most Q times. In particular, each time on receiving $\mathcal{A}$'s challenge oracle query msg, $\mathcal{B}$ submits a query msg to its own challenge oracle and on receiving the response $\mathtt{C}^*$ from its challenge oracle, it returns $\mathtt{C}^*$ to $\mathcal{A}$. Here, we use $\mathcal{Q}^*$ to denote the set of all messages submitted by $\mathcal{A}/\mathcal{B}$ and use $\mathcal{R}^*$ to denote the set of all circuits returned to $\mathcal{A}/\mathcal{B}$. Finally, after $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$, $\mathcal{B}$ also submits $\tilde{\mathtt{C}}$ to its challenger.

First, it is easy to see, the view of $\mathcal{A}$ in the environment simulated by $\mathcal{B}$ is identical to its view in the real experiment $\mathtt{ExptUR}$ defined in Definition 5.2. Thus, the submitted circuit $\tilde{\mathtt{C}}$ satisfies

$$\exists\, \mathtt{C}^* \in \mathcal{R}^*, |\{x \in \{0,1\}^n : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq \epsilon \cdot 2^n$$

and with a non-negligible probability,

$$\mathtt{Extract}(PP, \tilde{\mathtt{C}}, \mathtt{hint}^*) = \mathsf{SK\text{-}WPRF}.\mathtt{Extract}(pp, ek, \tilde{\mathtt{C}}) \notin \mathcal{Q}^*$$

Therefore, the circuit $\tilde{\mathtt{C}}$ submitted by $\mathcal{B}$ is approximate to the watermarked circuits returned by its challenge oracle and with a non-negligible probability, it is embedded with a different message, i.e., $\mathcal{B}$ will succeed in breaking the Q-bounded ϵ-unremovability of $\mathsf{SK\text{-}WPRF}$. This completes the proof of unremovability.

E.4 Security Analysis of Robust Unobfuscatable PRFs from One Way Functions

We present proof of Theorem 6.1 in this section. More precisely, we will prove the correctness, $negl(\lambda)$-robust learnability, and black-box pseudorandomness of UOF.

Correctness. For any message msg, let $K = (\alpha, \beta, k_{enc}, k_{IR}, k_{mask}, k_{pad}, msg)$ be the output of the key generation algorithm. Also, let $ct_1, \ldots, ct_{|\mathtt{P}|}, x_1, y_1, \gamma$, $\tilde{x}_1, \tilde{y}_1, x_3, y_3, \tilde{x}_3, \tilde{y}_3$ be the variables used in $\mathtt{Extract}(\mathtt{Eval}(K, \cdot))$.

First, $ct_1, \ldots, ct_\lambda$ are encryption of bits of α since they are contained in the output of the evaluation algorithm on input an input with prefix 000. Then, $ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|}$ will be encryption of bits of $\mathtt{P}(\alpha)$ as the repetitions in the extraction algorithm actually evaluates the circuit $\mathtt{P}$ homomorphically with input encrypted in $ct_1, \ldots, ct_\lambda$. Next, let $x_1 = (u_0, u_1, u_2, u_3, u_4, u_5)$, then we have

$$y_1 = \mathtt{Eval}(K, x_1) = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, u_1 \| u_2 \| u_3 \| u_4 \| u_5)$$

and

$$\tilde{y}_1 = \mathtt{Eval}(K, \tilde{x}_1) = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, u_1 \| u_2 \| u_3 \| u_4 \oplus \alpha \oplus \alpha \| u_5) \oplus (\beta \| 0^{m-\lambda})$$

Thus, the output of $\mathtt{P}(\alpha)$ will be $\beta \oplus \gamma$, i.e., $ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|}$ will be encryption of bits of $\beta \oplus \gamma$.

Now, let $x_3 = (u_0, u_1, u_2, u_3, u_4, u_5)$, then we have

$$y_3 = \mathtt{Eval}(K, x_3) = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, u_1 \| u_2 \| u_3 \| u_4 \oplus \beta \oplus \gamma \oplus \beta \| u_5) \oplus msg$$

In addition, we have

$$\tilde{y}_3 = \texttt{Eval}(K, \tilde{x}_3) = \mathsf{F}_{mask}.\texttt{Eval}(k_{mask}, u_1 \| u_2 \| u_3 \| u_4 \oplus \gamma \| u_5)$$

Therefore, $\tilde{y}_3 \oplus y_3 = msg$ and thus the extraction algorithm will always output the correct message on input the evaluation algorithm $\texttt{Eval}(K, \cdot)$.

Robust Learnability. Next, we prove the $negl(\lambda)$-robust learnability of UOF. First, we define the following games between a challenger and a PPT $negl(\lambda)$-admissible adversary $\mathcal{A}$:

- **Game 0.** This is the real experiment defined in Definition 6.3. More precisely, the challenger proceeds as follows:
  - In the beginning, the adversary submits a message msg^* to the challenger and the challenger samples $K = (\alpha, \beta, k_{enc}, k_{IR}, k_{mask}, k_{pad}, msg^*)$ by using the key generation algorithm of UOF.
  - The challenger returns K to $\mathcal{A}$, and the adversary submits a circuit $\texttt{C}$ that agrees with $\texttt{Eval}(K, \cdot)$ on all but negligible fraction of inputs.
  - Finally, the challenger runs the extraction algorithm as follows, where $\texttt{P}$,$|\texttt{P}|$ and j_L, j_R for $j \in [\lambda + 1, |\texttt{P}|]$ are defined as in the construction:
    1. $x'_0 \xleftarrow{\$} \{0,1\}^{n-n_0}$.
    2. $x_0 = 000 \| x'_0$.
    3. $y_0 = \texttt{C}(x_0)$.
    4. $(ct_1, \ldots, ct_\lambda) = y_0[1 : \lambda \cdot (\lambda + 1)]$.
    5. $x'_1 \xleftarrow{\$} \{0,1\}^{n-n_0}$.
    6. $\gamma \xleftarrow{\$} \{0,1\}^{\lambda}$.
    7. $x_1 = 011 \| x'_1$.
    8. $y_1 = \texttt{C}(x_1)$.
    9. For $j \in [\lambda + 1, |\texttt{P}|]$:
       (a) $u_1^{(j)} \xleftarrow{\$} \{0,1\}^{n_1}$.
       (b) $(u_3^{(j)}, u_4^{(j)}, u_5^{(j)}) \xleftarrow{\$} \{0,1\}^{n_3} \times \{0,1\}^{n_4} \times \{0,1\}^{n_5}$.
       (c) If $j_L \neq j_R$:
           i. $u_2^{(j)} = (ct_{j_L}, ct_{j_R})$.
       (d) If $j_L = j_R$:
           i. $(\bar{u}_1^{(j)}, \bar{u}_2^{(j)}) \xleftarrow{\$} \{0,1\}^{n_1} \times \{0,1\}^{n_2}$.
           ii. $(\bar{u}_4^{(j)}, \bar{u}_5^{(j)}) \xleftarrow{\$} \{0,1\}^{n_4} \times \{0,1\}^{n_5}$.
           iii. $\bar{u}_3^{(j)} = ct_{j_L}$.
           iv. $\bar{u}_0^{(j)} = 010$.
           v. $\bar{x}_{2,j} = \bar{u}_0^{(j)} \| \bar{u}_1^{(j)} \| \bar{u}_2^{(j)} \| \bar{u}_3^{(j)} \| \bar{u}_4^{(j)} \| \bar{u}_5^{(j)}$.
           vi. $\bar{y}_{2,j} = \texttt{C}(\bar{x}_{2,j})$.
           vii. $\bar{ct}_{j_L} = \bar{y}_{2,j}[1 : \lambda + 1]$.
           viii. $u_2^{(j)} = (ct_{j_L}, \bar{ct}_{j_L})$.
       (e) $u_0^{(j)} = 001$.
       (f) $x_{2,j} = u_0^{(j)} \| u_1^{(j)} \| u_2^{(j)} \| u_3^{(j)} \| u_4^{(j)} \| u_5^{(j)}$.

(g) $y_{2,j} = \mathtt{C}(x_{2,j})$.
(h) $ct_j = y_{2,j}[1 : \lambda + 1]$.

10. $(u_1, u_2, u_3, u_4) \xleftarrow{\$} \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} \times \{0,1\}^{n_4}$.
11. $u_5 = (ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|})$.
12. $u_0 = 101$.
13. $x_3 = u_0 \| u_1 \| u_2 \| u_3 \| u_4 \| u_5$.
14. $y_3 = \mathtt{C}(x_3)$.
15. $\tilde{u}_0 = 011$.
16. $\tilde{u}_4 = u_4 \oplus \gamma$.
17. $\tilde{x}_3 = \tilde{u}_0 \| u_1 \| u_2 \| u_3 \| \tilde{u}_4 \| u_5$.
18. $\tilde{y}_3 = \mathtt{C}(\tilde{x}_3)$.
19. $msg = \tilde{y}_3 \oplus y_3$.

The challenger **outputs 1**, which indicates that the adversary wins, iff $msg \neq msg^*$.

- **Game 1.** This is identical to Game 0 except that the challenger computes y_0 as $y_0 = \mathtt{Eval}(K, x_0)$.
- **Game 2.** This is identical to Game 1 except that the challenger changes the way to compute $ct_1, \ldots, ct_\lambda$. In particular, for $j \in [1, \lambda]$, it samples $r_j \xleftarrow{\$} \{0,1\}^\lambda$ and computes $ct_j = r_j \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_j) \oplus \alpha[j]$.
- **Game 3.** This is identical to Game 2 except that the challenger changes the way to compute $ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|}$. In particular, for $j \in [1, \lambda]$, it samples $r_j \xleftarrow{\$} \{0,1\}^\lambda$ and computes $ct_{|\mathtt{P}|-\lambda+j} = r_j \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_j) \oplus \mathtt{P}(\alpha)[j]$. More precisely, the challenger proceeds as follows when extracting the message in the last step:

1. $(u'_1, u'_2, u'_3, u'_4, u'_5) \xleftarrow{\$} \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} \times \{0,1\}^{n_4} \times \{0,1\}^{n_5}$.
2. $u'_0 = 011$.
3. $\gamma \xleftarrow{\$} \{0,1\}^\lambda$.
4. $x_1 = u'_0 \| u'_1 \| u'_2 \| u'_3 \| u'_4 \| u'_5$.
5. $y_1 = \mathtt{C}(x_1)$.
6. $\tilde{u}'_0 = 100$.
7. $\tilde{u}'_4 = u'_4 \oplus \alpha$.
8. $\tilde{x}_1 = \tilde{u}'_0 \| u'_1 \| u'_2 \| u'_3 \| \tilde{u}'_4 \| u'_5$.
9. $\tilde{y}_1 = \mathtt{C}(\tilde{x}_1)$.
10. $b = (\tilde{y}_1 \oplus y_1)[1 : \lambda]$.
11. For $j \in [1, \lambda]$:
    (a) $r_j \xleftarrow{\$} \{0,1\}^\lambda$.
    (b) $ct_{|\mathtt{P}|-\lambda+j} = r_j \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_j) \oplus (b \oplus \gamma)[j]$.
12. $(u_1, u_2, u_3, u_4) \xleftarrow{\$} \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} \times \{0,1\}^{n_4}$.
13. $u_5 = (ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|})$.
14. $u_0 = 101$.
15. $x_3 = u_0 \| u_1 \| u_2 \| u_3 \| u_4 \| u_5$.
16. $y_3 = \mathtt{C}(x_3)$.
17. $\tilde{u}_0 = 011$.
18. $\tilde{u}_4 = u_4 \oplus \gamma$.

19. $\tilde{x}_3 = \tilde{u}_0 \| u_1 \| u_2 \| u_3 \| \tilde{u}_4 \| u_5$.
20. $\tilde{y}_3 = \mathtt{C}(\tilde{x}_3)$.
21. $msg = \tilde{y}_3 \oplus y_3$.

- **Game 4.** This is identical to Game 3 except that the challenger computes y_1 as $y_1 = \mathtt{Eval}(K, x_1)$.
- **Game 5.** This is identical to Game 4 except that the challenger computes $\tilde{y}_1$ as $\tilde{y}_1 = \mathtt{Eval}(K, \tilde{x}_1)$.
- **Game 6.** This is identical to Game 5 except that the challenger changes the way to compute $ct_{|\mathtt{P}|-\lambda+1}, \ldots, ct_{|\mathtt{P}|}$. In particular, for $j \in [1, \lambda]$, it samples $r_j \xleftarrow{\$} \{0,1\}^\lambda$ and computes $ct_{|\mathtt{P}|-\lambda+j} = r_j \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_j) \oplus (\beta \oplus \gamma)[j]$.
- **Game 7.** This is identical to Game 6 except that the challenger computes y_3 as $y_3 = \mathtt{Eval}(K, x_3)$.
- **Game 8.** This is identical to Game 7 except that the challenger computes $\tilde{y}_3$ as $\tilde{y}_3 = \mathtt{Eval}(K, \tilde{x}_3)$.

Let $\mathcal{D} = \{x \in \{0,1\}^n : \mathtt{C}(x) \neq \mathtt{Eval}(K, x)\}$ be the set of inputs that $\mathtt{C}$ evaluates differently from the PRF. As the adversary is only allowed to change evaluation of the PRF on a negligible fraction of inputs, we have

$$\frac{|\mathcal{D}|}{2^n} \leq negl(\lambda)$$

Now, let $\mathcal{E}_i$ be the output of Game i and we next show that $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$, which implies that the adversary wins in Game 0 (i.e., the real experiment defined in Definition 6.3) with a negligible probability, via proving the following lemmas.

Lemma E.11. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. Game 0 and Game 1 are identical unless $x_0 \in \mathcal{D}$. Since x_0 is sampled uniformly from a subset of $\{0,1\}^n$, which contains 2^{n-3} elements, we have

$$\Pr[x_0 \in \mathcal{D}] \leq \frac{|\mathcal{D}|}{2^{n-3}} \leq 8 \cdot \frac{|\mathcal{D}|}{2^n} \leq negl(\lambda)$$

□

Lemma E.12. $|\Pr[\mathcal{E}_1 = 1] - \Pr[\mathcal{E}_2 = 1]| = 0$.

Proof. Let $x_0 = (u_0, u_1, u_2, u_3, u_4, u_5)$ and for $j \in [1, \lambda]$, let $ct_j = r_j \| c_j$. In Game 1, r_j is derived from output of $\mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, u_0 \| u_2 \| u_3 \| u_4 \| u_5, u_1)$ and $c_j = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_j) \oplus \alpha[j]$; while in Game 2, r_j is sampled uniformly from $\{0,1\}^\lambda$ and c_j is computed in the same way. Since in Game 1, u_1 is sampled uniformly from $\{0,1\}^{n_1}$, the invoker randomization property of F_{IR} ensures that the output of $\mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, u_0 \| u_2 \| u_3 \| u_4 \| u_5, u_1)$ will be uniform even if the other part of x_0 and the key k_{IR} is fixed and known. Thus, the distributions of $(r_1, \ldots, r_\lambda)$ are identical in both games. Therefore, the probabilities that the two games output 1 are identical. □

Lemma E.13. $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| \leq negl(\lambda)$.

Proof. We first define the following hybrids for $h \in [\lambda+1, |\mathtt{P}|+1]$, where we use p_h to denote the value of the h-th wire when computing $\mathtt{P}(\alpha)$.

- $\mathsf{H}_{h,0}$: Game $\mathsf{H}_{h,0}$ is identical to Game 2 except that it changes the way to compute ct_j for $j \in [\lambda+1, h)$. In particular, for $j \in [\lambda+1, h-1]$, it samples $r_j \xleftarrow{\$} \{0,1\}^\lambda$ and computes $ct_j = r_j \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_j) \oplus p_j$.
- $\mathsf{H}_{h,1}$: Game $\mathsf{H}_{h,1}$ is identical to Game $\mathsf{H}_{h,0}$ except that it computes $\bar{y}_{2,h}$ as $\bar{y}_{2,h} = \mathtt{Eval}(K, \bar{x}_{2,h})$ (if needed).
- $\mathsf{H}_{h,2}$: Game $\mathsf{H}_{h,2}$ is identical to Game $\mathsf{H}_{h,1}$ except that it changes the way to compute $\bar{ct}_{h_L}$ (if needed). In particular, it samples $\bar{r}_h \xleftarrow{\$} \{0,1\}^\lambda$ and computes $\bar{ct}_{h_L} = \bar{r}_h \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_h) \oplus p_{h_L}$, where h_L is defined as in the construction.
- $\mathsf{H}_{h,3}$: Game $\mathsf{H}_{h,3}$ is identical to Game $\mathsf{H}_{h,2}$ except that it computes $y_{2,h}$ as $y_{2,h} = \mathtt{Eval}(K, x_{2,h})$.
- $\mathsf{H}_{h,4}$: Game $\mathsf{H}_{h,4}$ is identical to Game $\mathsf{H}_{h,3}$ except that it changes the way to compute ct_h. In particular, it samples $r_h \xleftarrow{\$} \{0,1\}^\lambda$ and computes $ct_h = r_h \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_h) \oplus p_h$.

It is easy to see Game $\mathsf{H}_{\lambda+1,0}$ is identical to Game 2 and Game $\mathsf{H}_{|\mathtt{P}|+1,0}$ is identical to Game 3. Also, for $h \in [\lambda+1, |\mathtt{P}|]$, Game $\mathsf{H}_{h,4}$ is identical to $\mathsf{H}_{h+1,0}$. Thus, it is sufficient to argue indistinguishability between $\mathsf{H}_{h,0}$ and $\mathsf{H}_{h,4}$ for $h \in [\lambda+1, |\mathtt{P}|]$. Let $\mathcal{E}_{h,i}$ be the output of Game $\mathsf{H}_{h,i}$ for $h \in [\lambda+1, |\mathtt{P}|]$. Next, we prove this via the following claims.

Claim E.3. *For $h \in [\lambda+1, |\mathtt{P}|]$, $|\Pr[\mathcal{E}_{h,0} = 1] - \Pr[\mathcal{E}_{h,1} = 1]| \leq negl(\lambda)$.*

Proof. Game $\mathsf{H}_{h,0}$ and Game $\mathsf{H}_{h,1}$ are identical unless $\bar{x}_{2,h} \in \mathcal{D}$. Let $\bar{x}_{2,h} = (\bar{u}_0^{(h)}, \bar{u}_1^{(h)}, \bar{u}_2^{(h)}, \bar{u}_3^{(h)}, \bar{u}_4^{(h)}, \bar{u}_5^{(h)})$. As $\bar{u}_1^{(h)}, \bar{u}_2^{(h)}, \bar{u}_4^{(h)}, \bar{u}_5^{(h)}$ are uniform strings, and $\bar{u}_3^{(h)} = ct_{h_L}$, whose first λ bits are uniform (note that $h_L < h$), the string $\bar{x}_{2,h}$ is sampled uniformly from a subset of $\{0,1\}^n$ with 2^{n-4} elements. Thus, the probability that $\bar{x}_{2,h}$ is in $\mathcal{D}$, which only contains a negligible fraction of inputs in $\{0,1\}^n$, is negligible. □

Claim E.4. *For $h \in [\lambda+1, |\mathtt{P}|]$, $|\Pr[\mathcal{E}_{h,1} = 1] - \Pr[\mathcal{E}_{h,2} = 1]| = 0$.*

Proof. Let $\bar{x}_{2,h} = (\bar{u}_0^{(h)}, \bar{u}_1^{(h)}, \bar{u}_2^{(h)}, \bar{u}_3^{(h)}, \bar{u}_4^{(h)}, \bar{u}_5^{(h)})$ and $\bar{ct}_{h_L} = \bar{r}_h \| \bar{c}_h$. In Game $\mathsf{H}_{h,1}$, $\bar{r}_h$ is derived from output of $\mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, \bar{u}_0^{(h)} \| \bar{u}_2^{(h)} \| \bar{u}_3^{(h)} \| \bar{u}_4^{(h)} \| \bar{u}_5^{(h)}, \bar{u}_1^{(h)})$ and $\bar{c}_h = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_h) \oplus \mu$, where μ is the decryption of ct_{h_L} and is p_{h_L}. In Game $\mathsf{H}_{h,2}$, $\bar{r}_h$ is sampled uniformly from $\{0,1\}^\lambda$ and $\bar{c}_h = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_h) \oplus p_{h_L}$. The two distributions are identical due to the invoker randomization property of F_{IR} and the fact that $\bar{u}_1^{(h)}$ is sampled uniformly from $\{0,1\}^{n_1}$. □

Claim E.5. *For $h \in [\lambda+1, |\mathtt{P}|]$, $|\Pr[\mathcal{E}_{h,2} = 1] - \Pr[\mathcal{E}_{h,3} = 1]| \leq negl(\lambda)$.*

Proof. Game $\mathsf{H}_{h,2}$ and Game $\mathsf{H}_{h,3}$ are identical unless $x_{2,h} \in \mathcal{D}$. Let $x_{2,h} = (u_0^{(h)}, u_1^{(h)}, u_2^{(h)}, u_3^{(h)}, u_4^{(h)}, u_5^{(h)})$. Then, $u_1^{(h)}, u_3^{(h)}, u_4^{(h)}, u_5^{(h)}$ are uniform strings. Also, $u_2^{(h)}$ is either $ct_{h_L} \| ct_{h_R}$ or $ct_{h_L} \| \bar{ct}_{h_L}$, where in both cases, the first λ bits and the $(\lambda+2)$-bit to the $(2\lambda+1)$-bit of $u_2^{(h)}$ are sampled uniformly at random. Thus, the string $x_{2,h}$ is sampled uniformly from a subset of $\{0,1\}^n$ with 2^{n-5} elements and therefore, the probability that $x_{2,h}$ is in $\mathcal{D}$ is negligible. □

Claim E.6. *For $h \in [\lambda+1, |\mathtt{P}|]$, $|\Pr[\mathcal{E}_{h,3}=1] - \Pr[\mathcal{E}_{h,4}=1]| = 0$.*

Proof. Let $x_{2,h} = (u_0^{(h)}, u_1^{(h)}, u_2^{(h)}, u_3^{(h)}, u_4^{(h)}, u_5^{(h)})$ and $ct_h = r_h \| c_h$. In Game $\mathsf{H}_{h,3}$, r_h is derived from output of $\mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, u_0^{(h)} \| u_2^{(h)} \| u_3^{(h)} \| u_4^{(h)} \| u_5^{(h)}, u_1^{(h)})$ and $c_h = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_h) \oplus \mu$. Here, $\mu = \mu_1 \bar{\wedge} \mu_2$ and μ_1, μ_2 are the decryption of ct_{h_L} and ct_{h_R} ($\bar{ct}_{h_L}$ if $h_L = h_R$), respectively. Thus, we have $\mu_1 = p_{h_L}$ and $\mu_2 = p_{h_R}$ and therefore $\mu = p_{h_L} \bar{\wedge} p_{h_R} = p_h$. In Game $\mathsf{H}_{h,4}$, r_h is sampled uniformly from $\{0,1\}^\lambda$ and $c_h = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_h) \oplus p_h$. The two distributions are identical due to the invoker randomization property of F_{IR} and the fact that $u_1^{(h)}$ is sampled uniformly from $\{0,1\}^{n_1}$. □

□

Lemma E.14. *$|\Pr[\mathcal{E}_3=1] - \Pr[\mathcal{E}_4=1]| \leq negl(\lambda)$.*

Proof. Game 3 and Game 4 are identical unless $x_1 \in \mathcal{D}$. Since x_1 is sampled uniformly from a subset of $\{0,1\}^n$, which contains 2^{n-3} elements, the probability that x_1 is in $\mathcal{D}$ is negligible. □

Lemma E.15. *$|\Pr[\mathcal{E}_4=1] - \Pr[\mathcal{E}_5=1]| \leq negl(\lambda)$.*

Proof. Game 4 and Game 5 are identical unless $\tilde{x}_1 \in \mathcal{D}$. Since $\tilde{x}_1$ is equal to $100 \| u_1' \| u_2' \| u_3' \| u_4' \oplus \alpha \| u_5'$, where $u_1', u_2', u_3', u_4', u_5'$ are random strings, the probability that $\tilde{x}_1$ is in $\mathcal{D}$ is also negligible. □

Lemma E.16. *$|\Pr[\mathcal{E}_5=1] - \Pr[\mathcal{E}_6=1]| = 0$.*

Proof. Game 5 and Game 6 are identical if $b = \beta$. Since

$$y_1 = \mathtt{Eval}(K, x_1) = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, u_1' \| u_2' \| u_3' \| u_4' \| u_5')$$

and

$$\tilde{y}_1 = \mathtt{Eval}(K, \tilde{x}_1) = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, u_1' \| u_2' \| u_3' \| u_4' \oplus \alpha \oplus \alpha \| u_5) \oplus (\beta \| 0^{m-\lambda})$$

we have $b = (\tilde{y}_1 \oplus y_1)[1:\lambda] = \beta$. □

Lemma E.17. *$|\Pr[\mathcal{E}_6=1] - \Pr[\mathcal{E}_7=1]| \leq negl(\lambda)$.*

Proof. Game 6 and Game 7 are identical unless $x_3 \in \mathcal{D}$. First, $x_3 = (u_0, u_1, u_2, u_3, u_4, u_5)$, where $u_0 = 101$, u_1, u_2, u_3, u_4 are random strings, and $u_5 = (ct_{|\mathsf{P}|-\lambda+1}, \ldots, ct_{|\mathsf{P}|})$. Note that for $j \in [1, \lambda]$, $ct_{|\mathsf{P}|-\lambda+j} = r_j \| c_j$, where r_j is sampled uniformly at random from $\{0,1\}^\lambda$ and $c_j = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_j) \oplus \beta[j] \oplus \gamma[j]$, which is also a random bit since γ is sampled uniformly at random from $\{0,1\}^\lambda$. Thus, u_5 is also a random string. Therefore, the probability that x_3 is in $\mathcal{D}$ is negligible. □

Lemma E.18. $|\Pr[\mathcal{E}_7 = 1] - \Pr[\mathcal{E}_8 = 1]| \leq negl(\lambda)$.

Proof. Game 7 and Game 8 are identical unless $\tilde{x}_3 \in \mathcal{D}$. First, $\tilde{x}_3 = (\tilde{u}_0, u_1, u_2, u_3, \tilde{u}_4, u_5)$, where $\tilde{u}_0 = 011$, u_1, u_2, u_3 are random strings, $\tilde{u}_4 = u_4 \oplus \gamma$, $u_4 \xleftarrow{\$} \{0,1\}^\lambda$, and $u_5 = (ct_{|\mathsf{P}|-\lambda+1}, \ldots, ct_{|\mathsf{P}|})$. Similarly, we have u_5 uniform as γ is sampled uniformly from $\{0,1\}^\lambda$. Also note that although $\tilde{u}_4 = u_4 \oplus \gamma$, it reveals no information about γ since u_4 is a uniform string in $\{0,1\}^\lambda$. Thus, the string $u_1 \| u_2 \| u_3 \| \tilde{u}_4 \| u_5$ is still distributed uniformly in $\{0,1\}^{n-n_0}$. Therefore, the probability that $\tilde{x}_3$ is in $\mathcal{D}$ is also negligible. □

Lemma E.19. $\Pr[\mathcal{E}_8 = 1] = 0$.

Proof. In Game 8, we have

$$y_3 = \mathtt{Eval}(K, x_3) = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, u_1 \| u_2 \| u_3 \| u_4 \oplus \beta \oplus \gamma \oplus \beta \| u_5) \oplus msg^*$$

In addition, we have

$$\tilde{y}_3 = \mathtt{Eval}(K, \tilde{x}_3) = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, u_1 \| u_2 \| u_3 \| u_4 \oplus \gamma \| u_5)$$

This implies that $\tilde{y}_3 \oplus y_3 = msg^*$. Therefore, in Game 8, the challenger always gets msg^* after extraction and thus will never output 1. □

Combining Lemma E.11 to Lemma E.19, we have $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$, i.e., the probability that $\mathcal{A}$ wins in the real experiment for robust learnability is negligible. This completes the proof of $negl(\lambda)$-robust learnability.

Black-Box Pseudorandomness. Next, we prove the black-box pseudorandomness of UOF. First, we define the following games between a challenger and a PPT adversary $\mathcal{A}$:

- **Game 0.** In Game 0, the challenger answers $\mathcal{A}$'s oracle queries with $\mathcal{O}_0$.
  More precisely, in the beginning, the adversary submits a message msg to the challenger and the challenger samples $K = (\alpha, \beta, k_{enc}, k_{IR}, k_{mask}, k_{pad}, msg)$ by running the key generation algorithm $\mathtt{KeyGen}(1^\lambda, msg)$.
  Then, each time the adversary submits an input $x \in \{0,1\}^n$, the challenger first parses $x = (u_0, u_1, u_2, u_3, u_4, u_5) \in \{0,1\}^{n_0} \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} \times \{0,1\}^{n_4} \times \{0,1\}^{n_5}$. Then it answers $\mathcal{A}$'s query as follows:
  - If $u_0 = 0$:
    1. $w = (u_0, u_2, u_3, u_4, u_5)$.
    2. $(r_1, r_2, \ldots, r_{\lambda+2}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, w, u_1)$.

3. For $i \in [1, \lambda]$:
   (a) $ct_i = r_i \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_i) \oplus \alpha[i]$.
4. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)[1 : m - \lambda \cdot (\lambda + 1)]$.
5. **Return** $ct_1 \| \ldots \| ct_\lambda \| y_{pad}$.

– If $u_0 = 1$:
1. Parse $u_2 = (\bar{r}_1, \bar{c}_1, \bar{r}_2, \bar{c}_2) \in \{0,1\}^\lambda \times \{0,1\} \times \{0,1\}^\lambda \times \{0,1\}$.
2. $\mu_1 = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_1) \oplus \bar{c}_1$.
3. $\mu_2 = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_2) \oplus \bar{c}_2$.
4. $\mu = \mu_1 \barwedge \mu_2$.
5. $w = (u_0, u_2, u_3, u_4, u_5)$.
6. $(r_1, r_2, \ldots, r_{\lambda+2}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, w, u_1)$.
7. $ct = r_{\lambda+1} \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_{\lambda+1}) \oplus \mu$.
8. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)[1 : m - (\lambda + 1)]$.
9. **Return** $ct \| y_{pad}$.

– If $u_0 = 2$:
1. Parse $u_3 = (\bar{r}, \bar{c}) \in \{0,1\}^\lambda \times \{0,1\}$.
2. $\mu = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}) \oplus \bar{c}$.
3. $w = (u_0, u_2, u_3, u_4, u_5)$.
4. $(r_1, r_2, \ldots, r_{\lambda+2}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, w, u_1)$.
5. $ct = r_{\lambda+2} \| \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, r_{\lambda+2}) \oplus \mu$.
6. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)[1 : m - (\lambda + 1)]$.
7. **Return** $ct \| y_{pad}$.

– If $u_0 = 3$:
1. $z = (u_1, u_2, u_3, u_4, u_5)$.
2. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
3. **Return** y_{mask}.

– If $u_0 = 4$:
1. $u'_4 = u_4 \oplus \alpha$.
2. $z = (u_1, u_2, u_3, u'_4, u_5)$.
3. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
4. **Return** $(\beta \| 0^{m-\lambda}) \oplus y_{mask}$.

– If $u_0 = 5$:
1. Parse $u_5 = (\bar{r}_i, \bar{c}_i)_{i \in [1,\lambda]} \in (\{0,1\}^\lambda \times \{0,1\})^\lambda$.
2. For $i \in [1, \lambda]$:
   (a) $\mu_i = \mathsf{F}_{enc}.\mathtt{Eval}(k_{enc}, \bar{r}_i) \oplus \bar{c}_i$.
3. $\nu = \mu_1 \| \ldots \| \mu_\lambda$.
4. $u'_4 = u_4 \oplus \nu \oplus \beta$.
5. $z = (u_1, u_2, u_3, u'_4, u_5)$.
6. $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
7. **Return** $msg \oplus y_{mask}$.

– If $u_0 = 6$ or $u_0 = 7$:
1. $y_{pad} = \mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)$.
2. **Return** y_{pad}.

Finally, the game outputs 1 iff the adversary outputs 1.
Here, w.l.o.g., we assume that all inputs submitted by the adversary are *distinct*.

- **Game 1.** This is identical to Game 0 except that for each query, in case $u_0 \in \{0,1,2,6,7\}$, the challenger samples y_{pad} uniformly at random instead of evaluating it as (parts of) the output of $\mathsf{F}_{pad}.\mathtt{Eval}(k_{pad}, x)$.
- **Game 2.** This is identical to Game 1 except that for each query, in case $u_0 \in \{0,1,2\}$, the challenger samples $(r_1, \ldots, r_{\lambda+2}) \xleftarrow{\$} (\{0,1\}^\lambda)^{\lambda+2}$ instead of evaluating it as $(r_1, r_2, \ldots, r_{\lambda+2}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, u_0 \| u_2 \| u_3 \| u_4 \| u_5, u_1)$.
- **Game 3.** This is identical to Game 2 except that for each query, in case $u_0 \in \{0,1,2,5\}$, the challenger uses a random function f instead of the pseudorandom function F_{enc}.
- **Game 4.** This is identical to Game 3 except that for each query, in case $u_0 \in \{3,4,5\}$, the challenger uses a random function g instead of the pseudorandom function F_{mask}.
- **Game 5.** This is identical to Game 4 except that the challenger maintains a list $\mathcal{L}$. Here, $\mathcal{L}$ is initialized as an empty list in the beginning. Then on receiving a query $x = (u_0, u_1, u_2, u_3, u_4, u_5)$, the challenger puts (x, z) to the list if $u_0 \in \{3,4,5\}$, where z is the input to g. Recall that, z is computed as follows in each case:
  - If $u_0 = 3$:
    1. $z = (u_1, u_2, u_3, u_4, u_5)$.
  - If $u_0 = 4$:
    1. $u'_4 = u_4 \oplus \alpha$.
    2. $z = (u_1, u_2, u_3, u'_4, u_5)$.
  - If $u_0 = 5$:
    1. Parse $u_5 = (\bar{r}_i, \bar{c}_i)_{i \in [1,\lambda]} \in (\{0,1\}^\lambda \times \{0,1\})^\lambda$.
    2. For $i \in [1, \lambda]$:
       (a) $\mu_i = f(\bar{r}_i) \oplus \bar{c}_i$.
    3. $\nu = \mu_1 \| \ldots \| \mu_\lambda$.
    4. $u'_4 = u_4 \oplus \nu \oplus \beta$.
    5. $z = (u_1, u_2, u_3, u'_4, u_5)$.
- **Game 6.** This is identical to Game 5 except that for each query, in case $u_0 \in \{3,4,5\}$, the challenger returns a random string $y \in \{0,1\}^m$. Note that in Game 6, the challenger does not need to maintain the list $\mathcal{L}$ and $\mathcal{L}$ is only used in arguing the indistinguishability between Game 5 and Game 6.
- **Game 7.** This is identical to Game 6 except that for each query, in case $u_0 \in \{0,1,2\}$, the challenger returns a random string $y \in \{0,1\}^m$.

It is easy to see that in Game 7, the challenger answers the adversary's queries with $\mathcal{O}_1$. Let $\mathcal{E}_i$ be the output of Game i, then it is sufficient to show that $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_7 = 1]| \leq negl(\lambda)$. Next, we argue this via proving the following lemmas.

Lemma E.20. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. First, by pseudorandomness of F_{pad}, we can use a random function F to replace F_{pad} in answering $\mathcal{A}$'s queries. Next, as for each query, the input to F is x, which will be distinct (recall that we assume w.l.o.g. that all inputs submitted to the oracle are distinct), it is safe to replace output of F with a fresh random string. □

Lemma E.21. $|\Pr[\mathcal{E}_1 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$.

Proof. First, by pseudorandomness of F_{IR}, we can use a random function F to replace F_{IR} in answering $\mathcal{A}$'s queries. Next, for each query, the input to F, which is $u_0\|u_2\|u_3\|u_4\|u_5\|u_1$, will be distinct since the inputs x are distinct and the map from x to $u_0\|u_2\|u_3\|u_4\|u_5\|u_1$ is bijective. Thus, it is safe to replace output of F with a fresh random string. □

Lemma E.22. $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 2 and Game 3 comes from pseudorandomness of F_{enc} directly. □

Lemma E.23. $|\Pr[\mathcal{E}_3 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 3 and Game 4 comes from pseudorandomness of F_{mask} directly. □

Lemma E.24. $|\Pr[\mathcal{E}_4 = 1] - \Pr[\mathcal{E}_5 = 1]| = 0$.

Proof. The adversary's view in Game 4 and Game 5 are identical, thus the probabilities that the two games output 1 are also identical. □

Lemma E.25. $|\Pr[\mathcal{E}_5 = 1] - \Pr[\mathcal{E}_6 = 1]| \leq negl(\lambda)$.

Proof. In Game 5, the challenger will return $g(z) \oplus S$ to the adversary if $u_0 \in \{3, 4, 5\}$, where g is a random function and $S \in \{0^m, \beta\|0^{m-\lambda}, msg\}$; and in Game 6, the challenger will return a random string in $\{0,1\}^m$ if $u_0 \in \{3,4,5\}$. Note that Game 5 and Game 6 are identical if all z are distinct.

To prove that z are not likely to repeat, we first define

$$\mathcal{L}_\sigma = \{(x = (u_0, u_1, u_2, u_3, u_4, u_5), z) \in \mathcal{L} : u_0 = \sigma\}$$

for $\sigma \in \{3,4,5\}$. Then we define the following events in Game 5:

- col_3: There exists (x,z) and (x',z') in $\mathcal{L}_3$ s.t. $x \neq x'$ and $z = z'$.
- col_4: There exists (x,z) and (x',z') in $\mathcal{L}_4$ s.t. $x \neq x'$ and $z = z'$.
- col_5: There exists (x,z) and (x',z') in $\mathcal{L}_5$ s.t. $x \neq x'$ and $z = z'$.
- col_{34}: There exists $(x,z) \in \mathcal{L}_3$ and $(x',z') \in \mathcal{L}_4$ s.t. $x \neq x'$ and $z = z'$.
- col_{35}: There exists $(x,z) \in \mathcal{L}_3$ and $(x',z') \in \mathcal{L}_5$ s.t. $x \neq x'$ and $z = z'$.
- col_{45}: There exists $(x,z) \in \mathcal{L}_4$ and $(x',z') \in \mathcal{L}_5$ s.t. $x \neq x'$ and $z = z'$.

It is easy to see if none of the 6 events occurs, then all z are distinct. Thus, it is sufficient to show that the probability that at least one of the 6 event occurs is negligible.

Claim E.7. $\Pr[col_3] = 0$

Proof. Assuming col_3 occurs, i.e., there exists (x,z) and (x',z') in $\mathcal{L}_3$ s.t. $x \neq x'$ and $z = z'$. Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ and $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$. Then we have $u_0 = u'_0 = 3$, $z = (u_1, u_2, u_3, u_4, u_5)$ and $z' = (u'_1, u'_2, u'_3, u'_4, u'_5)$. Since $z = z'$, we have $u_i = u'_i$ for $i \in [1,5]$. This contradicts the assumption that $x \neq x'$, thus col_3 will not occur. □

Claim E.8. $\Pr[\mathit{col}_4] = 0$

Proof. Assuming col_4 occurs, i.e., there exists (x, z) and (x', z') in $\mathcal{L}_4$ s.t. $x \neq x'$ and $z = z'$. Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ and $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$. Then we have $u_0 = u'_0 = 4$, $z = (u_1, u_2, u_3, u_4 \oplus \alpha, u_5)$ and $z' = (u'_1, u'_2, u'_3, u'_4 \oplus \alpha, u'_5)$. Since $z = z'$, we have $u_i = u'_i$ for $i \in [1, 5]$. This contradicts the assumption that $x \neq x'$, thus col_4 will not occur. □

Claim E.9. $\Pr[\mathit{col}_5] = 0$

Proof. Assuming col_5 occurs, i.e., there exists (x, z) and (x', z') in $\mathcal{L}_5$ s.t. $x \neq x'$ and $z = z'$. Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ and $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$. Then we have $u_0 = u'_0 = 5$, $z = (u_1, u_2, u_3, u_4 \oplus \nu \oplus \beta, u_5)$ and $z' = (u'_1, u'_2, u'_3, u'_4 \oplus \nu' \oplus \beta, u'_5)$, where ν, ν' are decryption of u_5 and u'_5 respectively. Since $z = z'$, we have $u_i = u'_i$ for $i \in [1, 3]$ and $u_5 = u'_5$, which implies that $\nu = \nu'$. Thus, we also have $u_4 = u'_4$. This contradicts the assumption that $x \neq x'$, thus col_5 will not occur. □

Before arguing that the remaining 3 events also occur with a negligible probability, we first define "*mask query*" as a query $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ satisfying $u_0 \in \{3, 4, 5\}$ and define "*enc query*" as a query $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ satisfying $u_0 \in \{0, 1, 2\}$. Then we define a set of auxiliary adversaries $\mathcal{B}_\ell$, which is identical to $\mathcal{A}$ except that it will artificially abort the game (and outputs 1) after submitting ℓ mask queries.[22] We also define $\mathit{col}_{\sigma,\ell}$ for $\sigma \in \{3, 4, 5, 34, 35, 45\}$ to be the event that col_σ occurs when the challenger interacts with $\mathcal{B}_\ell$ (instead of $\mathcal{A}$) in Game 5. Assume that the PPT adversary $\mathcal{A}$ makes at most Q queries to the challenger, where Q is polynomial in λ, then $\mathcal{B}_{Q+1}$ is identical to $\mathcal{A}$. Thus, it is sufficient to prove that $\mathit{col}_{\sigma,Q+1}$ occurs with negligible probability for $\sigma \in \{34, 35, 45\}$.

To prove this, we first prove the following claim.

Claim E.10. *For $\ell \in [1, Q]$, if*

$$\Pr[\mathit{col}_{3,\ell} \vee \mathit{col}_{4,\ell} \vee \mathit{col}_{5,\ell} \vee \mathit{col}_{34,\ell} \vee \mathit{col}_{35,\ell} \vee \mathit{col}_{45,\ell}] \leq \mathit{negl}(\lambda)$$

then we have

$$\Pr[\mathit{col}_{35,\ell+1} \vee \mathit{col}_{45,\ell+1} \vee \mathit{col}_{34,\ell+1}] \leq \mathit{negl}(\lambda)$$

Proof. We first define the following auxiliary games:

- H_0: The challenger interacts with $\mathcal{B}_{\ell+1}$ and proceeds identically as it does in Game 5, except that after $\mathcal{B}_{\ell+1}$ stops or after it submits the $(\ell+1)$-th mask query and aborts (i.e., the challenger does not need to respond the $(\ell+1)$-th mask query from $\mathcal{B}_{\ell+1}$), the challenger **outputs 1** iff *at least one* of the three events $\mathit{col}_{35,\ell+1}$, $\mathit{col}_{45,\ell+1}$ and $\mathit{col}_{34,\ell+1}$ occurs.

[22] If $\mathcal{A}$ stops the game with an output before triggering $\mathcal{B}_\ell$'s abort condition, then $\mathcal{B}_\ell$ will also stop and outputs what $\mathcal{A}$ outputs.

- H_1: This is identical to Game H_0 except that the challenger returns random strings in the first ℓ mask queries.
- H_2: This is identical to Game H_1 except that the challenger does not check if $\mathit{col}_{35,\ell+1}$ or $\mathit{col}_{45,\ell+1}$ occurs. In particular, after $\mathcal{B}_{\ell+1}$ stops or aborts, the challenger outputs 1 iff $\mathit{col}_{34,\ell+1}$ occurs.
  Note that in Game H_2, the challenger does not need to compute and store z for an input $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ if $u_0 = 5$.
- H_3: This is identical to Game H_2 except that the challenger returns random strings for enc queries.

Let $\mathcal{F}_i$ be the output of Game H_i. It is easy to see $\Pr[\mathit{col}_{35,\ell+1} \vee \mathit{col}_{45,\ell+1} \vee \mathit{col}_{34,\ell+1}] = \Pr[\mathcal{F}_0 = 1]$. Thus, it is sufficient to prove that $\Pr[\mathcal{F}_0 = 1] \leq negl(\lambda)$. This can be guaranteed by the following claims:

Claim E.10.1. $|\Pr[\mathcal{F}_0 = 1] - \Pr[\mathcal{F}_1 = 1]| \leq negl(\lambda)$.

Proof. First, Game H_0 and Game H_1 are identical if all z appeared in the first ℓ mask queries are distinct. This will occur with all but negligible probability since $\Pr[\mathit{col}_{3,i} \vee \mathit{col}_{4,i} \vee \mathit{col}_{5,i} \vee \mathit{col}_{34,i} \vee \mathit{col}_{35,i} \vee \mathit{col}_{45,i}] \leq negl(\lambda)$. □

Claim E.10.2. $|\Pr[\mathcal{F}_1 = 1] - \Pr[\mathcal{F}_2 = 1]| \leq negl(\lambda)$.

Proof. Game H_1 and Game H_2 are identical unless $\mathit{col}_{35,\ell+1}$ or $\mathit{col}_{45,\ell+1}$ occurs.

In Game H_1, the challenger will return random strings for the first ℓ mask queries and answers to other oracles queries can be generated without using β. Thus, it does not need β when answering oracle queries and can postpone the selection of β *after* the adversary stops or aborts.[23] Therefore, all queries submitted by the adversary will be independent of β.

For any $(x, z) \in \mathcal{L}_5$ and $(x', z') \in \mathcal{L}_3$. Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ and $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$. Then we have $z = (u_1, u_2, u_3, u_4 \oplus \nu \oplus \beta, u_5)$ and $z' = (u'_1, u'_2, u'_3, u'_4, u'_5)$, where ν is determined by u_5. If $z = z'$, then we have $u_4 \oplus \nu \oplus \beta = u'_4$, which occurs with a negligible probability since β is sampled uniformly at random from $\{0,1\}^\lambda$ after $\mathcal{B}_{\ell+1}$ submits the queries.

For any $(x, z) \in \mathcal{L}_5$ and $(x', z') \in \mathcal{L}_4$. Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ and $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$. Then we have $z = (u_1, u_2, u_3, u_4 \oplus \nu \oplus \beta, u_5)$ and $z' = (u'_1, u'_2, u'_3, u'_4 \oplus \alpha, u'_5)$, where ν is determined by u_5. If $z = z'$, then we have $u_4 \oplus \nu \oplus \beta = u'_4 \oplus \alpha$, which occurs with a negligible probability since β is sampled uniformly at random from $\{0,1\}^\lambda$ after $\mathcal{B}_{\ell+1}$ submits the queries.

This completes proof of this Claim. □

Claim E.10.3. $|\Pr[\mathcal{F}_2 = 1] - \Pr[\mathcal{F}_3 = 1]| \leq negl(\lambda)$.

Proof. We first define the following hybrids for $h \in [0, Q]$

- $\mathsf{H}_{2,h}$: In Game $\mathsf{H}_{2,h}$, for the first h enc queries, the challenger answers them as in Game H_2, and for the remaining enc queries, it answers them as in Game H_3.

[23] It can also postpone the computation of z after that.

Note that $\mathsf{H}_{2,Q} = \mathsf{H}_2$ and $\mathsf{H}_{2,0} = \mathsf{H}_3$. Thus, it is sufficient to show indistinguishability between $\mathsf{H}_{2,h}$ and $\mathsf{H}_{2,h-1}$ for $h \in [1, Q]$.

Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ be the h-th enc query. We consider the following cases:

- **If $u_0 = 0$.** In this case, the challenger will output $r_1 \| c_1 \| \dots \| r_\lambda \| c_\lambda \| y_{pad}$. In both Game $\mathsf{H}_{2,h}$ and Game $\mathsf{H}_{2,h-1}$, $y_{pad} \xleftarrow{\$} \{0,1\}^{m-\lambda\cdot(\lambda+1)}$, and $r_i \xleftarrow{\$} \{0,1\}^\lambda$ for $i \in [1, \lambda]$. In Game $\mathsf{H}_{2,h}$, $c_i = f(r_i) \oplus \alpha[i]$ for $i \in [1, \lambda]$; while in Game $\mathsf{H}_{2,h-1}$, $c_i \xleftarrow{\$} \{0,1\}$ for $i \in [1, \lambda]$.
  Since for $i \in [1, \lambda]$, r_i is sampled uniformly from $\{0,1\}^\lambda$, the probability that the challenger has evaluated f on r_i in previous oracle queries and the probability that there exists distinct $i, j \in [1, \lambda]$ s.t. $r_i = r_j$ are negligible. In addition, all oracle queries after the h-th enc query are responded with a random string (the $(\ell+1)$-th mask query is not responded), thus, the challenger will also not evaluate $f(r_i)$ after the h-th enc query. Besides, the challenger does not need to evaluate z for an input $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$ if $u'_0 = 5$, thus, it will not evaluate $f(r_i)$ when determining if it should output 1. Therefore, it is safe to replace $f(r_i)$ (and thus c_i) with a random bit.
- **If $u_0 = 1$ or $u_0 = 2$.** In this case, the challenger will output $r \| c \| y_{pad}$. In both Game $\mathsf{H}_{2,h}$ and Game $\mathsf{H}_{2,h-1}$, $y_{pad} \xleftarrow{\$} \{0,1\}^{m-(\lambda+1)}$, and $r \xleftarrow{\$} \{0,1\}^\lambda$. In Game $\mathsf{H}_{2,h}$, $c = f(r) \oplus \mu$, where μ is either determined by u_2 (if $u_0 = 1$) or determined by u_3 (if $u_0 = 2$); while in Game $\mathsf{H}_{2,h-1}$, $c \xleftarrow{\$} \{0,1\}$.
  Since r is sampled uniformly from $\{0,1\}^\lambda$, the probability that the challenger has evaluated f on r in previous oracle queries is negligible. In addition, all oracle queries after the h-th enc query are responded with a random string (the $(\ell+1)$-th mask query is not responded), thus, the challenger will also not evaluate $f(r)$ after the h-th enc query. Besides, the challenger does not need to evaluate z for an input $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$ if $u'_0 = 5$, thus, it will not evaluate $f(r)$ when determining if it should output 1. Therefore, it is safe to replace $f(r)$ (and thus c) with a random bit.

As for the h-th enc query $x = (u_0, u_1, u_2, u_3, u_4, u_5)$, we have either $u_0 = 0$ or $u_0 \in \{1, 2\}$, and in both cases, the adversary's views are identical in the two games with all but negligible probability, indistinguishability between Game $\mathsf{H}_{2,h}$ and Game $\mathsf{H}_{2,h-1}$ follows.

Therefore, Game H_2 and Game H_3 are indistinguishable. □

Claim E.10.4. $\Pr[\mathcal{F}_3 = 1] \leq negl(\lambda)$.

Proof. In Game H_3, the challenger will return random strings for all oracle queries. Thus, it does not need α when answering oracle queries and can postpone the selection of α *after* the adversary aborts or stops.[24] Therefore, all queries submitted by the adversary will be independent of α.

[24] It can also postpone the computation of z after that.

For any $(x, z) \in \mathcal{L}_4$ and $(x', z') \in \mathcal{L}_3$. Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ and $x' = (u'_0, u'_1, u'_2, u'_3, u'_4, u'_5)$. Then we have $z = (u_1, u_2, u_3, u_4 \oplus \alpha, u_5)$ and $z' = (u'_1, u'_2, u'_3, u'_4, u'_5)$. If $z = z'$, then we have $u_4 \oplus \alpha = u'_4$, which occurs with a negligible probability since α is sampled uniformly at random from $\{0,1\}^\lambda$ after $\mathcal{B}_{\ell+1}$ submits the queries.

This completes proof of this Claim. □

□

We continue to show that $\Pr[col_{34,Q+1} \vee col_{35,Q+1} \vee col_{45,Q+1}] \leq negl(\lambda)$. First, from Claim E.7 to Claim E.9, we have $\Pr[col_3] = \Pr[col_4] = \Pr[col_5] = 0$, thus we have $\Pr[col_{3,i}] = \Pr[col_{4,i}] = \Pr[col_{5,i}] = 0$ for all $i \in [1, Q]$. In addition, $\Pr[col_{34,1}] = \Pr[col_{35,1}] = \Pr[col_{45,1}] = 0$ since only one mask query is made and there cannot be collision. Therefore, we have

$$\Pr[col_{3,1} \vee col_{4,1} \vee col_{5,1} \vee col_{34,1} \vee col_{35,1} \vee col_{45,1}] = 0$$

Then, by Claim E.10, we have

$$\Pr[col_{35,2} \vee col_{45,2} \vee col_{34,2}] \leq negl(\lambda)$$

Now we have

$$\Pr[col_{3,2} \vee col_{4,2} \vee col_{5,2} \vee col_{34,2} \vee col_{35,2} \vee col_{45,2}] \leq negl(\lambda)$$

and repeating this for Q times, we have

$$\Pr[col_{34,Q+1} \vee col_{35,Q+1} \vee col_{45,Q+1}] \leq negl(\lambda) \tag{6}$$

Since $\mathcal{B}_{Q+1}$ is identical to $\mathcal{A}$, Equation (6) implies that

$$\Pr[col_{34} \vee col_{35} \vee col_{45}] \leq negl(\lambda)$$

Then combing with Claim E.7 to Claim E.9, we have

$$\Pr[col_3 \vee col_4 \vee col_5 \vee col_{34} \vee col_{35} \vee col_{45}] \leq negl(\lambda)$$

That is, the inputs z to g will be distinct with all but negligible probability, and it is safe to replace outputs of g with random strings. This completes proof of Lemma E.25. □

Lemma E.26. $|\Pr[\mathcal{E}_6 = 1] - \Pr[\mathcal{E}_7 = 1]| \leq negl(\lambda)$.

Proof. Proof of Lemma E.26 is similar to proof of Claim E.10.3. For completeness, we also provide the detailed proof for Lemma E.26 below.

We also define "*enc query*" as a query $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ satisfying $u_0 \in \{0, 1, 2\}$. Assume that $\mathcal{A}$ makes at most Q queries to the challenger, where Q is polynomial in λ, then we define the following hybrids for $h \in [0, Q]$

- H_h: In Game H_h, for the first h enc queries, the challenger answers them as in Game 6, and for the remaining enc queries, it answers them as in Game 7.

Note that H_Q is identical to Game 6 and H_0 is identical to Game 7. Thus, it is sufficient to show indistinguishability between H_h and H_{h-1} for $h \in [1, Q]$.

Let $x = (u_0, u_1, u_2, u_3, u_4, u_5)$ be the h-th enc query. We consider the following cases:

- **If $u_0 = 0$.** In this case, the challenger will output $r_1 \| c_1 \| \dots \| r_\lambda \| c_\lambda \| y_{pad}$. In both Game H_h and Game H_{h-1}, $y_{pad} \xleftarrow{\$} \{0,1\}^{m-\lambda\cdot(\lambda+1)}$, and $r_i \xleftarrow{\$} \{0,1\}^\lambda$ for $i \in [1, \lambda]$. In Game H_h, $c_i = f(r_i) \oplus \alpha[i]$ for $i \in [1, \lambda]$; while in Game H_{h-1}, $c_i \xleftarrow{\$} \{0,1\}$ for $i \in [1, \lambda]$.
  Since for $i \in [1, \lambda]$, r_i is sampled uniformly from $\{0,1\}^\lambda$, the probability that the challenger has evaluated f on r_i in previous oracle queries and the probability that there exists distinct $i, j \in [1, \lambda]$ s.t. $r_i = r_j$ are negligible. In addition, all oracle queries after the h-th enc query are responded with a random string, thus, the challenger will also not evaluate $f(r_i)$ after the h-th enc query. Therefore, it is safe to replace $f(r_i)$ (and thus c_i) with a random bit.
- **If $u_0 = 1$ or $u_0 = 2$.** In this case, the challenger will output $r \| c \| y_{pad}$. In both Game H_h and Game H_{h-1}, $y_{pad} \xleftarrow{\$} \{0,1\}^{m-(\lambda+1)}$, and $r \xleftarrow{\$} \{0,1\}^\lambda$. In Game H_h, $c = f(r) \oplus \mu$, where μ is either determined by u_2 (if $u_0 = 1$) or determined by u_3 (if $u_0 = 2$); while in Game H_{h-1}, $c \xleftarrow{\$} \{0,1\}$.
  Since r is sampled uniformly from $\{0,1\}^\lambda$, the probability that the challenger has evaluated f on r in previous oracle queries is negligible. In addition, all oracle queries after the h-th enc query are responded with a random string, thus, the challenger will also not evaluate $f(r)$ after the h-th enc query. Therefore, it is safe to replace $f(r)$ (and thus c) with a random bit.

As for the h-th enc query $x = (u_0, u_1, u_2, u_3, u_4, u_5)$, we have either $u_0 = 0$ or $u_0 \in \{1, 2\}$, and in both cases, the adversary's views are identical in the two games with all but negligible probability, indistinguishability between Game H_h and Game H_{h-1} follows.

Therefore, Game 6 and Game 7 are indistinguishable. □

Combining Lemma E.20 to Lemma E.26, we have $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_7 = 1]| \leq negl(\lambda)$, i.e., the probability that $\mathcal{A}$ outputs 1 when interacting with $\mathcal{O}_0$ and the probability that it outputs 1 when interacting with $\mathcal{O}_1$ does not have a non-negligible difference. This completes the proof of black-box pseudorandomness.

E.5 Security Analysis of Special FHE

We present proof of Theorem D.1 in this section. More precisely, we will prove the perfect correctness, the ciphertext and key pseudorandomness, and the uniformity of transformed ciphertext of SFHE.

Perfect Correctness. For any public key $\boldsymbol{A} \in \mathbb{Z}_q^{(n+1)\times m}$ and for any ciphertext $\boldsymbol{C} \in \mathbb{Z}_q^{(n+1)\times N}$, we say that $\boldsymbol{C}$ is an encryption of $\mu \in \{0,1\}$ with error E if there exists $\boldsymbol{R} \in [-E, E]^{m\times N}$ s.t.

$$\boldsymbol{C} = \mu \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R} \mod q$$

We first bound size of the error for a valid ciphertext.

Lemma E.27. *For any message $\mu \in \{0,1\}$ and any $(pk, sk, ek) \leftarrow \mathtt{KeyGen}(1^\lambda)$, let $\boldsymbol{C}$ be a valid ciphertext of μ under (pk, sk, ek), then $\boldsymbol{C}$ is an encryption of μ with error Σ'.*

Proof. Let $pk = \boldsymbol{A}$, $sk = \boldsymbol{s}$ and $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$ be any tuple of keys output by the key generation algorithm. Also, let $\hat{\boldsymbol{t}}, \bar{\boldsymbol{t}}$ be the variables used in the key generation algorithm for generating (pk, sk, ek). Also, let $ek' = ((\hat{\boldsymbol{\psi}}'_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}'_i)_{i\in[1,K]})$ be any output of $\mathtt{RandEK}(pk, ek)$.

Note that a ciphertext is a valid ciphertext either if it is the output of the encryption algorithm or it is the homomorphic evaluation result over valid ciphertexts. We first bound the error size for a fresh ciphertext.

Claim E.11. *For any message $\mu \in \{0,1\}$, let $\boldsymbol{C} \leftarrow \mathtt{Enc}(\boldsymbol{A}, \mu)$, then $\boldsymbol{C}$ is an encryption of μ with error 1.*

Proof. As $\boldsymbol{C} \leftarrow \mathtt{Enc}(\boldsymbol{A}, \mu)$, there exists $\boldsymbol{R} \in \{0,1\}^{m\times N}$ s.t. $\boldsymbol{C} = \mu \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R} \mod q$. Thus, $\boldsymbol{C}$ is an encryption of μ with error $E \le 1$. □

Next, we bound the error size for ciphertext output by the evaluation algorithm. We first consider output of the auxiliary algorithm $\mathtt{DecNAND}$.

Claim E.12. *For any two ciphertexts $\boldsymbol{C}_0^*$, $\boldsymbol{C}_1^*$, where $\boldsymbol{C}_0^*$ is an encryption of μ_0 with error Σ' and $\boldsymbol{C}_1^*$ is an encryption of μ_1 with error Σ', let ek^* be either ek or ek', and let $\boldsymbol{C}^* \leftarrow \mathtt{DecNAND}(ek^*, \boldsymbol{C}_0^*, \boldsymbol{C}_1^*)$, then $\boldsymbol{C}^*$ is an encryption of $\mu = \mu_0 \barwedge \mu_1$ with error Σ'.*

Proof. Let $ek^* = ((\hat{\boldsymbol{\psi}}^*_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}^*_i)_{i\in[1,K]})$. Also, let $(\hat{\boldsymbol{v}}_0^*, \hat{w}_0^*) \leftarrow \mathtt{RedCT}(ek^*, \boldsymbol{C}_0^*)$ and $(\hat{\boldsymbol{v}}_1^*, \hat{w}_1^*) \leftarrow \mathtt{RedCT}(ek^*, \boldsymbol{C}_1^*)$ be the variables used in the algorithm $\mathtt{DecNAND}$ for generating $\boldsymbol{C}^*$. Besides, let $\mathtt{D} = \bar{\mathtt{D}}[\hat{\boldsymbol{v}}_0^*, \hat{w}_0^*, \hat{\boldsymbol{v}}_1^*, \hat{w}_1^*]$. We prove Claim E.12 via arguing the following claims.

Claim E.12.1. *For $i \in [1, K]$, $\check{\boldsymbol{\Psi}}^*_i$ is an encryption of $\bar{\boldsymbol{t}}[i]$ with error 2.*

Proof. If $ek^* = ek$, then

$$\check{\boldsymbol{\Psi}}^*_i = \bar{\boldsymbol{t}}[i] \cdot \boldsymbol{G} + \boldsymbol{A}\check{\boldsymbol{R}}_i \mod q$$

where $\check{\boldsymbol{R}}_i \in \{0,1\}^{m\times N}$. Also, if $ek^* = ek'$, then

$$\check{\boldsymbol{\Psi}}^*_i = \bar{\boldsymbol{t}}[i] \cdot \boldsymbol{G} + \boldsymbol{A}\check{\boldsymbol{R}}_i + \boldsymbol{A}\check{\boldsymbol{R}}'_i \mod q$$

where $\check{\boldsymbol{R}}_i \in \{0,1\}^{m\times N}$ and $\check{\boldsymbol{R}}'_i \in \{0,1\}^{m\times N}$. In both case, we have

$$\check{\boldsymbol{\Psi}}^*_i = \bar{\boldsymbol{t}}[i] \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R}_i \mod q$$

where $\boldsymbol{R}_i \in [0,2]^{m\times N}$, and Claim E.12.1 follows. □

Claim E.12.2. *For any positive integer E, for any two ciphertexts $\boldsymbol{C}_0$, $\boldsymbol{C}_1$, where $\boldsymbol{C}_0$ is an encryption of μ_0 with error E and $\boldsymbol{C}_1$ is an encryption of μ_1 with error E, let $\boldsymbol{C} \leftarrow \mathtt{NAND}(\boldsymbol{C}_0, \boldsymbol{C}_1)$, then $\boldsymbol{C}$ is an encryption of $\mu = \mu_1 \barwedge \mu_2$ with error $(N+1)E$.*

Proof. First, we have

$$\boldsymbol{C}_0 = \mu_0 \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R}_0 \mod q$$

$$\boldsymbol{C}_1 = \mu_1 \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R}_1 \mod q$$

where $\boldsymbol{R}_0 \in [-E,E]^{m\times N}$ and $\boldsymbol{R}_1 \in [-E,E]^{m\times N}$. Thus, we have

$$\begin{aligned}
\boldsymbol{C} &= \boldsymbol{G} - \boldsymbol{C}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1) \\
&= \boldsymbol{G} - (\mu_0 \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R}_0) \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1) \\
&= \boldsymbol{G} - (\mu_0 \cdot \boldsymbol{C}_1 + \boldsymbol{A}\boldsymbol{R}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1)) \\
&= \boldsymbol{G} - (\mu_0 \cdot (\mu_1 \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R}_1) + \boldsymbol{A}\boldsymbol{R}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1)) && \mod q \\
&= \boldsymbol{G} - (\mu_0\mu_1 \cdot \boldsymbol{G} + \mu_0 \cdot \boldsymbol{A}\boldsymbol{R}_1 + \boldsymbol{A}\boldsymbol{R}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1)) \\
&= (1-\mu_0\mu_1) \cdot \boldsymbol{G} + \boldsymbol{A} \cdot (-(\mu_0 \cdot \boldsymbol{R}_1 + \boldsymbol{R}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1))) \\
&= (1-\mu_0\mu_1) \cdot \boldsymbol{G} + \boldsymbol{A} \cdot \boldsymbol{R}
\end{aligned}$$

where $\boldsymbol{R} = -(\mu_0 \cdot \boldsymbol{R}_1 + \boldsymbol{R}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1))$. Note that as $\mu_0 \in \{0,1\}$ and $\boldsymbol{R}_1 \in [-E, E]^{m\times N}$, we have $\mu_0 \cdot \boldsymbol{R}_1 \in [-E,E]^{m\times N}$. Also, as $\boldsymbol{G}^{-1}(\boldsymbol{C}_1) \in \{0,1\}^{N\times N}$ and $\boldsymbol{R}_0 \in [-E,E]^{m\times N}$, we have $\boldsymbol{R}_0 \cdot \boldsymbol{G}^{-1}(\boldsymbol{C}_1) \in [-NE, NE]^{m\times N}$. Thus, we have $\boldsymbol{R} \in [-(N+1)E, (N+1)E]^{m\times N}$. This completes the proof. □

Claim E.12.3. *$\boldsymbol{C}^*$ is an encryption of $\mathtt{D}(\bar{\boldsymbol{t}})$ with error Σ'.*

Proof. First, by Claim E.12.1 and Claim E.12.2, $\boldsymbol{C}^*$ is an encryption of $\mathtt{D}(\bar{\boldsymbol{t}})$ with error $2(N+1)^d$, where d is the depth of $\mathtt{D}$.

Next, we bound the size of d. First, by Lemma 4.5 of [BV11], the first two steps in the circuit $\mathtt{D}$ is of depth $O(\log k)$.[25] Also, to compute step 3, we only need to check if $u_{0,\hat{l}} \vee u_{0,\hat{l}-1} = 1$, where we use $u_{0,i}$ to denote the i-th bit of the binary decomposition of u_0 (i.e. $u_0 = \sum_{i=1}^{\hat{l}} 2^{i-1} \cdot u_{0,i}$). This can be done in constant depth. Similar, we can compute step 4 in constant depth. Finally, it is easy to see step 5 can be computed in depth 1. Therefore, we have $d = O(\log k) < \omega(\log k)$ and $2(N+1)^d < \Sigma'$. □

[25] Lemma 4.5 of [BV11] shows that both steps are in $\mathtt{Arith}[O(\log k), 1]$, which is the class of arithmetic circuit over $GF(2)$ with $\{+,\times\}$ gates and depth $d = O(\log k)$. Note that the $+$ gate and the $\times$ gate over $GF(2)$ can be represented by constant number of NAND gates, thus if we compute the two steps with a circuit composed exclusively by NAND gates, the circuit depth is also in $O(\log k)$.

Claim E.12.4. *For any positive integer E, let $ek^\dagger = ((\hat{\boldsymbol{\psi}}^\dagger_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}^\dagger_i)_{i\in[1,K]})$ satisfying $\forall i \in [1,n], \iota \in [1,l], \exists \hat{\boldsymbol{a}}^\dagger_{i,\iota} \in \mathbb{Z}_p^k, \hat{e}^\dagger_{i,\iota} \in [-E,E]$:*

$$\hat{\boldsymbol{\psi}}^\dagger_{i,\iota} = \begin{pmatrix} \hat{\boldsymbol{a}}^\dagger_{i,\iota} \\ \hat{\boldsymbol{a}}^\dagger_{i,\iota} \cdot \hat{\boldsymbol{t}} + \hat{e}^\dagger_{i,\iota} + \lfloor \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \rceil \end{pmatrix} \mod p$$

Also, let $\boldsymbol{C}$ be an encryption of μ with error Σ'. Let $(\hat{\boldsymbol{v}}, \hat{w}) \leftarrow \mathtt{RedCT}(ek^\dagger, \widehat{\boldsymbol{C}})$. Then there exists $\hat{e} \in [-nl(E+1), nl(E+1)]$ s.t.

$$\hat{w} = \hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} + \hat{e} + \mu \cdot \frac{p+1}{2} \mod p$$

Proof. Let $\boldsymbol{C}'$ be the last l columns of $\boldsymbol{C}$ and let $\boldsymbol{c} = \boldsymbol{C}' \cdot \boldsymbol{h}$. Let $w = \boldsymbol{c}[n+1]$, $\boldsymbol{v} = \boldsymbol{c}[1:n]$. Let $v_{i,\iota}$ be the ι-th bit of $\boldsymbol{v}[i]$. Then we have

$$\hat{\boldsymbol{v}} = -\sum_{i=1}^{n}\sum_{\iota=1}^{l} v_{i,\iota} \cdot \hat{\boldsymbol{a}}_{i,\iota} \mod p$$

Also, we have

$$\begin{aligned}
&\hat{w} \\
=&\lfloor \frac{p}{q} w \rceil - \sum_{i=1}^{n}\sum_{\iota=1}^{l} v_{i,\iota} \cdot (\hat{\boldsymbol{a}}^\dagger_{i,\iota} \cdot \hat{\boldsymbol{t}} + \hat{e}^\dagger_{i,\iota} + \lfloor \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \rceil) \\
=&\lfloor \frac{p}{q} w \rceil - \sum_{i=1}^{n}\sum_{\iota=1}^{l} v_{i,\iota} \cdot \hat{\boldsymbol{a}}^\dagger_{i,\iota} \cdot \hat{\boldsymbol{t}} - \sum_{i=1}^{n}\sum_{\iota=1}^{l} v_{i,\iota} \cdot (\hat{e}^\dagger_{i,\iota} + \lfloor \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \rceil) \\
=&\hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} + \lfloor \frac{p}{q} w \rceil - \sum_{i=1}^{n}\sum_{\iota=1}^{l} v_{i,\iota} \cdot (\hat{e}^\dagger_{i,\iota} + \lfloor \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \rceil) && \mod p \\
=&\hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} + \frac{p}{q} w - E_1 - \sum_{i=1}^{n}\sum_{\iota=1}^{l} v_{i,\iota} \cdot \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \\
=&\hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} + \frac{p}{q} w - E_1 - \sum_{i=1}^{n} \frac{p}{q} \cdot \boldsymbol{v}[i] \cdot \boldsymbol{s}[i] \\
=&\hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} - E_1 + \frac{p}{q} \cdot (w - \boldsymbol{s} \cdot \boldsymbol{v})
\end{aligned}$$

where

$$E_1 = \frac{p}{q} w - \lfloor \frac{p}{q} w \rceil + \sum_{i=1}^{n}\sum_{\iota=1}^{l} v_{i,\iota} \cdot (\hat{e}^\dagger_{i,\iota} + \lfloor \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \rceil - \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i])$$

Since $-\frac{1}{2} \le \frac{p}{q} w - \lfloor \frac{p}{q} w \rceil \le \frac{1}{2}$, $\hat{e}^\dagger_{i,\iota} \in [-E, E]$, $-\frac{1}{2} \le \lfloor \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \rceil - \frac{p}{q} \cdot 2^{\iota-1} \cdot \boldsymbol{s}[i] \le \frac{1}{2}$, and $v_{i,\iota} \in \{0,1\}$,

$$|E_1| \le nl(E + \frac{1}{2}) + \frac{1}{2}$$

Also, as $\boldsymbol{C}$ is an encryption of μ with error Σ', there exists $\boldsymbol{R} \in [-\Sigma', \Sigma']^{m \times l}$ s.t.

$$\boldsymbol{C}' = \mu \cdot \boldsymbol{G}' + \boldsymbol{A}\boldsymbol{R} \mod q$$

where $\boldsymbol{G}'$ is the last l column of $\boldsymbol{G}$. This implies that

$$\boldsymbol{c} = \begin{pmatrix} \boldsymbol{0} \\ \mu \cdot \frac{q+1}{2} \end{pmatrix} + \boldsymbol{A}\boldsymbol{r} \mod q$$

where $\boldsymbol{r} \in [-l \cdot \Sigma', l \cdot \Sigma']^m$. Thus, we have

$$\boldsymbol{v} = \boldsymbol{B} \cdot \boldsymbol{r} \mod q$$

and

$$w = \mu \cdot \frac{q+1}{2} + \boldsymbol{s}^\intercal \cdot \boldsymbol{B} \cdot \boldsymbol{r} + \boldsymbol{e}^\intercal \cdot \boldsymbol{r} \mod q$$

where $\boldsymbol{e} \leftarrow \tilde{\mathcal{D}}_\sigma^m$. Therefore

$$w - \boldsymbol{s} \cdot \boldsymbol{v} = \mu \cdot \frac{q+1}{2} + \boldsymbol{e}^\intercal \cdot \boldsymbol{r} \mod q$$

That is,

$$\begin{aligned} & \hat{w} \\ =& \hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} - E_1 + \frac{p}{q} \cdot (\mu \cdot \frac{q+1}{2} + \boldsymbol{e}^\intercal \cdot \boldsymbol{r}) && \mod p \\ =& \hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} - E_2 + \mu \cdot \frac{p+1}{2} \end{aligned}$$

where

$$E_2 = E_1 + \frac{q-p}{2q} \cdot \mu - \frac{p}{q} \cdot \boldsymbol{e}^\intercal \cdot \boldsymbol{r}$$

Note that $\boldsymbol{e} \in [-\lambda \cdot \sigma, \lambda \cdot \sigma]^m$ (by definition of the truncated discrete Gaussian distribution), thus we have

$$|\boldsymbol{e}^\intercal \cdot \boldsymbol{r}| \leq m \cdot \lambda \cdot \sigma \cdot l \cdot \Sigma' < \Sigma \ll \frac{q}{p}$$

i.e., $|\frac{p}{q} \cdot \boldsymbol{e}^\intercal \cdot \boldsymbol{r}| < 1$. Also, $|\frac{q-p}{2q} \cdot \mu| < \frac{1}{2}$. Therefore, we have

$$|E_2| \leq nl(E + \frac{1}{2}) + \frac{1}{2} + \frac{1}{2} + 1 \leq nl(E + \frac{1}{2}) + 2 < nl(E+1)$$

Finally, as $\hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}}$, $\hat{w}$, μ, $\frac{p+1}{2}$ are all integers and $E_2 = \hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} - \hat{w} + \mu \cdot \frac{p+1}{2} \mod p$, E_2 is also an integer.

To summarize, there exists $\hat{e} \in [-nl(E+1), nl(E+1)]$ satisfying $\hat{w} = \hat{\boldsymbol{v}} \cdot \hat{\boldsymbol{t}} + \hat{e} + \mu \cdot \frac{p+1}{2}$ and this completes the proof. □

Claim E.12.5. *For $i \in [1,n], \iota \in [1,l]$, there exists $\hat{\boldsymbol{v}}^*_{i,\iota} \in \mathbb{Z}_p^k$ and $\hat{e}^*_{i,\iota} \in [-(nl+1)(\lambda\cdot\hat{\sigma}+1), (nl+1)(\lambda\cdot\hat{\sigma}+1)]$ s.t.*

$$\hat{\boldsymbol{\psi}}^*_{i,\iota} = \begin{pmatrix} \hat{\boldsymbol{v}}^*_{i,\iota} \\ \hat{\boldsymbol{v}}^*_{i,\iota}\cdot\hat{\boldsymbol{t}} + \hat{e}^*_{i,\iota} + \lfloor \frac{p}{q}\cdot 2^{\iota-1}\cdot \boldsymbol{s}[i] \rceil \end{pmatrix} \mod p$$

Proof. If $ek^* = ek$, then

$$\hat{\boldsymbol{\psi}}^*_{i,\iota} = \begin{pmatrix} \hat{\boldsymbol{a}}_{i,\iota} \\ \hat{\boldsymbol{a}}_{i,\iota}\cdot\hat{\boldsymbol{t}} + \hat{e}_{i,\iota} + \lfloor \frac{p}{q}\cdot 2^{\iota-1}\cdot \boldsymbol{s}[i] \rceil \end{pmatrix} \mod p$$

where $\hat{\boldsymbol{a}}_{i,\iota} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\hat{e}_{i,\iota} \leftarrow \tilde{\mathcal{D}}_{\hat{\sigma}}$. Note that by definition of the truncated discrete Gaussian distribution, $|\hat{e}_{i,\iota}| \le \lambda\cdot\hat{\sigma}$.

Also, if $ek^* = ek'$, then

$$\hat{\boldsymbol{\psi}}^*_{i,\iota} = \begin{pmatrix} \hat{\boldsymbol{a}}_{i,\iota} \\ \hat{\boldsymbol{a}}_{i,\iota}\cdot\hat{\boldsymbol{t}} + \hat{e}_{i,\iota} + \lfloor \frac{p}{q}\cdot 2^{\iota-1}\cdot \boldsymbol{s}[i] \rceil \end{pmatrix} + \begin{pmatrix} \hat{\boldsymbol{v}}_{i,\iota} \\ \hat{w}_{i,\iota} \end{pmatrix} \mod p$$

where $(\hat{\boldsymbol{v}}_{i,\iota}, \hat{w}_{i,\iota}) \leftarrow \mathtt{RedCT}(ek, \widehat{\boldsymbol{C}}'_{i,\iota})$, $\widehat{\boldsymbol{C}}'_{i,\iota} = \boldsymbol{A}\widehat{\boldsymbol{R}}'_{i,\iota} \mod q$, and $\widehat{\boldsymbol{R}}'_{i,\iota} \in \{0,1\}^{m\times N}$. It is easy to see $\widehat{\boldsymbol{C}}'_{i,\iota}$ is an encryption of 0 with error 1. Then by Claim E.12.4, there exists $E_{i,\iota} \in [-nl(\lambda\cdot\hat{\sigma}+1), nl(\lambda\cdot\hat{\sigma}+1)]$ s.t.

$$\hat{w}_{i,\iota} = \hat{\boldsymbol{v}}_{i,\iota}\cdot\hat{\boldsymbol{t}} + E_{i,\iota}$$

Thus,

$$\hat{\boldsymbol{\psi}}^*_{i,\iota} = \begin{pmatrix} \hat{\boldsymbol{a}}_{i,\iota} + \hat{\boldsymbol{v}}_{i,\iota} \\ (\hat{\boldsymbol{a}}_{i,\iota} + \hat{\boldsymbol{v}}_{i,\iota})\cdot\hat{\boldsymbol{t}} + \hat{E}_{i,\iota} + \lfloor \frac{p}{q}\cdot 2^{\iota-1}\cdot \boldsymbol{s}[i] \rceil \end{pmatrix} \mod p$$

where $\hat{E}_{i,\iota} = E_{i,\iota} + \hat{e}_{i,\iota}$ and we have

$$|\hat{E}_{i,\iota}| \le nl(\lambda\cdot\hat{\sigma}+1) + \lambda\cdot\hat{\sigma} < (nl+1)(\lambda\cdot\hat{\sigma}+1)$$

This completes proof of Claim E.12.5. □

Claim E.12.6. $\mathtt{D}(\bar{\boldsymbol{t}}) = \mu_0 \barwedge \mu_1$.

Proof. Recall that $\boldsymbol{C}^*_0$ is an encryption of μ_0 with error Σ'. Then by Claim E.12.5 and Claim E.12.4, there exists $\hat{e}_0 \in [-\widehat{\Sigma}, \widehat{\Sigma}]$ (recall that $\widehat{\Sigma} = nl((nl+1)(\lambda\cdot\hat{\sigma}+1)+1)$) s.t.

$$\hat{w}^*_0 = \hat{\boldsymbol{v}}^*_0\cdot\hat{\boldsymbol{t}} + \hat{e}_0 + \mu_0\cdot\frac{p+1}{2} \mod p$$

Thus, we have

$$\begin{aligned} u_0^* &= \widehat{\Sigma} + \hat{w}_0^* - \sum_{i=1}^{k} \sum_{\iota=1}^{\hat{l}} 2^{\iota-1} \cdot \hat{\boldsymbol{v}}_0^*[i] \cdot \overline{\boldsymbol{t}}[(i-1)\hat{l} + \iota] \\ &= \widehat{\Sigma} + \hat{w}_0^* - \sum_{i=1}^{k} \hat{\boldsymbol{v}}_0^*[i] \cdot \hat{\boldsymbol{t}}[i] && \bmod p \\ &= \widehat{\Sigma} + \hat{e}_0 + \mu_0 \cdot \frac{p+1}{2} \end{aligned}$$

Note that $0 \leq \widehat{\Sigma} + \hat{e}_0 \leq 2\widehat{\Sigma}$. Thus, if $\mu_0 = 0$,

$$0 \leq u_0^* \leq 2\widehat{\Sigma} < \frac{p+1}{4} \leq 2^{\hat{l}-2}$$

Also, if $\mu_0 = 1$,

$$2^{\hat{l}-2} < \frac{p+1}{2} \leq u_0^* \leq \frac{p+1}{2} + 2\widehat{\Sigma} < \frac{3(p+1)}{4} < p$$

Thus, the circuit $\mathtt{D}$ can recover the correct μ_0. Similarly, we can prove that the circuit $\mathtt{D}$ can recover the correct μ_1. Therefore, we have $\mathtt{D}(\overline{\boldsymbol{t}}) = \mu_0 \mathbin{\overline{\wedge}} \mu_1$. □

Combining Claim E.12.3 and Claim E.12.6, Claim E.12 follows.

□

Since the evaluation algorithm mainly uses the algorithm $\mathtt{DecNAND}$ to perform homomorphic operations over the ciphertext, Claim E.12 implies that if the input ciphertexts to the evaluation algorithm are encryption of a vector $\boldsymbol{x} \in \{0,1\}^{\ell_{in}}$ with error Σ', then the output ciphertexts are also encryption of the correct output with error Σ'. Formally, we have:

Corollary E.1. *For any ℓ_{in}, ℓ_{out} that are polynomial in λ, any circuit $\mathtt{C} : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell_{out}}$, any $\boldsymbol{x} \in \{0,1\}^{\ell_{in}}$, and for any vector of ciphertexts $\boldsymbol{ct}_x$ s.t. for $i \in [1, \ell_{in}]$, $\boldsymbol{ct}_x[i]$ is an encryption of $\boldsymbol{x}[i]$ with error Σ', let ek^* be either ek or ek', and let $\boldsymbol{ct}_y \leftarrow \mathtt{Eval}(ek^*, \boldsymbol{ct}_x, \mathtt{C})$. Then for $i \in [1, \ell_{out}]$, $\boldsymbol{ct}_y[i]$ is an encryption of $\mathtt{C}(\boldsymbol{x})[i]$ with error Σ'.*

Now, we denote a ciphertext output by the encryption algorithm as a level 0 valid ciphertext. Also, for any positive integer Q, we say that a ciphertext is a level Q valid ciphertext if it is outputted by the evaluation algorithm, where the input ciphertexts are valid ciphertexts of levels not exceeding $Q-1$ and at least one of them is a level $Q-1$ valid ciphertext.[26] We prove Lemma E.27 via proving that for any non-negative integer Q, a level Q valid ciphertext of μ is an encryption of μ with error Σ'.

[26] Looking ahead, we only need to deal with level 2 valid ciphertext in the security analysis of our FHE-based robust unobfuscatable PRF.

First, by Claim E.11, a level 0 valid ciphertext of μ is an encryption of μ with error Σ'. Now, for any positive integer Q, assume that for all $0 \leq P < Q$, any level P valid ciphertext of μ is an encryption of μ with error Σ', then by Corollary E.1, any level Q valid ciphertext of μ is also an encryption of μ with error Σ'. This completes the proof. □

We then show that a ciphertext with a small error can be decrypted correctly by the decryption algorithms.

Lemma E.28. *For any message $\mu \in \{0,1\}$, for any $(pk, sk, ek) \leftarrow \mathtt{KeyGen}(1^\lambda)$, and for any $rpk \leftarrow \mathtt{RandPK}(pk)$, let $\boldsymbol{C}$ be an encryption of μ with error Σ', and let $\boldsymbol{c} \leftarrow \mathtt{TranCT}(rpk, \boldsymbol{C})$, then we have*

$$\mathtt{Dec}(sk, \boldsymbol{C}) = \mu$$

and

$$\mathtt{TCTDec}(sk, \boldsymbol{c}) = \mu$$

Proof. Let $\boldsymbol{C}'$ be the last l columns of $\boldsymbol{C}$ and let $\boldsymbol{c}' = \boldsymbol{C}' \cdot \boldsymbol{h}$. Let $w = \boldsymbol{c}'[n+1]$, $\boldsymbol{v} = \boldsymbol{c}'[1:n]$. As $\boldsymbol{C}$ is an encryption of μ with error Σ', there exists $\boldsymbol{R} \in [-\Sigma', \Sigma']^{m \times l}$ s.t.

$$\boldsymbol{C}' = \mu \cdot \boldsymbol{G}' + \boldsymbol{A}\boldsymbol{R} \mod q$$

where $\boldsymbol{G}'$ is the last l column of $\boldsymbol{G}$. This implies that

$$\boldsymbol{c}' = \begin{pmatrix} \boldsymbol{0} \\ \mu \cdot \frac{q+1}{2} \end{pmatrix} + \boldsymbol{A}\boldsymbol{r} \mod q$$

where $\boldsymbol{r} \in [-l \cdot \Sigma', l \cdot \Sigma']^m$. Thus, we have

$$\boldsymbol{v} = \boldsymbol{B} \cdot \boldsymbol{r} \mod q$$

and

$$w = \mu \cdot \frac{q+1}{2} + (\boldsymbol{s}^\intercal \cdot \boldsymbol{B} + \boldsymbol{e}^\intercal) \cdot \boldsymbol{r} \mod q$$

where $\boldsymbol{e} \leftarrow \tilde{\mathcal{D}}_\sigma^m$.

We first prove that the standard decryption algorithm can recover the correct message from $\boldsymbol{C}$. First, we have

$$u = w - \boldsymbol{s} \cdot \boldsymbol{v} = \mu \cdot \frac{q+1}{2} + \boldsymbol{e}^\intercal \cdot \boldsymbol{r} \mod q$$

Note that $\boldsymbol{e} \in [-\lambda \cdot \sigma, \lambda \cdot \sigma]^m$ (by definition of the truncated discrete Gaussian distribution), thus we have

$$|\boldsymbol{e}^\intercal \cdot \boldsymbol{r}| \leq m \cdot \lambda \cdot \sigma \cdot l \cdot \Sigma' \leq \Sigma \ll \frac{q}{4}$$

Therefore, if $\mu = 1$, we have $|u - \frac{q+1}{2}| \leq \Sigma$ and the decryption algorithm will output 1. In addition, if $\mu = 0$, we have either $u < \frac{q}{4}$ or $u > \frac{3q}{4}$, which implies that $|u - \frac{q+1}{2}| > \frac{q-2}{4} \gg \Sigma$ and thus the decryption algorithm will output 0.

Next, we prove that the decryption algorithm for transformed ciphertexts is also able to recover the correct message. Recall that $rp\mathcal{k} = \boldsymbol{A}'$ where

$$\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{R}' \mod q$$

and $\boldsymbol{R}' \in \{0,1\}^{m \times \bar{m}}$. Also,

$$\boldsymbol{c} = \begin{pmatrix} \boldsymbol{v} \\ \\ w \end{pmatrix} + \boldsymbol{A}'\boldsymbol{x} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ z \end{pmatrix} \mod q$$

where $\boldsymbol{x} \in \{0,1\}^{\bar{m}}$, $z \in [\Sigma, \frac{q+1}{2} - \Sigma]$. Thus, we have

$$\boldsymbol{c} = \begin{pmatrix} \boldsymbol{B} \cdot (\boldsymbol{r} + \boldsymbol{R}' \cdot \boldsymbol{x}) \\ \\ \mu \cdot \frac{q+1}{2} + (\boldsymbol{s}^\intercal \cdot \boldsymbol{B} + \boldsymbol{e}^\intercal) \cdot (\boldsymbol{r} + \boldsymbol{R}' \cdot \boldsymbol{x}) + z \end{pmatrix} \mod q \tag{7}$$

Let $w' = \boldsymbol{c}[n+1]$, $\boldsymbol{v}' = \boldsymbol{c}[1:n]$, then we have

$$u' = w' - \boldsymbol{s} \cdot \boldsymbol{v}' = \mu \cdot \frac{q+1}{2} + \boldsymbol{e}^\intercal \cdot (\boldsymbol{r} + \boldsymbol{R}' \cdot \boldsymbol{x}) + z \mod q$$

Since $\boldsymbol{r} \in [-l \cdot \Sigma', l \cdot \Sigma']^m$, $\boldsymbol{R}' \in \{0,1\}^{m \times \bar{m}}$ and $\boldsymbol{x} \in \{0,1\}^{\bar{m}}$, we have $\boldsymbol{r} + \boldsymbol{R}' \cdot \boldsymbol{x} \in [-(l \cdot \Sigma' + \bar{m}), l \cdot \Sigma' + \bar{m}]^m$. Therefore,

$$|\boldsymbol{e}^\intercal \cdot (\boldsymbol{r} + \boldsymbol{R}' \cdot \boldsymbol{x})| \leq m \cdot \lambda \cdot \sigma \cdot (l \cdot \Sigma' + \bar{m}) \leq \Sigma - 2 \tag{8}$$

Moreover, as $z \in [\Sigma, \frac{q+1}{2} - \Sigma]$, we have $\boldsymbol{e}^\intercal \cdot (\boldsymbol{r} + \boldsymbol{R}' \cdot \boldsymbol{x}) + z \in [2, \frac{q+1}{2} - 2]$. Thus, if $\mu = 0$, we have $u' \in [2, \frac{q+1}{2} - 2] \subset [0, \frac{q+1}{2})$, and thus the algorithm $\mathtt{TCTDec}$ will output 0. In addition, if $\mu = 1$, we have $u' \in [2 + \frac{q+1}{2}, q-1]$, and thus the algorithm $\mathtt{TCTDec}$ will output 1.

This completes proof of Lemma E.28. □

Combining Lemma E.27 and Lemma E.28, the perfect correctness of SFHE follows.

Ciphertext and key Pseudorandomness. Before proving the ciphertext and key pseudorandomness of SFHE, we first give a formal description of our assumption. Roughly speaking, we assume that the bootstrappable version of GSW scheme presented in [GSW13] (denoted as GSW in this section), which uses the dimension-modulus reduction technique proposed in [BV11], has pseudorandom ciphertext.

The Assumption. Now, we give a formal definition of our assumption.

Definition E.1. *Let $(pk, sk, ek) \leftarrow \mathtt{KeyGen}(1^\lambda)$, then for all PPT adversary $\mathcal{A}$, we have:*

$$|\Pr[\mathcal{A}^{\mathcal{O}_E^0(\cdot)}(pk, ek) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_E^1(\cdot)()}(pk, ek) = 1]| \leq negl(\lambda)$$

Here we define the oracles that can be queried by the adversary as follows:

– $\mathcal{O}_E^0$ *is an oracle that takes as input a message* $msg \in \{0,1\}$ *and returns* $\mathtt{Enc}(pk, msg)$.
– $\mathcal{O}_E^1$ *is an oracle that takes as input a message* $msg \in \{0,1\}$ *and returns a random matrix in* $\mathbb{Z}_q^{(n+1)\times N}$.

Remark E.2. The key generation algorithm of our special FHE scheme is identical to the key generation algorithm of GSW. Our encryption algorithm works slightly differently from the encryption algorithm of GSW. In more detail, the output of our encryption algorithm is $\boldsymbol{C} = \mu \cdot \boldsymbol{G} + \boldsymbol{A}\boldsymbol{R} \mod q$, and the output of the encryption algorithm of GSW is the bit decomposition of $\boldsymbol{C}$. As one can transform between these two forms of ciphertexts efficiently, circular IND-CPA security of GSW, which should be assumed if one hopes to prove that GSW is a secure *fully* homomorphic encryption scheme, implies circular IND-CPA security of our special FHE scheme.

Here, we require that the special FHE scheme has pseudorandom ciphertext instead of requiring that its ciphertexts of 0 and 1 are indistinguishable. This requirement is close to the circular IND-CPA security. In addition, the non-circular version of the requirement comes from the standard LWE assumptions as shown below in Remark E.3. Thus, it should be satisfied if GSW is secure.

Remark E.3. It is easy to see the assumption described above can be implied by the standard LWE assumption directly if we do not require circular security (i.e., $(\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]}$ in the evaluation key is removed). In particular, by the decision-$\mathrm{LWE}_{k,nl,p,\tilde{\mathcal{D}}_{\tilde{\sigma}}}$ assumption, $(\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}$ are indistinguishable from random vectors in $\mathbb{Z}_p^{k+1}$. Thus, the adversary cannot learn any information about $\boldsymbol{s}$ from the evaluation key. Then by the decision-$\mathrm{LWE}_{n,m,q,\tilde{\mathcal{D}}_\sigma}$ assumption, $\boldsymbol{A}$ is indistinguishable from a random matrix in $\mathbb{Z}_q^{(n+1)\times m}$. Finally, if $\boldsymbol{A}$ is a random matrix, then the ciphertexts are statistically indistinguishable from random matrices in $\mathbb{Z}_q^{(n+1)\times N}$ due to the leftover hash lemma, and the (non-circular version of the) above assumption follows.

The Proof. The ciphertext and key pseudorandomness defined in Appendix D.1 comes from the above assumption by a direct reduction. More precisely, assuming there exists an adversary $\mathcal{A}$ that breaks the ciphertext and key pseudorandomness of SFHE, then we can construct an adversary B that invalidates the above assumption as follows.

On receiving the pubic key pk and the evaluation key $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$, the adversary $\mathcal{B}$ sends (pk, ek) to $\mathcal{A}$. Then it answers $\mathcal{A}$'s oracle queries as follows:

- On receiving a message msg to the oracle $\mathcal{O}_{enc}$, $\mathcal{B}$ submits a query msg to its own oracle $\mathcal{O}_E$ and on receiving the response $\boldsymbol{C}$ from its oracle, it returns $\boldsymbol{C}$ to $\mathcal{A}$.
- On receiving an oracle query to $\mathcal{O}_{rpk}$, $\mathcal{B}$ submits $\lceil \bar{m}/N \rceil$ queries to its oracle $\mathcal{O}_E$, where the submitted messages are all 0. Then on receiving the responses $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_{\lceil \bar{m}/N \rceil}$, it sets $\boldsymbol{C} = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_{\lceil \bar{m}/N \rceil})$ and returns the first $\bar{m}$ columns of $\boldsymbol{C}$.

- On receiving an oracle query to $\mathcal{O}_{rek}$, $\mathcal{B}$ submits $(K + nl)$ queries to its oracle $\mathcal{O}_E$, where the submitted messages are all 0. Then on receiving the responses $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_{K+nl}$, it first computes

$$\check{\boldsymbol{\Psi}}_i' = \check{\boldsymbol{\Psi}}_i + \boldsymbol{C}_i \mod q$$

for $i \in [1, K]$. Then for $i \in [1, n]$ and $\iota \in [1, l]$, it computes

$$(\hat{\boldsymbol{v}}_{i,\iota}, \hat{w}_{i,\iota}) \leftarrow \mathtt{RedCT}(ek, \boldsymbol{C}_{K+(i-1)l+\iota})$$

and

$$\hat{\boldsymbol{\psi}}_{i,\iota}' = \hat{\boldsymbol{\psi}}_{i,\iota} + \begin{pmatrix} \hat{\boldsymbol{v}}_{i,\iota} \\ \\ \hat{w}_{i,\iota} \end{pmatrix} \mod p$$

It returns $ek' = ((\hat{\boldsymbol{\psi}}_{i,\iota}')_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i')_{i\in[1,K]})$ to $\mathcal{A}$.

Finally, after $\mathcal{A}$ outputs a bit b, $\mathcal{B}$ also outputs b.

It is easy to see, if $\mathcal{B}$'s oracle queries are answered by $\mathcal{O}_E^0$, then $\mathcal{B}$ answers $\mathcal{A}$'s oracle queries to $\mathcal{O}_{enc}$, $\mathcal{O}_{rpk}$ and $\mathcal{O}_{rek}$ with $\mathcal{O}_{enc}^0$, $\mathcal{O}_{rpk}^0$ and $\mathcal{O}_{rek}^0$ respectively.

On the other hand, if $\mathcal{B}$'s oracle queries are answered by $\mathcal{O}_E^1$, then $\mathcal{B}$ will use random matrices to answer $\mathcal{A}$'s oracle queries to $\mathcal{O}_{enc}$ and $\mathcal{O}_{rpk}$, i.e., they are answered with $\mathcal{O}_{enc}^1$ and $\mathcal{O}_{rpk}^1$ respectively. In addition, for each randomized evaluation key $ek' = ((\hat{\boldsymbol{\psi}}_{i,\iota}')_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i')_{i\in[1,K]})$ returned by $\mathcal{B}$, $\check{\boldsymbol{\Psi}}_i'$ will be a random matrix in $\mathbb{Z}_q^{(n+1)\times N}$ if the matrices $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_K$ used to generate them are random matrices in $\mathbb{Z}_q^{(n+1)\times N}$. Also, by Claim E.13 stated and argued below, $\hat{\boldsymbol{\psi}}_{i,\iota}'$ will be a random vector in $\mathbb{Z}_p^{k+1}$ if the matrices $\boldsymbol{C}_{K+1}, \ldots, \boldsymbol{C}_{K+nl}$ used to generate them are random matrices in $\mathbb{Z}_q^{(n+1)\times N}$. Thus, if $\mathcal{B}$'s oracle queries are answered by $\mathcal{O}_E^1$, then it answers $\mathcal{A}$'s oracle queries to $\mathcal{O}_{rek}$ with $\mathcal{O}_{rek}^1$.

Therefore, if $\mathcal{A}$ can distinguish the oracles $(\mathcal{O}_{enc}^0, \mathcal{O}_{rpk}^0, \mathcal{O}_{rek}^0)$ from the oracles $(\mathcal{O}_{enc}^1, \mathcal{O}_{rpk}^1, \mathcal{O}_{rek}^1)$ with a non-negligible probability, then $\mathcal{B}$ can also distinguish its oracles $\mathcal{O}_E^0$ from $\mathcal{O}_E^1$ with a non-negligible probability. This completes the reduction.

Claim E.13. *Let* $(pk, sk, ek) \leftarrow \mathtt{KeyGen}(1^\lambda)$ *and* $\boldsymbol{C} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times N}$*, then* $(\hat{\boldsymbol{v}}, \hat{w}) \leftarrow \mathtt{RedCT}(ek, \boldsymbol{C})$ *is statistically indistinguishable from a random vector in* $\mathbb{Z}_p^{k+1}$*.*

Proof. Let $ek = ((\hat{\boldsymbol{\psi}}_{i,\iota})_{i\in[1,n],\iota\in[1,l]}, (\check{\boldsymbol{\Psi}}_i)_{i\in[1,K]})$, $\boldsymbol{C}'$ be the last l columns of $\boldsymbol{C}$ and $\boldsymbol{c} = \boldsymbol{C}' \cdot \boldsymbol{h}$. Also, let $w = \boldsymbol{c}[n+1]$ and $\boldsymbol{v} = \boldsymbol{c}[1:n]$. Let $v_{i,\iota}$ be the ι-th bit of $\boldsymbol{v}[i]$. Besides, for $i \in [1, n]$, $\iota \in [1, l]$, let $\hat{\boldsymbol{a}}_{i,\iota} = \hat{\boldsymbol{\psi}}_{i,\iota}[1:k]$ and let $\hat{b}_{i,\iota} = \hat{\boldsymbol{\psi}}_{i,\iota}[k+1]$. Note that

$$\hat{w} = \lfloor \frac{p}{q} w \rceil - \sum_{i=1}^{n} \sum_{\iota=1}^{l} v_{i,\iota} \cdot \hat{b}_{i,\iota} \mod p$$

and

$$\hat{\boldsymbol{v}} = -\sum_{i=1}^{n} \sum_{\iota=1}^{l} v_{i,\iota} \cdot \hat{\boldsymbol{a}}_{i,\iota} \mod p$$

As $\boldsymbol{C}$ is distributed uniformly over $\mathbb{Z}_q^{(n+1)\times N}$ and $\boldsymbol{h}$ is a fixed non-zero vector independent of $\boldsymbol{C}$, $\boldsymbol{c}$ is a uniform vector in $\mathbb{Z}_q^{(n+1)}$. Thus, $\boldsymbol{v}$ is a random vector in $\mathbb{Z}_q^n$, which implies that the binary string $(v_{i,\iota})_{i\in[1,n],\iota\in[1,l]}$ has min-entropy of $n\log q > k\log p + \omega(\log\lambda)$. Let $\hat{\boldsymbol{A}} = (\hat{\boldsymbol{a}}_{1,1},\ldots,\hat{\boldsymbol{a}}_{n,l})$, then $\hat{\boldsymbol{A}}$ is a uniform matrix in $\mathbb{Z}_p^{k\times nl}$(recall that $\hat{\boldsymbol{a}}_{i,\iota} \xleftarrow{\$} \mathbb{Z}_p^k$ in the key genration algorithm). By the leftover hash lemma, $\hat{\boldsymbol{v}}$ is statistically indistinguishable from a random vector in $\mathbb{Z}_p^k$.

The fact that $\boldsymbol{c}$ is a uniform vector in $\mathbb{Z}_q^{(n+1)}$ also implies that w is a random number in $\mathbb{Z}_q$. Then by the following Claim, $\lfloor\frac{p}{q}w\rceil$ is statistically indistinguishable from a random number in $\mathbb{Z}_p$, i.e., $\hat{w}$ is statistically indistinguishable from a random number in $\mathbb{Z}_p$.

This completes proof of Claim E.13.

Claim E.13.1. *Let $x \xleftarrow{\$} \mathbb{Z}_q$, $y = \lfloor\frac{p}{q}\cdot x\rceil$, and $y' \xleftarrow{\$} \mathbb{Z}_p$, then $y \approx_s y'$.*

Proof. Let $\Delta = \lfloor\frac{q}{p}\rfloor$ and $\delta = \frac{q}{p} - \Delta$. Note that $\Delta \gg 1$ and $0 \le \delta < 1$. For any $b \in \mathbb{Z}_p$, let $a \in \mathbb{Z}_q$ satisfying that $b = \lfloor\frac{p}{q}\cdot a\rceil$ and $b-1 = \lfloor\frac{p}{q}\cdot(a-1)\rceil$. Then we have

$$
\begin{aligned}
&\lfloor\frac{p}{q}\cdot(a+\Delta-1)\rceil && \lfloor\frac{p}{q}\cdot(a+\Delta+1)\rceil\\
=&\lfloor\frac{p}{q}\cdot(a-1)+\frac{p}{q}\cdot\Delta\rceil && =\lfloor\frac{p}{q}\cdot a+\frac{p}{q}\cdot(\Delta+1)\rceil\\
=&\lfloor\frac{p}{q}\cdot(a-1)+\frac{p}{q}\cdot(\frac{q}{p}-\delta)\rceil && =\lfloor\frac{p}{q}\cdot a+\frac{p}{q}\cdot(\frac{q}{p}+1-\delta)\rceil\\
=&\lfloor\frac{p}{q}\cdot(a-1)+1-\frac{p}{q}\cdot\delta\rceil && =\lfloor\frac{p}{q}\cdot a+1+\frac{p}{q}\cdot(1-\delta)\rceil\\
=&1+\lfloor\frac{p}{q}\cdot(a-1)-\frac{p}{q}\cdot\delta\rceil && =1+\lfloor\frac{p}{q}\cdot a+\frac{p}{q}\cdot(1-\delta)\rceil\\
\le&1+b-1=b && \ge 1+b
\end{aligned}
$$

As $\lfloor\frac{p}{q}\cdot(a+\Delta-1)\rceil \ge \lfloor\frac{p}{q}\cdot a\rceil = b$, we always have $\lfloor\frac{p}{q}\cdot(a+\Delta-1)\rceil = b$. Also, we always have $\lfloor\frac{p}{q}\cdot(a+\Delta+1)\rceil > b$.

Thus, for any $b \in \mathbb{Z}_p$, the number of $a \in \mathbb{Z}_q$ s.t. $b = \lfloor\frac{p}{q}\cdot a\rceil$ is at least Δ and is at most $\Delta+1$. This implies that for any $b \in \mathbb{Z}_p$,

$$\frac{\Delta}{q} \le \Pr[a \xleftarrow{\$} \mathbb{Z}_q : b = \lfloor\frac{p}{q}\cdot a\rceil] \le \frac{\Delta+1}{q}$$

Therefore, we have

$$\begin{aligned}
\mathtt{SD}(y, y') &= \frac{1}{2} \cdot \sum_{b \in \mathbb{Z}_p} |\Pr[a \xleftarrow{\$} \mathbb{Z}_q : b = \lfloor \frac{p}{q} \cdot a \rceil] - \frac{1}{p}| \\
&\leq \frac{1}{2} \cdot p \cdot \max(\frac{1}{p} - \frac{\Delta}{q}, \frac{\Delta + 1}{q} - \frac{1}{p}) \\
&= \frac{1}{2} \cdot p \cdot \max(\frac{\delta}{q}, \frac{1 - \delta}{q}) \\
&\leq \frac{1}{2} \cdot p \cdot \frac{1}{q} \\
&= \frac{p}{2q} \leq \frac{1}{2^{\omega(\log \lambda)}}
\end{aligned}$$

which is negligible. □

□

Uniformity of Transformed Ciphertext. We first define the set of bad randomness for $\mathtt{RandPK}$. Recall that the randomness used in the public key randomization algorithm is a uniform matrix $\boldsymbol{R} \in \{0,1\}^{m \times \bar{m}}$. We say that $\boldsymbol{R} \in \mathcal{B}$ if its columns are *not linear independent*, i.e., there exists $\boldsymbol{x} \in \mathbb{Z}_q^{\bar{m}} \backslash \{0^{\bar{m}}\}$ s.t. $\boldsymbol{R}\boldsymbol{x} = 0 \mod q$.

Note that for any fixed $\boldsymbol{x} \in \mathbb{Z}_q^{\bar{m}} \backslash \{0^{\bar{m}}\}$, there exists $i^* \in [1, \bar{m}]$ s.t. $\boldsymbol{x}[i^*] \neq 0$. Also, for any $\boldsymbol{R} \in \{0,1\}^{m \times \bar{m}}$, let $\boldsymbol{r}_i$ be the i-th column of $\boldsymbol{R}$. Then $\boldsymbol{R}\boldsymbol{x} = 0 \mod q$ iff

$$\boldsymbol{r}_{i^*} = \boldsymbol{x}[i^*]^{-1} \cdot (\sum_{j \neq i^*} \boldsymbol{x}[j] \cdot \boldsymbol{r}_j) \mod q$$

Thus, if $\boldsymbol{R} \xleftarrow{\$} \{0,1\}^{m \times \bar{m}}$, then

$$\Pr[\boldsymbol{R} \cdot \boldsymbol{x} = 0 \mod q] = \Pr[\boldsymbol{r}_{i^*} = \boldsymbol{x}[i^*]^{-1} \cdot (\sum_{j \neq i^*} \boldsymbol{x}[j] \cdot \boldsymbol{r}_j) \mod q] \leq \frac{1}{2^m}$$

Therefore, the probability that there exists $\boldsymbol{x} \in \mathbb{Z}_q^{\bar{m}} \backslash \{0^{\bar{m}}\}$ s.t. $\boldsymbol{R}\boldsymbol{x} = 0 \mod q$ does not exceed $\frac{q^{\bar{m}}}{2^m}$, i.e.,

$$\Pr[\boldsymbol{R} \in \mathcal{B}] \leq \frac{q^{\bar{m}}}{2^m} \leq \frac{1}{2^\lambda}$$

which is negligible.

Next, we define the set $\mathcal{C}_0$ and $\mathcal{C}_1$. For any $(pk, sk, ek) \leftarrow \mathtt{KeyGen}(1^\lambda)$, let $sk = \boldsymbol{s}$, we define

$$\mathcal{C}_0 = \{(\boldsymbol{v}^\intercal, w)^\intercal \in \mathbb{Z}_q^{n+1} : \boldsymbol{v}^\intercal \in \mathbb{Z}_q^n \ \wedge \ \exists u \in [0, \frac{q-3}{2}], w = \boldsymbol{s} \cdot \boldsymbol{v} + u \mod q\}$$

$$\mathcal{C}_1 = \{(\boldsymbol{v}^\intercal, w)^\intercal \in \mathbb{Z}_q^{n+1} : \boldsymbol{v}^\intercal \in \mathbb{Z}_q^n \wedge \exists u \in [0, \frac{q-3}{2}], w = \boldsymbol{s} \cdot \boldsymbol{v} + \frac{q+1}{2} + u \mod q\}$$

It is easy to see $\mathcal{C}_0 \cap \mathcal{C}_1 = \emptyset$. Also,

$$\frac{|\mathcal{C}_0|}{q^{n+1}} = \frac{q^n \cdot (\frac{q-1}{2})}{q^{n+1}} = \frac{1}{2} - \frac{1}{2q}$$

$$\frac{|\mathcal{C}_1|}{q^{n+1}} = \frac{q^n \cdot (\frac{q-1}{2})}{q^{n+1}} = \frac{1}{2} - \frac{1}{2q}$$

Both are negligibly close to 1/2.

Uniformity of Transformed Ciphertext. Now, we are ready to prove the uniformity of transformed ciphertext for SFHE. For any message $\mu \in \{0,1\}$ and for any $\boldsymbol{R} \in \{0,1\}^{m \times \bar{m}} \backslash \mathcal{B}$ (i.e., columns of $\boldsymbol{R}$ are linearly independent), let $(pk, sk, ek) \leftarrow \mathtt{KeyGen}(1^\lambda)$ and let $pk = \boldsymbol{A}$, $sk = \boldsymbol{s}$. Let $\boldsymbol{A}' = \boldsymbol{A} \cdot \boldsymbol{R}$. Let $\boldsymbol{C}$ be any valid ciphertext of μ. Also, let $\boldsymbol{c}_0 \leftarrow \mathtt{TranCT}(\boldsymbol{A}', \boldsymbol{C})$ and let $\boldsymbol{c}_1 \xleftarrow{\$} \mathcal{C}_\mu$. Let $\mathtt{info} = (\boldsymbol{A}, \boldsymbol{s}, ek, \boldsymbol{R}, \boldsymbol{C}, \mu)$, we next prove that

$$(\mathtt{info}, \boldsymbol{c}_0) \approx_s (\mathtt{info}, \boldsymbol{c}_1)$$

First, as shown in the proof of Lemma E.28 (Equation (7)), we have

$$\boldsymbol{c}_0 = \begin{pmatrix} \boldsymbol{B} \cdot \boldsymbol{r} + \boldsymbol{B} \cdot \boldsymbol{R} \cdot \boldsymbol{x} \\ \mu \cdot \frac{q+1}{2} + (\boldsymbol{s}^\intercal \cdot \boldsymbol{B} + \boldsymbol{e}^\intercal) \cdot (\boldsymbol{r} + \boldsymbol{R} \cdot \boldsymbol{x}) + z \end{pmatrix} \mod q$$

where $\boldsymbol{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{r} \in [-l \cdot \Sigma', l \cdot \Sigma']^m$ and $\boldsymbol{e} \leftarrow \tilde{\mathcal{D}}_\sigma^m$ are variables determined by $\mathtt{info}$ and $\boldsymbol{x} \xleftarrow{\$} \{0,1\}^{\bar{m}}$ and $z \xleftarrow{\$} [\Sigma, \frac{q+1}{2} - \Sigma]$ are variables sampled in the ciphertext transformation algorithm.

Let $\boldsymbol{v}_0 = \boldsymbol{B} \cdot \boldsymbol{r} + \boldsymbol{B} \cdot \boldsymbol{R} \cdot \boldsymbol{x}$, $E = \boldsymbol{e}^\intercal \cdot (\boldsymbol{r} + \boldsymbol{R} \cdot \boldsymbol{x})$ and $u_0 = E + z$. Also, let $\boldsymbol{v}_1 \xleftarrow{\$} \mathbb{Z}_q^n$ and $u_1 \xleftarrow{\$} [0, \frac{q-3}{2}]$. Let $\mathtt{F}_{\boldsymbol{s},\mu} : \mathbb{Z}_q^n \times \mathbb{Z}_q \to \mathbb{Z}_q^n \times \mathbb{Z}_q$ be a deterministic function that for any $\boldsymbol{v} \in \mathbb{Z}_q^n$, $u \in \mathbb{Z}_q$,

$$\mathtt{F}_{\boldsymbol{s},\mu}(\boldsymbol{v}, u) = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{s} \cdot \boldsymbol{v} + \mu \cdot \frac{q+1}{2} + u \end{pmatrix}$$

Then it is easy to see $\boldsymbol{c}_0 = \mathtt{F}_{\boldsymbol{s},\mu}(\boldsymbol{v}_0, u_0)$ and $\boldsymbol{c}_1 = \mathtt{F}_{\boldsymbol{s},\mu}(\boldsymbol{v}_1, u_1)$. Also, let

$$\mathtt{G}(\boldsymbol{A}, \boldsymbol{s}, ek, \boldsymbol{R}, \boldsymbol{C}, \mu, \boldsymbol{v}, u) = (\boldsymbol{A}, \boldsymbol{s}, ek, \boldsymbol{R}, \boldsymbol{C}, \mu, \mathtt{F}_{\boldsymbol{s},\mu}(\boldsymbol{v}, u))$$

then we have $\mathtt{G}(\mathtt{info}, \boldsymbol{v}_0, u_0) = (\mathtt{info}, \boldsymbol{c}_0)$ and $\mathtt{G}(\mathtt{info}, \boldsymbol{v}_1, u_1) = (\mathtt{info}, \boldsymbol{c}_1)$. Thus, by Lemma B.2,

$$\mathtt{SD}((\mathtt{info}, \boldsymbol{c}_0), (\mathtt{info}, \boldsymbol{c}_1)) \leq \mathtt{SD}((\mathtt{info}, \boldsymbol{v}_0, u_0), (\mathtt{info}, \boldsymbol{v}_1, u_1)) \tag{9}$$

We next prove that $\mathtt{SD}((\mathtt{info}, \boldsymbol{v}_0, u_0), (\mathtt{info}, \boldsymbol{v}_1, u_1)) \leq negl(\lambda)$.

Since columns of $\boldsymbol{R}$ are linearly independent, for any distinct $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \{0,1\}^{\bar{m}}$, $\boldsymbol{R} \cdot \boldsymbol{x}_1 \neq \boldsymbol{R} \cdot \boldsymbol{x}_2 \mod q$. Thus, the min-entropy of $\boldsymbol{R} \cdot \boldsymbol{x}$ is $\bar{m}$. Then we have

$$\begin{aligned} H_\infty(\boldsymbol{R} \cdot \boldsymbol{x} \mid \mathtt{info}, u_0) &\geq H_\infty(\boldsymbol{R} \cdot \boldsymbol{x} \mid \mathtt{info}) - \log q \\ &= \bar{m} - \log q \geq (n+1)l + \lambda - l = nl + \lambda \geq n \log q + \lambda \end{aligned}$$

where the first inequality comes from Lemma B.3 and the second equality comes from the fact that $\boldsymbol{x}$ and $\mathtt{info}$ are independent. In addition, $\boldsymbol{B}$ is a random matrix, then by the leftover hash lemma,

$$(\mathtt{info}, \boldsymbol{v}_0, u_0) \approx_s (\mathtt{info}, \boldsymbol{v}_1, u_0) \tag{10}$$

It remains to prove that $(\mathtt{info}, \boldsymbol{v}_1, u_0) \approx_s (\mathtt{info}, \boldsymbol{v}_1, u_1)$.

We have shown in the proof of Lemma E.28 (Equation (8)) that $E \in [-\Sigma, \Sigma]$. Also note that $z \xleftarrow{\$} [\Sigma, \frac{q+1}{2} - \Sigma]$. Thus, for any α, β and for any $a \in [2\Sigma, \frac{q+1}{2} - 2\Sigma]$, we have

$$\begin{aligned}
&\Pr[u_0 = a \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \\
=&\sum_{b \in [-\Sigma, \Sigma]} \Pr[E = b \wedge z = a - b \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \\
=&\sum_{b \in [-\Sigma, \Sigma]} \Pr[E = b \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \cdot \Pr[z = a - b] \\
=&\sum_{b \in [-\Sigma, \Sigma]} \Pr[E = b \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \cdot \frac{1}{Q} \\
=&\frac{1}{Q}
\end{aligned}$$

where $Q = \frac{q+1}{2} - 2\Sigma + 1$. Next, let $\mathcal{T}$ be a set that for ay $(\alpha, \beta, a) \in \mathcal{T}$,

$$\Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta \wedge u_0 = a] < \Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta \wedge u_1 = a]$$

Note that for any α, β, a,

$$\begin{aligned}
&\Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta \wedge u_1 = a] - \Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta \wedge u_0 = a] \\
=&\Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \cdot (\Pr[u_1 = a \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] - \Pr[u_0 = a \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta]) \\
=&\Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \cdot (\Pr[u_1 = a] - \Pr[u_0 = a \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta])
\end{aligned}$$

As for any $a \in [2\Sigma, \frac{q+1}{2} - 2\Sigma]$ and for any α, β,

$$\Pr[u_0 = a \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] = \frac{1}{Q} > \frac{1}{(q-1)/2} = \Pr[u_1 = a]$$

and for any $a \notin [0, \frac{q-3}{2}]$, $\Pr[u_1 = a] = 0$, we must have $a \in [0, 2\Sigma) \cup (\frac{q+1}{2} - 2\Sigma, \frac{q-3}{2}]$ if we hope $(\alpha, \beta, a) \in \mathcal{T}$. Also, since for any $a \in [0, 2\Sigma) \cup (\frac{q+1}{2} - 2\Sigma, \frac{q-3}{2}]$, $\Pr[u_1 = a] = \frac{1}{(q-1)/2}$, we have

$$\Pr[u_1 = a] - \Pr[u_0 = a \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \leq \frac{1}{(q-1)/2}$$

for any $a \in [0, 2\Sigma) \cup (\frac{q+1}{2} - 2\Sigma, \frac{q-3}{2}]$.

Then by Lemma B.2,

$$
\begin{aligned}
&\mathtt{SD}((\mathtt{info}, \boldsymbol{v}_1, u_0), (\mathtt{info}, \boldsymbol{v}_1, u_1)) \\
=&\sum_{(\alpha,\beta,a)\in\mathcal{T}} (\Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta \wedge u_1 = a] - \Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta \wedge u_0 = a]) \\
=&\sum_{(\alpha,\beta,a)\in\mathcal{T}} (\Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \cdot (\Pr[u_1 = a] - \Pr[u_0 = a \mid \mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta])) \\
\leq&\frac{1}{(q-1)/2} \cdot \sum_{(\alpha,\beta,a)\in\mathcal{T}} \Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta] \\
=&\frac{1}{(q-1)/2} \cdot (\sum_{a\in[0,2\Sigma)\cup(\frac{q+1}{2}-2\Sigma,\frac{q-3}{2}]} \sum_{(\alpha,\beta)s.t.(\alpha,\beta,a)\in\mathcal{T}} \Pr[\mathtt{info} = \alpha \wedge \boldsymbol{v}_1 = \beta]) \\
\leq&\frac{1}{(q-1)/2} \cdot (\sum_{a\in[0,2\Sigma)\cup(\frac{q+1}{2}-2\Sigma,\frac{q-3}{2}]} 1) \\
=&(4\Sigma - 2) \cdot \frac{1}{(q+1)/2} \\
\leq&\frac{1}{2^{\omega(\log\lambda)}}
\end{aligned}
\tag{11}
$$

which is negligible.

Combining Equation (9), (10) and (11), the uniformity of transformed ciphertext property follows.

E.6 Security Analysis of Robust Unobfuscatable PRFs from Fully Homomorphic Encryption

We present proof of Theorem 6.2 in this section. More precisely, we will prove the correctness, $(\frac{1}{6} - \varepsilon)$-robust learnability, and black-box pseudorandomness of UOF.

Correctness. For any message $msg \in \{0,1\}^{m_2}$, let $K = (\alpha, \beta, k_{IR}, k_{mask}, pk, sk, ek, msg)$ be the output of the key generation algorithm $\mathtt{KeyGen}(1^\lambda, msg)$. We prove correctness of UOF via arguing the following Lemmas.

Lemma E.29. *Let $\boldsymbol{ct}_\beta$ be any vector of valid ciphertexts of β (under (pk, sk, ek)), $rp\mathcal{k}$ and $re\mathcal{k}$ be any outputs of $\mathsf{SFHE}.\mathtt{RandPK}(pk)$ and $\mathsf{SFHE}.\mathtt{RandEK}(pk, ek)$ respectively. Let $msg' \leftarrow \mathtt{ExtractIII}(\mathtt{Eval}(K, \cdot), \boldsymbol{ct}_\beta, rp\mathcal{k}, re\mathcal{k})$, then we have $msg = msg'$.*

Proof. Let $\psi^{(i)}, \tilde{u}_0^{(i)}, u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)}, \gamma^{(i)}, \boldsymbol{ct}_{\beta,\gamma}^{(i)}, (tct_1^{(i)}, \ldots, tct_\lambda^{(i)}), x_3^{(i)}, y_3^{(i)}, \tilde{x}_3^{(i)}, \tilde{y}_3^{(i)}, msg^{(i)}$ be variables used in the i-th repetition when running the algorithm $\mathtt{ExtractIII}(\mathtt{Eval}(K, \cdot), \boldsymbol{ct}_\beta, rp\mathcal{k}, re\mathcal{k})$.

First, as $\boldsymbol{ct}_\beta$ are valid ciphertexts of β and $re\mathcal{k}$ is a randomized evaluation key of ek, $\boldsymbol{ct}_{\beta,\gamma}^{(i)}$ are valid ciphertexts of $\beta \oplus \gamma^{(i)}$. Then by the perfect correctness

of SFHE, $\mathsf{SFHE}.\mathtt{TCTDec}(sk, tct_j^{(i)}) = (\beta \oplus \gamma^{(i)})[j]$ for $j \in [1, \lambda]$. Thus, we have

$$\begin{aligned}
&y_3^{(i)}[m_1+1, m] \\
=&\mathtt{Eval}(K, (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}))[m_1+1, m] \\
=&msg \oplus \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)} - 2, u_1^{(i)}, u_2^{(i)} \oplus \beta \oplus \gamma^{(i)} \oplus \beta, u_3^{(i)})) \\
=&msg \oplus \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)} - 2, u_1^{(i)}, u_2^{(i)} \oplus \gamma^{(i)}, u_3^{(i)})
\end{aligned}$$

Also, we have

$$\begin{aligned}
&\tilde{y}_3^{(i)}[m_1+1, m] \\
=&\mathtt{Eval}(K, (\tilde{u}_0^{(i)}, u_1^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)}))[m_1+1, m] \\
=&\mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (\tilde{u}_0^{(i)}, u_1^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)})) \\
=&\mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)} - 2, u_1^{(i)}, u_2^{(i)} \oplus \gamma^{(i)}, u_3^{(i)})
\end{aligned}$$

Thus, we have $msg^{(i)} = (y_3^{(i)} \oplus \tilde{y}_3^{(i)})[m_1+1 : m] = msg$. As $msg^{(i)} = msg$ for all $i \in [1, N]$, we have $msg' = msg$. □

Lemma E.30. *Let $\boldsymbol{ct}_\alpha$ be any vector of valid ciphertexts of α, $rp\mathcal{K}$ and $re\mathcal{K}$ be any outputs of $\mathsf{SFHE}.\mathtt{RandPK}(pk)$ and $\mathsf{SFHE}.\mathtt{RandEK}(pk, ek)$ respectively. Let $msg' \leftarrow \mathtt{ExtractII}(\mathtt{Eval}(K, \cdot), \boldsymbol{ct}_\alpha, rp\mathcal{K}, re\mathcal{K})$, then we have $msg = msg'$.*

Proof. Let $\psi^{(i)}, u_0^{(i)}, \tilde{u}_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}, x_2^{(i)}, y_2^{(i)}, \mathtt{P}^{(i)}, \boldsymbol{ct}_\beta^{(i)}, msg^{(i)}$ be variables used in the i-th repetition when running the algorithm $\mathtt{ExtractII}(\mathtt{Eval}(K, \cdot), \boldsymbol{ct}_\alpha, rp\mathcal{K}, re\mathcal{K})$.

First, we have

$$\begin{aligned}
&y_2^{(i)}[m_1+1, m] \\
=&\mathtt{Eval}(K, (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}))[m_1+1, m] \\
=&\mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}))
\end{aligned}$$

Next, let $\tilde{y}_2^{(i)}$ be the variable $\tilde{y}_2$ used in evaluating the circuit $\mathtt{P}^{(i)}(\alpha)$, then we have

$$\begin{aligned}
&\tilde{y}_2^{(i)}[m_1+1, m] \\
=&\mathtt{Eval}(K, (\tilde{u}_0^{(i)}, u_1^{(i)}, u_2^{(i)} \oplus \alpha, u_3^{(i)}))[m_1+1, m] \\
=&(\beta \| 0^{m_2-\lambda}) \oplus \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (\tilde{u}_0^{(i)} - 1, u_1^{(i)}, u_2^{(i)} \oplus \alpha \oplus \alpha, u_3^{(i)})) \\
=&(\beta \| 0^{m_2-\lambda}) \oplus \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}))
\end{aligned}$$

Thus, we have $\mathtt{P}^{(i)}(\alpha) = (\tilde{y}_2^{(i)} \oplus y_2^{(i)})[m_1+1, m_1+\lambda] = \beta$.

Next, as $\boldsymbol{ct}_\alpha$ are valid ciphertexts of α and $re\mathcal{K}$ is a randomized evaluation key of ek, $\boldsymbol{ct}_\beta^{(i)}$ are valid ciphertexts of $\mathtt{P}^{(i)}(\alpha) = \beta$. Then by Lemma E.29, $\mathtt{ExtractIII}(\mathtt{Eval}(K, \cdot), \boldsymbol{ct}_\beta^{(i)}, rp\mathcal{K}, re\mathcal{K}) = msg$, i.e., $msg^{(i)} = msg$. As $msg^{(i)} = msg$ for all $i \in [1, N]$, we have $msg' = msg$. □

Lemma E.31. *Let* $msg' \leftarrow \mathtt{ExtractI}(\mathtt{Eval}(K,\cdot))$*, then we have* $msg = msg'$*.*

Proof. Let $x_1^{(i)}, y_1^{(i)}, \boldsymbol{ct}_\alpha^{(i)}, \mathit{rp\mathcal{K}}^{(i)}, \mathit{re\mathcal{K}}^{(i)}, msg^{(i)}$ be variables used in the i-th repetition when running the algorithm $\mathtt{ExtractI}(\mathtt{Eval}(K,\cdot))$.

It is easy to see $\boldsymbol{ct}_\alpha^{(i)}[j] \leftarrow \mathsf{SFHE}.\mathtt{Enc}(pk, \alpha[j])$ for $j \in [1,\lambda]$, $\mathit{rp\mathcal{K}}^{(i)} \leftarrow \mathsf{SFHE}.\mathtt{RandPK}(pk)$, and $\mathit{re\mathcal{K}}^{(i)} \leftarrow \mathsf{SFHE}.\mathtt{RandEK}(pk, ek)$. Then by definition of valid ciphertexts of SFHE, $\boldsymbol{ct}_\alpha^{(i)}$ is a vector of valid ciphertexts for α. Next, by Lemma E.30, $\mathtt{ExtractII}(\mathtt{Eval}(K,\cdot), \boldsymbol{ct}_\alpha^{(i)}, \mathit{rp\mathcal{K}}^{(i)}, \mathit{re\mathcal{K}}^{(i)}) = msg$, i.e., $msg^{(i)} = msg$. As $msg^{(i)} = msg$ for all $i \in [1,N]$, we have $msg' = msg$. □

Correctness of UOF comes from Lemma E.31 directly.

Robust Learnability. For any message $msg \in \{0,1\}^{m_2}$, let $K = (\alpha, \beta, k_{IR}, k_{mask}, pk, sk, ek, msg)$ be the output of the key generation algorithm $\mathtt{KeyGen}(1^\lambda, msg)$. Let $\mathtt{C}$ be any circuit from $\{0,1\}^n$ to $\{0,1\}^m$ s.t. $|\{x \in \{0,1\}^n : \mathtt{C}(x) \neq \mathtt{Eval}(K,x)\}| \leq (\frac{1}{6} - \varepsilon)\cdot 2^n$. We prove $(\frac{1}{6}-\varepsilon)$-robust learnability of UOF via arguing the following Lemmas.

Lemma E.32. *Let* $\boldsymbol{ct}_\beta$ *be any vector of valid ciphertexts of* β*, and let* $\mathit{re\mathcal{K}}$ *be any output of* $\mathsf{SFHE}.\mathtt{RandEK}(pk, ek)$*. Let* $r_{rpk} \in \{0,1\}^{l_{rpk}}$ *be any string that is not in the "bad randomness subset"* $\mathcal{B}$ *for* $\mathsf{SFHE}.\mathtt{RandPK}$ *(the set is defined in Appendix D.1), and let* $\mathit{rp\mathcal{K}} = \mathsf{SFHE}.\mathtt{RandPK}(pk; r_{rpk})$*. Let* $msg' \leftarrow \mathtt{ExtractIII}(\mathtt{C}, \boldsymbol{ct}_\beta, \mathit{rp\mathcal{K}}, \mathit{re\mathcal{K}})$*, then we have* $\Pr[msg \neq msg'] \leq negl(\lambda)$*.*

Proof. Let $\psi^{(i)}, \tilde{u}_0^{(i)}, u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)}, \gamma^{(i)}, \boldsymbol{ct}_{\beta,\gamma}^{(i)}, (\mathit{tct}_1^{(i)}, \ldots, \mathit{tct}_\lambda^{(i)}), x_3^{(i)}, y_3^{(i)}, \tilde{x}_3^{(i)}, \tilde{y}_3^{(i)}, msg^{(i)}$ be variables used in the i-th repetition when running the algorithm $\mathtt{ExtractIII}(\mathtt{C}, \boldsymbol{ct}_\beta, \mathit{rp\mathcal{K}}, \mathit{re\mathcal{K}})$.

First, we define the following two deterministic functions[27]:

$$f(\psi, u_1, u_2, u_3) = (3\cdot\psi + 2, u_1, u_2, u_3)$$
$$g(\psi, u_1, u_2, u_3) = (3\cdot\psi, u_1, u_2 \oplus \mathtt{D}(sk, u_3) \oplus \beta, u_3)$$

where

$$\mathtt{D}(sk, u_3) = (\mathsf{SFHE}.\mathtt{TCTDec}(sk, u_3[(j-1)L_t + 1 : jL_t]))_{j\in[1,\lambda]}$$

Then, we define a subset $\mathcal{G}$ of $[0, \lfloor\frac{2^\lambda}{3}\rfloor - 1] \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3}$ as:

$$\begin{aligned}\mathcal{G} = \{(\psi, u_1, u_2, u_3) \in [0, \lfloor\frac{2^\lambda}{3}\rfloor - 1] \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} :\\ \exists x, \tilde{x} \in \{0,1\}^n,\ x = f(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(x) = \mathtt{Eval}(K, x) \wedge\\ \tilde{x} = g(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(\tilde{x}) = \mathtt{Eval}(K, \tilde{x})\}\end{aligned}$$

Next, we prove a few facts about $\mathcal{G}$.

[27] As $\mathsf{SFHE}.\mathtt{TCTDec}$ is a deterministic algorithm, $\mathtt{D}$ is also deterministic, thus the two functions are deterministic.

Claim E.14. *Let* $\psi \xleftarrow{\$} [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1]$, $u_1 \xleftarrow{\$} \{0,1\}^{n_1}$, $u_2 \xleftarrow{\$} \{0,1\}^{n_2}$, *and* $u_3 \xleftarrow{\$} \{0,1\}^{n_3}$, *then we have*

$$\Pr[(\psi, u_1, u_2, u_3) \in \mathcal{G}] \geq \frac{1}{2} + 2\varepsilon$$

Proof. Let $\mathcal{U} = [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1] \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3}$ and let

$$\mathcal{B}_1 = \{(\psi, u_1, u_2, u_3) \in \mathcal{U} : \exists x \in \{0,1\}^n, x = f(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(x) \neq \mathtt{Eval}(K, x)\}$$

$$\mathcal{B}_2 = \{(\psi, u_1, u_2, u_3) \in \mathcal{U} : \exists \tilde{x} \in \{0,1\}^n, \tilde{x} = g(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(\tilde{x}) \neq \mathtt{Eval}(K, \tilde{x})\}$$

First, it is easy to see for any $(\psi, u_1, u_2, u_3) \in \mathcal{U}$, $(\psi, u_1, u_2, u_3) \in \mathcal{G}$ if and only if $(\psi, u_1, u_2, u_3) \notin \mathcal{B}_1$ and $(\psi, u_1, u_2, u_3) \notin \mathcal{B}_2$. Thus, we have

$$|\mathcal{G}| = |\mathcal{U}| - |\mathcal{B}_1 \cup \mathcal{B}_2| \geq |\mathcal{U}| - (|\mathcal{B}_1| + |\mathcal{B}_2|) \tag{12}$$

Next, we bound the size of $\mathcal{B}_1$ and $\mathcal{B}_2$.

Claim E.14.1. $|\mathcal{B}_1| + |\mathcal{B}_2| \leq (\frac{1}{6} - \varepsilon) \cdot 2^n$.

Proof. Let

$$\bar{\mathcal{B}}_1 = \{x \in \{0,1\}^n : \exists (\psi, u_1, u_2, u_3) \in \mathcal{B}_1, x = f(\psi, u_1, u_2, u_3)\}$$

$$\bar{\mathcal{B}}_2 = \{x \in \{0,1\}^n : \exists (\psi, u_1, u_2, u_3) \in \mathcal{B}_2, x = g(\psi, u_1, u_2, u_3)\}$$

Obviously, f is an injective function. Thus, we have $|\bar{\mathcal{B}}_1| = |\mathcal{B}_1|$.

Also, assuming g is not injective, then there exists distinct tuples (ψ, u_1, u_2, u_3) and $(\psi', u_1', u_2', u_3')$ s.t. $g(\psi, u_1, u_2, u_3) = g(\psi', u_1', u_2', u_3')$. First, we have $u_1 = u_1'$ and $u_3 = u_3'$. Also, since $\mathtt{D}$ is deterministic, $\mathtt{D}(sk, u_3)$ is always equal to $\mathtt{D}(sk, u_3')$. This implies that $u_2 = u_2'$. Finally, as $3\psi = 3\psi'$, $\psi = \psi'$. This contradicts the assumption that g is not injective. Therefore, g is injective and we also have $|\bar{\mathcal{B}}_2| = |\mathcal{B}_2|$.

Besides, as for any $x = (u_0, u_1, u_2, u_3) \in \bar{\mathcal{B}}_1$, $u_0 = 2 \mod 3$ and for any $x' = (u_0', u_1', u_2', u_3') \in \bar{\mathcal{B}}_2$, $u_0' = 0 \mod 3$, we have $x \neq x'$, i.e., $\bar{\mathcal{B}}_1 \cap \bar{\mathcal{B}}_2 = \emptyset$. Therefore,

$$|\bar{\mathcal{B}}_1 \cup \bar{\mathcal{B}}_2| = |\bar{\mathcal{B}}_1| + |\bar{\mathcal{B}}_2| = |\mathcal{B}_1| + |\mathcal{B}_2|$$

Finally, since for any $x \in \bar{\mathcal{B}}_1 \cup \bar{\mathcal{B}}_2$, $\mathtt{C}(x) \neq \mathtt{Eval}(K, x)$, we have

$$|\mathcal{B}_1| + |\mathcal{B}_2| = |\bar{\mathcal{B}}_1 \cup \bar{\mathcal{B}}_2| \leq |\{x \in \{0,1\}^n : \mathtt{C}(x) \neq \mathtt{Eval}(K, x)\}| \leq (\frac{1}{6} - \varepsilon) \cdot 2^n$$

□

By Equation (12) and Claim E.14.1, we have

$$\begin{aligned}\frac{|\mathcal{G}|}{|\mathcal{U}|} &\geq \frac{|\mathcal{U}| - (|\mathcal{B}_1| + |\mathcal{B}_2|)}{|\mathcal{U}|} \\ &\geq 1 - \frac{(\frac{1}{6} - \varepsilon) \cdot 2^n}{\lfloor \frac{2^\lambda}{3} \rfloor \cdot 2^{n_1} \cdot 2^{n_2} \cdot 2^{n_3}} \\ &= 1 - \frac{(\frac{1}{6} - \varepsilon) \cdot 2^\lambda}{\lfloor \frac{2^\lambda}{3} \rfloor} \\ &\geq 1 - \frac{(\frac{1}{6} - \varepsilon) \cdot (3 \cdot \lfloor \frac{2^\lambda}{3} \rfloor + 3)}{\lfloor \frac{2^\lambda}{3} \rfloor} \\ &= 1 - (\frac{1}{2} - 3\varepsilon) - \frac{\frac{1}{2} - 3\varepsilon}{\lfloor \frac{2^\lambda}{3} \rfloor} \\ &\geq \frac{1}{2} + 2\varepsilon\end{aligned} \tag{13}$$

Therefore, the probability that $(\psi, u_1, u_2, u_3) \xleftarrow{\$} \mathcal{U}$ is in $\mathcal{G}$ is at least $\frac{1}{2} + 2\varepsilon$. □

Claim E.15. *For any $i \in [1, N]$, $\Pr[(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}] \geq \frac{1}{2} + \varepsilon$.*

Proof. First, as $\boldsymbol{ct}_\beta$ are valid ciphertexts of β and rek is a randomized evaluation key of ek, $\boldsymbol{ct}_{\beta,\gamma}^{(i)}$ are valid ciphertexts of $\beta \oplus \gamma^{(i)}$. Since $r_{rpk} \in \{0,1\}^{l_{rpk}} \setminus \mathcal{B}$ and $\mathit{rpk} = \mathsf{SFHE}.\mathtt{RandPK}(pk; r_{rpk})$, by the uniformity of transformed ciphertext property of SFHE, we have $\mathit{tct}_j^{(i)} \approx_s \mathit{tct}_j'^{(i)}$ for $j \in [1, \lambda]$, where $\mathit{tct}_j'^{(i)} \xleftarrow{\$} \mathcal{C}_{\beta \oplus \gamma^{(i)}[j]}$ (this set is defined in Appendix D.1).[28] Then by Corollary D.1 and the fact that γ is sampled uniformly at random, we have $\mathit{tct}_j^{(i)} \approx_s \mathit{tct}_j''^{(i)}$, where $\mathit{tct}_j''^{(i)} \xleftarrow{\$} \{0,1\}^{L_t}$. Thus, we have $u_3^{(i)} \approx_s u_3'^{(i)}$, where $u_3'^{(i)} \xleftarrow{\$} \{0,1\}^{n_3}$. As $\psi^{(i)} \xleftarrow{\$} [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1]$, $u_1^{(i)} \xleftarrow{\$} \{0,1\}^{n_1}$, and $u_2^{(i)} \xleftarrow{\$} \{0,1\}^{n_2}$, by Claim E.14 we have

$$\Pr[(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}] \geq \Pr[(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3'^{(i)}) \in \mathcal{G}] - negl(\lambda) \geq \frac{1}{2} + \varepsilon$$

□

Claim E.16. *For any $i \in [1, N]$, if $(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}$, then $msg^{(i)} = msg$.*

Proof. First, as $u_0^{(i)} = 3\psi^{(i)} + 2$ and $x_3^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$, $x_3^{(i)} = f(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$. Since $(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}$, we have $\mathtt{C}(x_3^{(i)}) = \mathtt{Eval}(K, x_3^{(i)})$.

Also, as $\boldsymbol{ct}_\beta$ are valid ciphertexts of β and rek is a randomized evaluation key of ek, $\boldsymbol{ct}_{\beta,\gamma}^{(i)}$ are valid ciphertexts of $\beta \oplus \gamma^{(i)}$. Then by the perfect correctness of SFHE, $\mathsf{SFHE}.\mathtt{TCTDec}(sk, \mathit{tct}_j^{(i)}) = (\beta \oplus \gamma^{(i)})[j]$ for $j \in [1, \lambda]$. Thus, we have

$$\tilde{u}_2^{(i)} = u_2^{(i)} \oplus \gamma^{(i)} = u_2^{(i)} \oplus (\beta \oplus \gamma^{(i)}) \oplus \beta = u_2^{(i)} \oplus \mathtt{D}(sk, u_3^{(i)}) \oplus \beta$$

[28] Note that the indistinguishability holds even when the distinguisher is given K.

In addition, we have $\tilde{u}_0^{(i)} = 3\psi^{(i)}$. Therefore, we have $\tilde{x}_3^{(i)} = (\tilde{u}_0^{(i)}, u_1^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)}) = g(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$, which implies $\mathtt{C}(\tilde{x}_3^{(i)}) = \mathtt{Eval}(K, \tilde{x}_3^{(i)})$.

Now, we have

$$\begin{aligned}
& y_3^{(i)}[m_1+1, m] \\
=& \mathtt{C}(u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})[m_1+1, m] \\
=& \mathtt{Eval}(K, (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}))[m_1+1, m] \\
=& msg \oplus \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)} - 2, u_1^{(i)}, u_2^{(i)} \oplus \beta \oplus \gamma^{(i)} \oplus \beta, u_3^{(i)})) \\
=& msg \oplus \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)} - 2, u_1^{(i)}, u_2^{(i)} \oplus \gamma^{(i)}, u_3^{(i)})
\end{aligned}$$

Also, we have

$$\begin{aligned}
& \tilde{y}_3^{(i)}[m_1+1, m] \\
=& \mathtt{C}(\tilde{u}_0^{(i)}, u_1^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)})[m_1+1, m] \\
=& \mathtt{Eval}(K, (\tilde{u}_0^{(i)}, u_1^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)}))[m_1+1, m] \\
=& \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (\tilde{u}_0^{(i)}, u_1^{(i)}, \tilde{u}_2^{(i)}, u_3^{(i)})) \\
=& \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)} - 2, u_1^{(i)}, u_2^{(i)} \oplus \gamma^{(i)}, u_3^{(i)})
\end{aligned}$$

Thus, we have $msg^{(i)} = (y_3^{(i)} \oplus \tilde{y}_3^{(i)})[m_1+1 : m] = msg$. □

Combining Claim E.15 and Claim E.16, we have $\Pr[msg^{(i)} = msg] \geq \frac{1}{2} + \varepsilon$. Let X_i be a random variable over $\{0,1\}$ that $X_i = 1$ iff $msg^{(i)} = msg$. Let $X = \sum_{i=1}^{N} X_i$ and let $\mu = \sum_{i=1}^{N} \Pr[msg^{(i)} = msg]$. Note that $\mu \geq \frac{1+2\varepsilon}{2} \cdot N$ and thus we have $(1 - \frac{2\varepsilon}{1+2\varepsilon}) \cdot \mu \geq \frac{1}{1+2\varepsilon} \cdot \frac{1+2\varepsilon}{2} \cdot N = \frac{N}{2}$. Then by the Chernoff bound, we have

$$\begin{aligned}
& \Pr[X \leq \frac{N}{2}] \\
\leq & \Pr[X \leq (1 - \frac{2\varepsilon}{1+2\varepsilon}) \cdot \mu] \\
\leq & e^{-\frac{4\varepsilon^2}{2(1+2\varepsilon)^2} \cdot \mu} \\
\leq & e^{-\frac{4\varepsilon^2}{2(1+2\varepsilon)^2} \cdot \frac{1+2\varepsilon}{2} \cdot N} \\
= & e^{-\frac{\varepsilon^2}{1+2\varepsilon} \cdot N} \leq e^{-\frac{\varepsilon^2}{3} \cdot N} = e^{-\frac{\lambda}{3}}
\end{aligned}$$

which is negligible. This implies that with all but negligible probability, the number of $i \in [1, N]$ s.t. $msg = msg^{(i)}$ are over $\frac{N}{2}$, and thus we have $\Pr[msg' \neq msg] \leq negl(\lambda)$. □

Lemma E.33. *Let $\boldsymbol{ct}_\alpha$ be any vector of valid ciphertexts of α, and let rek be any output of $\mathsf{SFHE}.\mathtt{RandEK}(pk, ek)$. Let $r_{rpk} \in \{0,1\}^{l_{rpk}}$ be any string that is not in the "bad randomness subset" $\mathcal{B}$ for $\mathsf{SFHE}.\mathtt{RandPK}$ (the set is defined in Appendix D.1), and let $\mathit{rpk} = \mathsf{SFHE}.\mathtt{RandPK}(pk; r_{rpk})$. Let $msg' \leftarrow \mathtt{ExtractII}(\mathtt{C}, \boldsymbol{ct}_\alpha, \mathit{rpk}, \mathit{rek})$, then we have $\Pr[msg \neq msg'] \leq negl(\lambda)$.*

Proof. Let $\psi^{(i)}, u_0^{(i)}, \tilde{u}_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}, x_2^{(i)}, y_2^{(i)}, \mathtt{P}^{(i)}, \boldsymbol{ct}_\beta^{(i)}, msg^{(i)}$ be variables used in the i-th repetition when running the algorithm $\mathtt{ExtractII}(\mathtt{C}, \boldsymbol{ct}_\alpha, rp\mathcal{K}, re\mathcal{K})$. Also, let $\tilde{x}_2^{(i)}, \tilde{y}_2^{(i)}$ be the variable $\tilde{x}_2, \tilde{y}_2$ used in evaluating the circuit $\mathtt{P}^{(i)}(\alpha)$.

First, we define the following two deterministic functions:

$$f(\psi, u_1, u_2, u_3) = (3 \cdot \psi, u_1, u_2, u_3)$$
$$g(\psi, u_1, u_2, u_3) = (3 \cdot \psi + 1, u_1, u_2 \oplus \alpha, u_3)$$

Then, we define a subset $\mathcal{G}$ of $[0, \lfloor \frac{2^\lambda}{3} \rfloor - 1] \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3}$ as:

$$\begin{aligned}\mathcal{G} = \{(\psi, u_1, u_2, u_3) \in [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1] \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3} : \\ \exists x, \tilde{x} \in \{0,1\}^n,\ x = f(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(x) = \mathtt{Eval}(K, x) \wedge \\ \tilde{x} = g(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(\tilde{x}) = \mathtt{Eval}(K, \tilde{x})\}\end{aligned}$$

Next, we prove a few facts about $\mathcal{G}$.

Claim E.17. *Let* $\psi \xleftarrow{\$} [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1]$, $u_1 \xleftarrow{\$} \{0,1\}^{n_1}$, $u_2 \xleftarrow{\$} \{0,1\}^{n_2}$, *and* $u_3 \xleftarrow{\$} \{0,1\}^{n_3}$, *then we have*

$$\Pr[(\psi, u_1, u_2, u_3) \in \mathcal{G}] \geq \frac{1}{2} + 2\varepsilon$$

Proof. Proof of Claim E.17 is similar to the proof of Claim E.14. We give the detailed proof below for completeness.

Let $\mathcal{U} = [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1] \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3}$ and let

$$\mathcal{B}_1 = \{(\psi, u_1, u_2, u_3) \in \mathcal{U} : \exists x \in \{0,1\}^n, x = f(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(x) \neq \mathtt{Eval}(K, x)\}$$

$$\mathcal{B}_2 = \{(\psi, u_1, u_2, u_3) \in \mathcal{U} : \exists \tilde{x} \in \{0,1\}^n, \tilde{x} = g(\psi, u_1, u_2, u_3) \wedge \mathtt{C}(\tilde{x}) \neq \mathtt{Eval}(K, \tilde{x})\}$$

Again, it is easy to see for any $(\psi, u_1, u_2, u_3) \in \mathcal{U}$, $(\psi, u_1, u_2, u_3) \in \mathcal{G}$ if and only if $(\psi, u_1, u_2, u_3) \notin \mathcal{B}_1$ and $(\psi, u_1, u_2, u_3) \notin \mathcal{B}_2$. Thus, we have

$$|\mathcal{G}| = |\mathcal{U}| - |\mathcal{B}_1 \cup \mathcal{B}_2| \geq |\mathcal{U}| - (|\mathcal{B}_1| + |\mathcal{B}_2|) \tag{14}$$

Next, we bound the size of $\mathcal{B}_1$ and $\mathcal{B}_2$.

Claim E.17.1. $|\mathcal{B}_1| + |\mathcal{B}_2| \leq (\frac{1}{6} - \varepsilon) \cdot 2^n$.

Proof. Let

$$\bar{\mathcal{B}}_1 = \{x \in \{0,1\}^n : \exists (\psi, u_1, u_2, u_3) \in \mathcal{B}_1, x = f(\psi, u_1, u_2, u_3)\}$$

$$\bar{\mathcal{B}}_2 = \{x \in \{0,1\}^n : \exists (\psi, u_1, u_2, u_3) \in \mathcal{B}_2, x = g(\psi, u_1, u_2, u_3)\}$$

Obviously, f and g are injective functions. Thus, we have $|\bar{\mathcal{B}}_1| = |\mathcal{B}_1|$ and $|\bar{\mathcal{B}}_2| = |\mathcal{B}_2|$.

Besides, as for any $x = (u_0, u_1, u_2, u_3) \in \bar{\mathcal{B}}_1$, $u_0 = 0 \mod 3$ and for any $x' = (u'_0, u'_1, u'_2, u'_3) \in \bar{\mathcal{B}}_2$, $u'_0 = 1 \mod 3$, we have $x \neq x'$, i.e., $\bar{\mathcal{B}}_1 \cap \bar{\mathcal{B}}_2 = \emptyset$. Therefore,

$$|\bar{\mathcal{B}}_1 \cup \bar{\mathcal{B}}_2| = |\bar{\mathcal{B}}_1| + |\bar{\mathcal{B}}_2| = |\mathcal{B}_1| + |\mathcal{B}_2|$$

Finally, since for any $x \in \bar{\mathcal{B}}_1 \cup \bar{\mathcal{B}}_2$, $\mathtt{C}(x) \neq \mathtt{Eval}(K, x)$, we have

$$|\mathcal{B}_1| + |\mathcal{B}_2| = |\bar{\mathcal{B}}_1 \cup \bar{\mathcal{B}}_2| \leq |\{x \in \{0,1\}^n : \mathtt{C}(x) \neq \mathtt{Eval}(K, x)\}| \leq (\frac{1}{6} - \varepsilon) \cdot 2^n$$

□

Similar to Equation (13), we have

$$\frac{|\mathcal{G}|}{|\mathcal{U}|} \geq \frac{1}{2} + 2\varepsilon$$

from Equation (14) and Claim E.17.1.

Therefore, the probability that $(\psi, u_1, u_2, u_3) \xleftarrow{\$} \mathcal{U}$ is in $\mathcal{G}$ is at least $\frac{1}{2} + 2\varepsilon$. □

Claim E.18. *For any $i \in [1, N]$, $\Pr[(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}] \geq \frac{1}{2} + 2\varepsilon$.*

Proof. This comes from Claim E.17 directly as here we have $\psi^{(i)} \xleftarrow{\$} [0, \lfloor \frac{2^\lambda}{3} \rfloor - 1]$, $u_1^{(i)} \xleftarrow{\$} \{0,1\}^{n_1}$, $u_2^{(i)} \xleftarrow{\$} \{0,1\}^{n_2}$, and $u_3^{(i)} \xleftarrow{\$} \{0,1\}^{n_3}$. □

Claim E.19. *For any $i \in [1, N]$, if $(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}$, then*

$$\Pr[msg^{(i)} \neq msg] \leq negl(\lambda)$$

Proof. First, as $u_0^{(i)} = 3\psi^{(i)}$ and $x_2^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$, $x_2^{(i)} = f(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$. Since $(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}$, we have $\mathtt{C}(x_2^{(i)}) = \mathtt{Eval}(K, x_2^{(i)})$.

Also, as $\tilde{u}_0^{(i)} = 3\psi^{(i)} + 1$ and $\tilde{x}_2^{(i)} = (\tilde{u}_0^{(i)}, u_1^{(i)}, u_2^{(i)} \oplus \alpha, u_3^{(i)})$, $\tilde{x}_2^{(i)} = g(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$. Since $(\psi^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}) \in \mathcal{G}$, we have $\mathtt{C}(\tilde{x}_2^{(i)}) = \mathtt{Eval}(K, \tilde{x}_2^{(i)})$.

Now, we have

$$\begin{aligned}
& y_2^{(i)}[m_1 + 1, m] \\
=& \mathtt{C}(u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})[m_1 + 1, m] \\
=& \mathtt{Eval}(K, (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}))[m_1 + 1, m] \\
=& \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)}))
\end{aligned}$$

Also, we have

$$\begin{aligned}
&\tilde{y}_2^{(i)}[m_1+1,m]\\
=&\mathtt{C}(\tilde{u}_0^{(i)},u_1^{(i)},u_2^{(i)}\oplus\alpha,u_3^{(i)})[m_1+1,m]\\
=&\mathtt{Eval}(K,(\tilde{u}_0^{(i)},u_1^{(i)},u_2^{(i)}\oplus\alpha,u_3^{(i)}))[m_1+1,m]\\
=&(\beta\|0^{m_2-\lambda})\oplus\mathsf{F}_{mask}.\mathtt{Eval}(k_{mask},(\tilde{u}_0^{(i)}-1,u_1^{(i)},u_2^{(i)}\oplus\alpha\oplus\alpha,u_3^{(i)}))\\
=&(\beta\|0^{m_2-\lambda})\oplus\mathsf{F}_{mask}.\mathtt{Eval}(k_{mask},(u_0^{(i)},u_1^{(i)},u_2^{(i)},u_3^{(i)}))
\end{aligned}$$

Thus, we have $\mathtt{P}^{(i)}(\alpha)=(\tilde{y}_2^{(i)}\oplus y_2^{(i)})[m_1+1,m_1+\lambda]=\beta$.

Next, as $\boldsymbol{ct}_\alpha$ are valid ciphertexts of α and rek is a randomized evaluation key of ek, $\boldsymbol{ct}_\beta^{(i)}$ are valid ciphertexts of $\mathtt{P}^{(i)}(\alpha)=\beta$. In addition, rpk is generated by a "good" randomness. Thus, by Lemma E.32,

$$\Pr[\mathtt{ExtractIII}(\mathtt{C},\boldsymbol{ct}_\beta^{(i)},\mathit{rpk},\mathit{rek})\neq msg]\leq negl(\lambda)$$

i.e., $\Pr[msg^{(i)}\neq msg]\leq negl(\lambda)$. □

Combining Claim E.18 and Claim E.19, we have

$$\begin{aligned}
&\Pr[msg^{(i)}=msg]\\
\geq&\Pr[msg^{(i)}=msg\wedge(\psi^{(i)},u_1^{(i)},u_2^{(i)},u_3^{(i)})\in\mathcal{G}]\\
=&\Pr[msg^{(i)}=msg\mid(\psi^{(i)},u_1^{(i)},u_2^{(i)},u_3^{(i)})\in\mathcal{G}]\cdot\Pr[(\psi^{(i)},u_1^{(i)},u_2^{(i)},u_3^{(i)})\in\mathcal{G}]\\
\geq&(1-negl(\lambda))\cdot(\frac{1}{2}+2\varepsilon)\\
\geq&\frac{1}{2}+\varepsilon
\end{aligned}$$

Let X_i be a random variable over $\{0,1\}$ that $X_i=1$ iff $msg^{(i)}=msg$. Let $X=\sum_{i=1}^N X_i$ and let $\mu=\sum_{i=1}^N\Pr[msg^{(i)}=msg]$. Note that $\mu\geq\frac{1+2\varepsilon}{2}\cdot N$ and thus we have $(1-\frac{2\varepsilon}{1+2\varepsilon})\cdot\mu\geq\frac{1}{1+2\varepsilon}\cdot\frac{1+2\varepsilon}{2}\cdot N=\frac{N}{2}$. Then by the Chernoff bound, we have

$$\begin{aligned}
&\Pr[X\leq\frac{N}{2}]\\
\leq&\Pr[X\leq(1-\frac{2\varepsilon}{1+2\varepsilon})\cdot\mu]\\
\leq&e^{-\frac{4\varepsilon^2}{2(1+2\varepsilon)^2}\cdot\mu}\\
\leq&e^{-\frac{4\varepsilon^2}{2(1+2\varepsilon)^2}\cdot\frac{1+2\varepsilon}{2}\cdot N}\\
=&e^{-\frac{\varepsilon^2}{1+2\varepsilon}\cdot N}\leq e^{-\frac{\varepsilon^2}{3}\cdot N}=e^{-\frac{\lambda}{3}}
\end{aligned}$$

which is negligible. This implies that with all but negligible probability, the number of $i\in[1,N]$ s.t. $msg=msg^{(i)}$ are over $\frac{N}{2}$, and thus we have $\Pr[msg'\neq msg]\leq negl(\lambda)$. □

Lemma E.34. *Let* $msg' \leftarrow \texttt{ExtractI}(\texttt{C})$*, then we have* $\Pr[msg \neq msg'] \leq negl(\lambda)$*.*

Proof. Let $x_1^{(i)}, y_1^{(i)}, \boldsymbol{ct}_\alpha^{(i)}, \mathit{rpk}^{(i)}, \mathit{rek}^{(i)}, msg^{(i)}$ be variables used in the i-th repetition when running the algorithm $\texttt{ExtractI}(\texttt{C})$. First, we show that if $\texttt{C}(x_1^{(i)}) = \texttt{Eval}(K, x_1^{(i)})$, then we have $msg^{(i)} = msg$ with all but negligible probability.

Claim E.20. *If* $\texttt{C}(x_1^{(i)}) = \texttt{Eval}(K, x_1^{(i)})$*,* $\Pr[msg^{(i)} \neq msg] \leq negl(\lambda)$*.*

Proof. Let $x_1^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, u_3^{(i)})$, and let

$$(r_1^{(i)}, \ldots, r_\lambda^{(i)}, r_{rpk}^{(i)}, r_{rek}^{(i)}) = \mathsf{F}_{IR}.\texttt{Eval}(k_{IR}, (u_0^{(i)}, u_2^{(i)}, u_3^{(i)}), u_1^{(i)})$$

as $\texttt{C}(x_1^{(i)}) = \texttt{Eval}(K, x_1^{(i)})$, we have

$$\forall j \in [1, \lambda], \boldsymbol{ct}_\alpha^{(i)}[j] = \mathsf{SFHE}.\texttt{Enc}(pk, \alpha[j]; r_j^{(i)})$$

$$\mathit{rpk}^{(i)} = \mathsf{SFHE}.\texttt{RandPK}(pk; r_{rpk}^{(i)})$$

$$\mathit{rek}^{(i)} = \mathsf{SFHE}.\texttt{RandEK}(pk, ek; r_{rek}^{(i)})$$

This implies that $\boldsymbol{ct}_\alpha^{(i)}$ is a vector of valid ciphertexts for α and rek is a randomized evaluation key for ek.

In addition, since $u_1^{(i)}$ is sampled uniformly from $\{0,1\}^{n_1}$, the invoker randomization property of F_{IR} ensures that the output of $\mathsf{F}_{IR}.\texttt{Eval}(k_{IR}, (u_0^{(i)}, u_2^{(i)}, u_3^{(i)}), u_1^{(i)})$ will be uniform even if the other part of $x_1^{(i)}$ and the key k_{IR} is fixed and known. Note that in the definition of special FHE (and more precisely Equation (2)), we require that $\Pr[r_{rpk} \xleftarrow{\$} \{0,1\}^{l_{rpk}} : r_{rpk} \in \mathcal{B}] \leq negl(\lambda)$, where $\mathcal{B}$ is the bad randomness subset for $\mathsf{SFHE}.\texttt{RandPK}$. Thus, we have

$$\Pr[r_{rpk}^{(i)} \in \mathcal{B}] \leq negl(\lambda)$$

Therefore, by Lemma E.33 we have

$$\begin{aligned}
&\Pr[\texttt{ExtractII}(\texttt{C}, \boldsymbol{ct}_\alpha^{(i)}, \mathit{rpk}^{(i)}, \mathit{rek}^{(i)}) = msg] \\
\geq &\Pr[\texttt{ExtractII}(\texttt{C}, \boldsymbol{ct}_\alpha^{(i)}, \mathit{rpk}^{(i)}, \mathit{rek}^{(i)}) = msg \wedge r_{rpk}^{(i)} \notin \mathcal{B}] \\
= &\Pr[\texttt{ExtractII}(\texttt{C}, \boldsymbol{ct}_\alpha^{(i)}, \mathit{rpk}^{(i)}, \mathit{rek}^{(i)}) = msg \mid r_{rpk}^{(i)} \notin \mathcal{B}] \cdot \Pr[r_{rpk}^{(i)} \notin \mathcal{B}] \\
\geq &(1 - negl(\lambda)) \cdot (1 - negl(\lambda)) \\
\geq &1 - negl(\lambda)
\end{aligned}$$

Thus, we have $\Pr[msg^{(i)} \neq msg] \leq negl(\lambda)$. □

Next, since $x_1^{(i)} \xleftarrow{\$} \{0,1\}^n$, we have

$$\Pr[\mathtt{C}(x_1^{(i)}) \neq \mathtt{Eval}(K, x_1^{(i)})] = \frac{|\{x \in \{0,1\}^n : \mathtt{C}(x) \neq \mathtt{Eval}(K,x)\}|}{2^n} \leq \frac{1}{6} - \varepsilon$$

Thus, we have

$$\begin{aligned}
&\Pr[msg^{(i)} = msg] \\
\geq &\Pr[msg^{(i)} = msg \wedge \mathtt{C}(x_1^{(i)}) = \mathtt{Eval}(K, x_1^{(i)}))] \\
= &\Pr[msg^{(i)} = msg \mid \mathtt{C}(x_1^{(i)}) = \mathtt{Eval}(K, x_1^{(i)}))] \cdot \Pr[\mathtt{C}(x_1^{(i)}) = \mathtt{Eval}(K, x_1^{(i)}))] \\
\geq &(1 - negl(\lambda)) \cdot (\frac{5}{6} + \varepsilon) \geq \frac{5}{6}
\end{aligned}$$

Let X_i be a random variable over $\{0,1\}$ that $X_i = 1$ iff $msg^{(i)} = msg$. Let $X = \sum_{i=1}^{N} X_i$ and let $\mu = \sum_{i=1}^{N} \Pr[msg^{(i)} = msg]$. Note that $\mu \geq \frac{5}{6} \cdot N$. Then by the Chernoff bound, we have

$$\Pr[X \leq \frac{N}{2}] \leq \Pr[X \leq (1 - \frac{2}{5}) \cdot \mu] \leq e^{-\frac{2}{25} \cdot \mu} \leq e^{-\frac{2}{25} \cdot \frac{5}{6} \cdot N} = e^{-\frac{1}{15} \cdot N} \leq e^{-\frac{\lambda}{15}}$$

which is negligible. This implies that with all but negligible probability, the number of $i \in [1, N]$ s.t. $msg = msg^{(i)}$ are over $\frac{N}{2}$, and thus we have $\Pr[msg' \neq msg] \leq negl(\lambda)$. □

The $(\frac{1}{6} - \varepsilon)$-robust learnability of UOF comes from Lemma E.34 directly.

Black-Box Pseudorandomness. Next, we prove the black-box pseudorandomness of UOF. First, we define the following games between a challenger and a PPT adversary $\mathcal{A}$:

- **Game 0.** In Game 0, the challenger answers $\mathcal{A}$'s oracle queries with $\mathcal{O}_0$.
  More precisely, in the beginning, the adversary submits a message msg to the challenger and the challenger samples $K = (\alpha, \beta, k_{IR}, k_{mask}, pk, sk, ek, msg)$ by running $\mathtt{KeyGen}(1^\lambda, msg)$.
  Then, each time the adversary submits an input $x \in \{0,1\}^n$, the challenger parses $x = (u_0, u_1, u_2, u_3) \in \{0,1\}^{n_0} \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \{0,1\}^{n_3}$ and proceeds as follows:
  1. $w = (u_0, u_2, u_3)$.
  2. $(r_1, \ldots, r_\lambda, r_{rpk}, r_{rek}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, w, u_1)$.
  3. For $i \in [1, \lambda]$:
     (a) $ct_i = \mathsf{SFHE}.\mathtt{Enc}(pk, \alpha[i]; r_i)$.
  4. $rp\mathcal{k} = \mathsf{SFHE}.\mathtt{RandPK}(pk; r_{rpk})$.
  5. $re\mathcal{k} = \mathsf{SFHE}.\mathtt{RandEK}(pk, ek; r_{rek})$.
  6. $y_{ct} = ct_1 \| ct_2 \| \ldots \| ct_\lambda \| rp\mathcal{k} \| re\mathcal{k}$
  7. $d = u_0 \mod 3$, where $d \in \{0,1,2\}$.
  8. If $d = 0$:
     (a) $z = (u_0, u_1, u_2, u_3)$.
     (b) $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.

(c) **Return** $y_{ct} \| y_{mask}$.

9. If $d = 1$:
   (a) $u'_0 = u_0 - 1$.
   (b) $u'_2 = u_2 \oplus \alpha$.
   (c) $z = (u'_0, u_1, u'_2, u_3)$.
   (d) $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
   (e) **Return** $y_{ct} \| ((\beta \| 0^{m_2 - \lambda}) \oplus y_{mask})$.
10. If $d = 2$:
   (a) Parse $u_3 = (\mathit{tct}_i)_{i \in [1,\lambda]} \in (\{0,1\}^{L_t})^{\lambda}$.
   (b) For $i \in [1, \lambda]$:
      i. $\mu_i = \mathsf{SFHE}.\mathtt{TCTDec}(sk, \mathit{tct}_i)$.
   (c) $\nu = \mu_1 \| \ldots \| \mu_\lambda$.
   (d) $u'_2 = u_2 \oplus \nu \oplus \beta$.
   (e) $u'_0 = u_0 - 2$.
   (f) $z = (u'_0, u_1, u'_2, u_3)$.
   (g) $y_{mask} = \mathsf{F}_{mask}.\mathtt{Eval}(k_{mask}, z)$.
   (h) **Return** $y_{ct} \| (msg \oplus y_{mask})$.

Finally, the game outputs 1 iff the adversary outputs 1.
Here, w.l.o.g., we assume that all inputs submitted by the adversary are *distinct*.

- **Game 1.** This is identical to Game 0 except that for each query, the challenger samples $(r_1, \ldots, r_\lambda, r_{rpk}, r_{rek}) \xleftarrow{\$} \{0,1\}^{n_1}$ instead of evaluating it as $(r_1, \ldots, r_\lambda, r_{rpk}, r_{rek}) = \mathsf{F}_{IR}.\mathtt{Eval}(k_{IR}, (u_0, u_2, u_3), u_1)$.
- **Game 2.** This is identical to Game 1 except that the challenger uses a random function f instead of the pseudorandom function F_{mask} when answering $\mathcal{A}$'s oracle queries.
- **Game 3.** This is identical to Game 2 except that the challenger maintains a list $\mathcal{L}$. Here, $\mathcal{L}$ is initialized as an empty list in the beginning. Then on receiving a query $x = (u_0, u_1, u_2, u_3)$, the challenger puts (x, z) to the list where z is the input to f. Recall that, z is computed as follows in each case:
  - If $u_0 = 0 \mod 3$:
    1. $z = (u_0, u_1, u_2, u_3)$.
  - If $u_0 = 1 \mod 3$:
    1. $u'_0 = u_0 - 1$.
    2. $u'_2 = u_2 \oplus \alpha$.
    3. $z = (u'_0, u_1, u'_2, u_3)$.
  - If $u_0 = 2 \mod 3$:
    1. Parse $u_3 = (\mathit{tct}_i)_{i \in [1,\lambda]} \in (\{0,1\}^{L_t})^{\lambda}$.
    2. For $i \in [1, \lambda]$:
       (a) $\mu_i = \mathsf{SFHE}.\mathtt{TCTDec}(sk, \mathit{tct}_i)$.
    3. $\nu = \mu_1 \| \ldots \| \mu_\lambda$.
    4. $u'_2 = u_2 \oplus \nu \oplus \beta$.
    5. $u'_0 = u_0 - 2$.
    6. $z = (u'_0, u_1, u'_2, u_3)$.
- **Game 4.** This is identical to Game 3 except that for each query, the challenger samples the second part of the output from $\{0,1\}^{m_2}$. In more detail, the challenger proceeds as follows on receiving a query x:

1. For $i \in [1, \lambda]$:
   (a) $ct_i \leftarrow \mathsf{SFHE}.\mathtt{Enc}(pk, \alpha[i])$.
2. $rp\mathcal{K} \leftarrow \mathsf{SFHE}.\mathtt{RandPK}(pk)$.
3. $re\mathcal{K} \leftarrow \mathsf{SFHE}.\mathtt{RandEK}(pk, ek)$.
4. $y_{ct} = ct_1 \| ct_2 \| \ldots \| ct_\lambda \| rp\mathcal{K} \| re\mathcal{K}$.
5. $y_{mask} \xleftarrow{\$} \{0,1\}^{m_2}$.
6. **Return** $y_{ct} \| y_{mask}$.

Note that in Game 4, the challenger does not need to maintain the list $\mathcal{L}$ and $\mathcal{L}$ is only used in arguing the indistinguishability between Game 3 and Game 4.

- **Game 5.** This is identical to Game 4 except that for each query, the challenger samples $y_{ct} \xleftarrow{\$} \{0,1\}^{m_1}$.

It is easy to see that in Game 5, the challenger answers the adversary's queries with $\mathcal{O}_1$. Let $\mathcal{E}_i$ be the output of Game i, then it is sufficient to show that $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$. Next, we argue this via proving the following lemmas.

Lemma E.35. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. First, by pseudorandomness of F_{IR}, we can use a random function F to replace F_{IR} in answering $\mathcal{A}$'s queries. Next, for each query, the input to F, which is $u_0 \| u_2 \| u_3 \| u_1$, will be distinct since the inputs x are distinct and the map from x to $u_0 \| u_2 \| u_3 \| u_1$ is bijective. Thus, it is safe to replace output of F with a fresh random string. □

Lemma E.36. $|\Pr[\mathcal{E}_1 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 1 and Game 2 comes from pseudorandomness of F_{mask} directly. □

Lemma E.37. $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| = 0$.

Proof. The adversary's view in Game 2 and Game 3 are identical, thus the probabilities that the two games output 1 are also identical. □

Lemma E.38. $|\Pr[\mathcal{E}_3 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. In Game 3, the second part of the oracle's output is $f(z) \oplus S$, where f is a random function and $S \in \{0^{m_2}, \beta \| 0^{m_2 - \lambda}, msg\}$; and in Game 4, the second part of the oracle's output is a random string in $\{0,1\}^{m_2}$. Note that Game 3 and Game 4 are identical if all z are distinct.

To prove that z are not likely to repeat, we first define

$$\mathcal{L}_\sigma = \{(x = (u_0, u_1, u_2, u_3), z) \in \mathcal{L} : u_0 = \sigma \mod 3\}$$

for $\sigma \in \{0, 1, 2\}$. Then we define the following events in Game 3:

- col_0: There exists (x, z) and (x', z') in $\mathcal{L}_0$ s.t. $x \neq x'$ and $z = z'$.

- col_1: There exists (x, z) and (x', z') in $\mathcal{L}_1$ s.t. $x \neq x'$ and $z = z'$.
- col_2: There exists (x, z) and (x', z') in $\mathcal{L}_2$ s.t. $x \neq x'$ and $z = z'$.
- col_{01}: There exists $(x, z) \in \mathcal{L}_0$ and $(x', z') \in \mathcal{L}_1$ s.t. $x \neq x'$ and $z = z'$.
- col_{02}: There exists $(x, z) \in \mathcal{L}_0$ and $(x', z') \in \mathcal{L}_2$ s.t. $x \neq x'$ and $z = z'$.
- col_{12}: There exists $(x, z) \in \mathcal{L}_1$ and $(x', z') \in \mathcal{L}_2$ s.t. $x \neq x'$ and $z = z'$.

It is easy to see if none of the 6 events occurs, then all z are distinct. Thus, it is sufficient to show that the probability that at least one of the 6 event occurs is negligible.

Claim E.21. $\Pr[\mathit{col}_0] = 0$

Proof. As for any $(x, z) \in \mathcal{L}_0$, $z = x$, there does not exist (x, z) and (x', z') in $\mathcal{L}_0$ s.t. $x \neq x'$ and $z = z'$. □

Claim E.22. $\Pr[\mathit{col}_1] = 0$

Proof. Assuming col_1 occurs, i.e., there exists (x, z) and (x', z') in $\mathcal{L}_1$ s.t. $x \neq x'$ and $z = z'$. Let $x = (u_0, u_1, u_2, u_3)$ and $x' = (u'_0, u'_1, u'_2, u'_3)$. Then we have $z = (u_0 - 1, u_1, u_2 \oplus \alpha, u_3)$ and $z' = (u'_0 - 1, u'_1, u'_2 \oplus \alpha, u'_3)$. Since $z = z'$, we have $u_0 - 1 = u'_0 - 1$ (i.e., $u_0 = u'_0$), $u_1 = u'_1$, $u_2 \oplus \alpha = u'_2 \oplus \alpha$ (i.e., $u_2 = u'_2$), and $u_3 = u'_3$. This contradicts the assumption that $x \neq x'$, thus col_1 will not occur. □

Claim E.23. $\Pr[\mathit{col}_2] = 0$

Proof. Assuming col_2 occurs, i.e., there exists (x, z) and (x', z') in $\mathcal{L}_2$ s.t. $x \neq x'$ and $z = z'$. Let $x = (u_0, u_1, u_2, u_3)$ and $x' = (u'_0, u'_1, u'_2, u'_3)$. Then we have $z = (u_0 - 2, u_1, u_2 \oplus \nu \oplus \beta, u_3)$ and $z' = (u'_0 - 2, u'_1, u'_2 \oplus \nu' \oplus \beta, u'_3)$ where ν, ν' are decryption of u_3 and u'_3 respectively. Since $z = z'$, we have $u_0 - 2 = u'_0 - 2$ (i.e., $u_0 = u'_0$), $u_1 = u'_1$, and $u_3 = u'_3$, which implies that $\nu = \nu'$. Thus, we also have $u_2 = u'_2$. This contradicts the assumption that $x \neq x'$, thus col_2 will not occur. □

Before arguing that the remaining 3 events also occur with a negligible probability, we first define a set of auxiliary adversaries $\mathcal{B}_\ell$, which is identical to $\mathcal{A}$ except that it will artificially abort the game (and outputs 1) after submitting ℓ queries.[29] We also define $\mathit{col}_{\sigma,\ell}$ for $\sigma \in \{0, 1, 2, 01, 02, 12\}$ to be the event that col_σ occurs when the challenger interacts with $\mathcal{B}_\ell$ (instead of $\mathcal{A}$) in Game 3. Assume that the PPT adversary $\mathcal{A}$ makes at most Q queries to the challenger, where Q is polynomial in λ, then $\mathcal{B}_{Q+1}$ is identical to $\mathcal{A}$. Thus, it is sufficient to prove that $\mathit{col}_{\sigma,Q+1}$ occurs with negligible probability for $\sigma \in \{01, 02, 12\}$.

To prove this, we first prove the following claim.

[29] If $\mathcal{A}$ stops the game with an output before triggering $\mathcal{B}_\ell$'s abort condition, then $\mathcal{B}_\ell$ will also stop and outputs what $\mathcal{A}$ outputs.

Claim E.24. *For $\ell \in [1, Q]$, if*

$$\Pr[\mathit{col}_{0,\ell} \vee \mathit{col}_{1,\ell} \vee \mathit{col}_{2,\ell} \vee \mathit{col}_{01,\ell} \vee \mathit{col}_{02,\ell} \vee \mathit{col}_{12,\ell}] \leq negl(\lambda)$$

then we have

$$\Pr[\mathit{col}_{02,\ell+1} \vee \mathit{col}_{12,\ell+1} \vee \mathit{col}_{01,\ell+1}] \leq negl(\lambda)$$

Proof. We first define the following auxiliary games:

- H_0: The challenger interacts with $\mathcal{B}_{\ell+1}$ and proceeds identically as it does in Game 3, except that after $\mathcal{B}_{\ell+1}$ stops or after it submits the $(\ell+1)$-th query and aborts (i.e., the challenger does not need to respond the $(\ell+1)$-th query from $\mathcal{B}_{\ell+1}$), the challenger **outputs 1** iff *at least one* of the three events $\mathit{col}_{02,\ell+1}$, $\mathit{col}_{12,\ell+1}$, and $\mathit{col}_{01,\ell+1}$ occurs.
- H_1: This is identical to Game H_0 except that for each query, the challenger samples the second part of the output from $\{0,1\}^{m_2}$. More precisely, the challenger proceeds as follows on receiving a query x:
  1. For $i \in [1, \lambda]$:
     (a) $ct_i \leftarrow \mathsf{SFHE.Enc}(pk, \alpha[i])$.
  2. $\mathit{rpk} \leftarrow \mathsf{SFHE.RandPK}(pk)$.
  3. $\mathit{rek} \leftarrow \mathsf{SFHE.RandEK}(pk, ek)$.
  4. $y_{ct} = ct_1 \| ct_2 \| \ldots \| ct_\lambda \| \mathit{rpk} \| \mathit{rek}$.
  5. $y_{mask} \xleftarrow{\$} \{0,1\}^{m_2}$.
  6. **Return** $y_{ct} \| y_{mask}$.
  7. If $u_0 = 0 \mod 3$:
     (a) $z = (u_0, u_1, u_2, u_3)$.
  8. If $u_0 = 1 \mod 3$:
     (a) $u'_0 = u_0 - 1$.
     (b) $u'_2 = u_2 \oplus \alpha$.
     (c) $z = (u'_0, u_1, u'_2, u_3)$.
  9. If $u_0 = 2 \mod 3$:
     (a) Parse $u_3 = (\mathit{tct}_i)_{i \in [1,\lambda]} \in (\{0,1\}^{L_t})^\lambda$.
     (b) For $i \in [1, \lambda]$:
        i. $\mu_i = \mathsf{SFHE.TCTDec}(sk, \mathit{tct}_i)$.
     (c) $\nu = \mu_1 \| \ldots \| \mu_\lambda$.
     (d) $u'_2 = u_2 \oplus \nu \oplus \beta$.
     (e) $u'_0 = u_0 - 2$.
     (f) $z = (u'_0, u_1, u'_2, u_3)$.
  10. Put (x, z) to $\mathcal{L}$.
- H_2: This is identical to Game H_1 except that the challenger does not check if $\mathit{col}_{02,\ell+1}$ or $\mathit{col}_{12,\ell+1}$ occurs. In particular, after $\mathcal{B}_{\ell+1}$ stops or aborts, the challenger outputs 1 iff $\mathit{col}_{01,\ell+1}$ occurs.
  Note that in Game H_2, the challenger does not need to compute and store z for an input $x = (u_0, u_1, u_2, u_3)$ if $u_0 = 2 \mod 3$.
- H_3: This is identical to Game H_2 except that for each query, the challenger returns a random string.

Let $\mathcal{F}_i$ be the output of Game H_i. It is easy to see $\Pr[\mathit{col}_{02,\ell+1} \vee \mathit{col}_{12,\ell+1} \vee \mathit{col}_{01,\ell+1}] = \Pr[\mathcal{F}_0 = 1]$. Thus, it is sufficient to prove that $\Pr[\mathcal{F}_0 = 1] \leq negl(\lambda)$. This can be guaranteed by the following claims:

Claim E.24.1. $|\Pr[\mathcal{F}_0 = 1] - \Pr[\mathcal{F}_1 = 1]| \leq negl(\lambda)$.

Proof. First, as the challenger only needs to answer the first ℓ oracle queries from the adversary $\mathcal{B}_{\ell+1}$, Game H_0 and Game H_1 are identical if all z appeared in the first ℓ queries are distinct. This will occur with all but negligible probability since $\Pr[\mathit{col}_{0,i} \vee \mathit{col}_{1,i} \vee \mathit{col}_{2,i} \vee \mathit{col}_{01,i} \vee \mathit{col}_{02,i} \vee \mathit{col}_{12,i}] \leq negl(\lambda)$. □

Claim E.24.2. $|\Pr[\mathcal{F}_1 = 1] - \Pr[\mathcal{F}_2 = 1]| \leq negl(\lambda)$.

Proof. Game H_1 and Game H_2 are identical unless $\mathit{col}_{02,\ell+1}$ or $\mathit{col}_{12,\ell+1}$ occurs.

First, as the challenger does not need β when answering oracle queries from the adversary. It can postpone the selection of β *after* the adversary stops or aborts.[30] Therefore, all queries submitted by the adversary will be independent of β.

For any $(x, z) \in \mathcal{L}_2$ and $(x', z') \in \mathcal{L}_0$. Let $x = (u_0, u_1, u_2, u_3)$ and $x' = (u'_0, u'_1, u'_2, u'_3)$. Then we have $z = (u_0 - 2, u_1, u_2 \oplus \nu \oplus \beta, u_3)$ and $z' = (u'_0, u'_1, u'_2, u'_3)$, where ν is determined by u_3. If $z = z'$, then we have $u_2 \oplus \nu \oplus \beta = u'_2$, which occurs with a negligible probability since β is sampled uniformly at random from $\{0,1\}^\lambda$ after $\mathcal{B}_{\ell+1}$ submits the queries.

For any $(x, z) \in \mathcal{L}_2$ and $(x', z') \in \mathcal{L}_1$. Let $x = (u_0, u_1, u_2, u_3)$ and $x' = (u'_0, u'_1, u'_2, u'_3)$. Then we have $z = (u_0 - 2, u_1, u_2 \oplus \nu \oplus \beta, u_3)$ and $z' = (u'_0 - 1, u'_1, u'_2 \oplus \alpha, u'_3)$, where ν is determined by u_3. If $z = z'$, then we have $u_2 \oplus \nu \oplus \beta = u'_2 \oplus \alpha$, which occurs with a negligible probability since β is sampled uniformly at random from $\{0,1\}^\lambda$ after $\mathcal{B}_{\ell+1}$ submits the queries. This completes proof of this Claim. □

Claim E.24.3. $|\Pr[\mathcal{F}_2 = 1] - \Pr[\mathcal{F}_3 = 1]| \leq negl(\lambda)$.

Proof. In Game H_2, the challenger does not need the secret key of SFHE when answering oracle queries. Also, as the challenger only computes z for an input $x = (u_0, u_1, u_2, u_3)$ s.t. $u_0 = 0 \mod 3$ or $u_0 = 1 \mod 3$, it also does not need the secret key of SFHE when determining if it should output 1. Thus, indistinguishability between Game H_2 and Game H_3 comes from the ciphertext and key pseudorandomness property of SFHE by a direct reduction. □

Claim E.24.4. $\Pr[\mathcal{F}_3 = 1] \leq negl(\lambda)$.

Proof. In Game H_3, the challenger will return random strings for all oracle queries. Thus, it does not need α when answering oracle queries and can postpone the selection of α *after* the adversary stops or aborts.[31] Therefore, all queries submitted by the adversary will be independent of α.

For any $(x, z) \in \mathcal{L}_1$ and $(x', z') \in \mathcal{L}_0$. Let $x = (u_0, u_1, u_2, u_3)$ and $x' = (u'_0, u'_1, u'_2, u'_3)$. Then we have $z = (u_0 - 1, u_1, u_2 \oplus \alpha, u_3)$ and $z' = (u'_0, u'_1, u'_2, u'_3)$. If

[30] It can also postpone the computation of z after that.

[31] It can also postpone the computation of z after that.

$z = z'$, then we have $u_2 \oplus \alpha = u'_2$, which occurs with a negligible probability since α is sampled uniformly at random from $\{0,1\}^\lambda$ after $\mathcal{B}_{\ell+1}$ submits the queries.

This completes proof of this Claim. □

□

We continue to show that $\Pr[\mathit{col}_{01,Q+1} \vee \mathit{col}_{02,Q+1} \vee \mathit{col}_{12,Q+1}] \leq negl(\lambda)$. First, from Claim E.21 to Claim E.23, we have $\Pr[\mathit{col}_0] = \Pr[\mathit{col}_1] = \Pr[\mathit{col}_2] = 0$, thus we have $\Pr[\mathit{col}_{0,i}] = \Pr[\mathit{col}_{1,i}] = \Pr[\mathit{col}_{2,i}] = 0$ for all $i \in [1, Q]$. In addition, $\Pr[\mathit{col}_{01,1}] = \Pr[\mathit{col}_{02,1}] = \Pr[\mathit{col}_{12,1}] = 0$ since only one query is made and there cannot be collision. Therefore, we have

$$\Pr[\mathit{col}_{0,1} \vee \mathit{col}_{1,1} \vee \mathit{col}_{2,1} \vee \mathit{col}_{01,1} \vee \mathit{col}_{02,1} \vee \mathit{col}_{12,1}] = 0$$

Then, by Claim E.24, we have

$$\Pr[\mathit{col}_{02,2} \vee \mathit{col}_{12,2} \vee \mathit{col}_{01,2}] \leq negl(\lambda)$$

Now we have

$$\Pr[\mathit{col}_{0,2} \vee \mathit{col}_{1,2} \vee \mathit{col}_{2,2} \vee \mathit{col}_{01,2} \vee \mathit{col}_{02,2} \vee \mathit{col}_{12,2}] \leq negl(\lambda)$$

and repeating this for Q times, we have

$$\Pr[\mathit{col}_{01,Q+1} \vee \mathit{col}_{02,Q+1} \vee \mathit{col}_{12,Q+1}] \leq negl(\lambda) \tag{15}$$

Since $\mathcal{B}_{Q+1}$ is identical to $\mathcal{A}$, Equation (15) implies that

$$\Pr[\mathit{col}_{01} \vee \mathit{col}_{02} \vee \mathit{col}_{12}] \leq negl(\lambda)$$

Then combing with Claim E.21 to Claim E.23, we have

$$\Pr[\mathit{col}_0 \vee \mathit{col}_1 \vee \mathit{col}_2 \vee \mathit{col}_{01} \vee \mathit{col}_{02} \vee \mathit{col}_{12}] \leq negl(\lambda)$$

That is, the inputs z to f will be distinct with all but negligible probability, and it is safe to replace outputs of f with random strings. This completes proof of Lemma E.38. □

Lemma E.39. $|\Pr[\mathcal{E}_4 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 4 and Game 5 comes from the ciphertext and key pseudorandomness property of SFHE by a direct reduction. □

Combining Lemma E.35 to Lemma E.39, we have $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$, i.e., the probability that $\mathcal{A}$ outputs 1 when interacting with $\mathcal{O}_0$ and the probability that it outputs 1 when interacting with $\mathcal{O}_1$ does not have a non-negligible difference. This completes the proof of black-box pseudorandomness.

E.7 Security Analysis of the General Construction of Public-Key Watermarkable PRFs

We present proof of Theorem 7.1 in this section. More precisely, we will prove the functionality preserving property, extraction correctness, watermarking meaningfulness, pseudorandomness for unmarked keys, pseudorandomness for marked keys, and Q-bounded ϵ-unremovability of WPRF.

Functionality Preserving. For any message $msg \in \{0,1\}^\kappa$, let $pp_{hw} \leftarrow \mathsf{HWF}.\mathtt{Setup}(1^\lambda)$, $pp_{uo} \leftarrow \mathsf{UOF}.\mathtt{Setup}(1^\lambda)$, and $PP = (pp_{hw}, pp_{uo})$. Also, let $(k_{hw}, \mathtt{hint}) \leftarrow \mathsf{HWF}.\mathtt{KeyGen}(pp_{hw})$, $k_f \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$, $k_{uo} \leftarrow \mathsf{UOF}.\mathtt{KeyGen}(pp_{uo}, \mathtt{hint} \| k_f)$, and $K = (k_{hw}, k_f, k_{uo})$. Let $\mathtt{C}_{hw} \leftarrow \mathsf{HWF}.\mathtt{Mark}(pp_{hw}, k_{hw}, msg)$ and $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}(x) = (\mathtt{C}_{hw}(x) \oplus \mathsf{F}.\mathtt{Eval}(k_f, x), \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k_{uo}, x))$.

As for any $x \in \{0,1\}^n$, $\mathtt{C}(x) \neq \mathtt{Eval}(PP, K, x)$ iff $\mathtt{C}_{hw}(x) \neq \mathsf{HWF}.\mathtt{Eval}(pp_{hw}, k_{hw}, x)$, the functionality preserving property of WPRF comes from the functionality preserving property of HWF directly.

Extraction Correctness. For any message $msg \in \{0,1\}^\kappa$, let $pp_{hw} \leftarrow \mathsf{HWF}.\mathtt{Setup}(1^\lambda)$, $pp_{uo} \leftarrow \mathsf{UOF}.\mathtt{Setup}(1^\lambda)$, and $PP = (pp_{hw}, pp_{uo})$. Also, let $(k_{hw}, \mathtt{hint}) \leftarrow \mathsf{HWF}.\mathtt{KeyGen}(pp_{hw})$, $k_f \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$, $k_{uo} \leftarrow \mathsf{UOF}.\mathtt{KeyGen}(pp_{uo}, \mathtt{hint} \| k_f)$, and $K = (k_{hw}, k_f, k_{uo})$. Let $\mathtt{C}_{hw} \leftarrow \mathsf{HWF}.\mathtt{Mark}(pp_{hw}, k_{hw}, msg)$ and $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}(x) = (\mathtt{C}_{hw}(x) \oplus \mathsf{F}.\mathtt{Eval}(k_f, x), \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k_{uo}, x))$.

Now, assume the extraction algorithm takes as input the watermarked circuit $\mathtt{C}$, and let $\mathtt{C}'_{uo}$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}'_{uo}(x) = \mathtt{C}(x)[m_1 + 1 : m]$, $(\mathtt{hint}', k'_f) \leftarrow \mathsf{UOF}.\mathtt{Extract}(pp_{uo}, \mathtt{C}'_{uo})$, $\mathtt{C}'_{hw}$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}'_{hw}(x) = \mathtt{C}(x)[1 : m_1] \oplus \mathsf{F}.\mathtt{Eval}(k'_f, x)$, and $msg' \leftarrow \mathsf{HWF}.\mathtt{Extract}(pp_{hw}, \mathtt{C}'_{hw}, \mathtt{hint}')$. First, by the correctness of UOF, we have $\mathtt{hint}' = \mathtt{hint}$ and $k'_f = k_f$. Thus, for any $x \in \{0,1\}^n$, we have $\mathtt{C}'_{hw}(x) = \mathtt{C}_{hw}(x)$. Then, by the extraction correctness of HWF, we have $msg' = msg$ with all but negligible probability.

On the other hand, assume the extraction algorithm takes as input the circuit $\mathtt{Eval}(PP, K, \cdot)$, and let $\mathtt{C}'_{uo}$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}'_{uo}(x) = \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k_{uo}, x)$, $(\mathtt{hint}', k'_f) \leftarrow \mathsf{UOF}.\mathtt{Extract}(pp_{uo}, \mathtt{C}'_{uo})$, $\mathtt{C}'_{hw}$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}'_{hw}(x) = \mathsf{HWF}.\mathtt{Eval}(pp_{hw}, k_{hw}, x) \oplus \mathsf{F}.\mathtt{Eval}(k_f, x) \oplus \mathsf{F}.\mathtt{Eval}(k'_f, x)$, and $msg' \leftarrow \mathsf{HWF}.\mathtt{Extract}(pp_{hw}, \mathtt{C}'_{hw}, \mathtt{hint}')$. First, by the correctness of UOF, we have $\mathtt{hint}' = \mathtt{hint}$ and $k'_f = k_f$. Thus, for any $x \in \{0,1\}^n$, we have $\mathtt{C}'_{hw}(x) = \mathsf{HWF}.\mathtt{Eval}(pp_{hw}, k_{hw}, x)$. Then, by the extraction correctness of HWF, we have $msg' = \perp$ with all but negligible probability.

This completes proof of extraction correctness.

Watermarking Meaningfulness. For any fixed circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$, and for any fixed $pp_{uo} \leftarrow \mathsf{UOF}.\mathtt{Setup}(1^\lambda)$, let $\mathtt{C}_{uo}$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}_{uo}(x) = \mathtt{C}(x)[m_1 + 1 : m]$, $(\mathtt{hint}, k_f) \leftarrow \mathsf{UOF}.\mathtt{Extract}(pp_{uo}, \mathtt{C}_{uo})$, and $\mathtt{C}_{hw}$ be a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}_{hw}(x) = \mathtt{C}(x)[1 : m_1] \oplus \mathsf{F}.\mathtt{Eval}(k_f, x)$.

Let $pp_{hw} \leftarrow \mathsf{HWF}.\mathtt{Setup}(1^\lambda)$, then by the watermarking meaningfulness of HWF, we have $\Pr[\mathsf{HWF}.\mathtt{Extract}(pp_{hw}, \mathtt{C}_{hw}, \mathtt{hint}) \neq \perp] \leq negl(\lambda)$, and the watermarking meaningfulness of WPRF follows.

Pseudorandomness for Marked/Unmarked Keys. The pseudorandomness properties of WPRF comes from the pseudorandomness of UOF and the pseudorandomness of F. Next, we only give a detailed proof for the pseudorandomness for marked keys, and the pseudorandomness for unmarked keys can be proved in a similar way.

First, we define the following games between a challenger and a PPT adversary $\mathcal{A}$:

- **Game 0.** In Game 0, the challenger answers $\mathcal{A}$'s oracle queries with $\mathcal{O}_0$.
  More precisely, the challenger first samples $PP = (pp_{hw}, pp_{uo}) \leftarrow \mathtt{Setup}(1^\lambda)$.
  It also samples $(k_{hw}, \mathtt{hint}) \leftarrow \mathsf{HWF}.\mathtt{KeyGen}(pp_{hw})$, $k_f \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$, $k_{uo} \leftarrow \mathsf{UOF}.\mathtt{KeyGen}(pp_{uo}, \mathtt{hint} \| k_f)$, and sets $K = (k_{hw}, k_f, k_{uo})$.
  Then it sends PP to the adversary and receives a message $msg \in \{0,1\}^\kappa$.
  Next, it computes $\mathsf{C}_{hw} \leftarrow \mathsf{HWF}.\mathtt{Mark}(pp_{hw}, k_{hw}, msg)$ and answers the adversary's oracle queries. In particular, each time the adversary submits an input $x \in \{0,1\}^n$, the challenger computes:
  1. $y_{hw} = \mathsf{C}_{hw}(x)$.
  2. $y_f = \mathsf{F}.\mathtt{Eval}(k_f, x)$.
  3. $y_{uo} = \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k_{uo}, x)$.

  and returns $y = (y_{hw} \oplus y_f, y_{uo})$.
  Finally, the game outputs 1 iff the adversary outputs 1.
  Here, w.l.o.g., we assume that all inputs submitted by the adversary are distinct.
- **Game 1.** This is identical to Game 0 except that for each oracle query, the challenger samples y_{uo} uniformly at random instead of evaluating it as $y_{uo} = \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k_{uo}, x)$.
- **Game 2.** This is identical to Game 1 except that for each oracle query, the challenger samples y_f uniformly at random instead of evaluating it as $y_f = \mathsf{F}.\mathtt{Eval}(k_f, x)$.

It is easy to see that in Game 2, the challenger returns a random string in $\{0,1\}^m$ for each oracle query, thus it actually answers the adversary's queries with $\mathcal{O}_1$. Let $\mathcal{E}_i$ be the output of Game i, then it is sufficient to show that $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$. Next, we argue this via proving the following lemmas.

Lemma E.40. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 0 and Game 1 comes from black-box pseudorandomness of UOF directly. □

Lemma E.41. $|\Pr[\mathcal{E}_1 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 1 and Game 2 comes from pseudorandomness of F directly. Note that as in Game 1, the PRF key k_{uo} of UOF is not used, the challenger does not need to generate it. Thus, it can answer $\mathcal{A}$'s oracle queries given only oracle access to $\mathsf{F}.\mathtt{Eval}(k_f, \cdot)$. □

Combining Lemma E.40 and Lemma E.41, we have $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$, i.e., the probability that A outputs 1 when interacting with $\mathcal{O}_0$ and the probability that it outputs 1 when interacting with $\mathcal{O}_1$ does not have a non-negligible difference. This completes the proof of pseudorandomness.

Unremovability. Finally, we prove the unremovability of WPRF. First, we define the following games between a challenger and a PPT ϵ-unremoving-admissible adversary $\mathcal{A}$:

- **Game 0.** This is the real experiment $\texttt{ExptUR}$. More precisely, the challenger proceeds as follows.
  - **I.** First, the challenger samples $pp_{hw} \leftarrow \mathsf{HWF}.\texttt{Setup}(1^\lambda)$ and $pp_{uo} \leftarrow \mathsf{UOF}.\texttt{Setup}(1^\lambda)$. It also samples $(k^*_{hw}, \texttt{hint}^*) \leftarrow \mathsf{HWF}.\texttt{KeyGen}(pp_{hw})$, $k^*_f \leftarrow \mathsf{F}.\texttt{KeyGen}(1^\lambda)$, and $k^*_{uo} \leftarrow \mathsf{UOF}.\texttt{KeyGen}(pp_{uo}, \texttt{hint}^* \| k^*_f)$.
  - **II.** Next, the challenger sends $PP = (pp_{hw}, pp_{uo})$ to the adversary and answers the adversary's challenge oracle queries (for at most Q times) as follow:
    - On input a message $msg \in \{0,1\}^\kappa$, the challenger computes $\mathtt{C}_{hw} \leftarrow \mathsf{HWF}.\texttt{Mark}(pp_{hw}, k^*_{hw}, msg)$ and returns a circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ s.t. for any $x \in \{0,1\}^n$:

$$\mathtt{C}(x) = (\mathtt{C}_{hw}(x) \oplus \mathsf{F}.\texttt{Eval}(k^*_f, x), \mathsf{UOF}.\texttt{Eval}(pp_{uo}, k^*_{uo}, x))$$

    Let $\mathcal{Q}^*$ to be the set of all messages submitted to the challenge oracle and let $\mathcal{R}^*$ to be the set of all circuits returned by the challenge oracle.
  - **III.** Finally, after $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$, the challenger proceeds as follow:
    1. Set $\tilde{\mathtt{C}}_{uo}$ as a circuit that for any $x \in \{0,1\}^n$, $\tilde{\mathtt{C}}_{uo}(x) = \tilde{\mathtt{C}}(x)[m_1 + 1 : m]$.
    2. $(\texttt{hint}, k_f) \leftarrow \mathsf{UOF}.\texttt{Extract}(pp_{uo}, \tilde{\mathtt{C}}_{uo})$.
    3. If $(\texttt{hint}, k_f) = \perp$: **output** 1.
    4. Set $\tilde{\mathtt{C}}_{hw}$ as a circuit that for any $x \in \{0,1\}^n$, $\tilde{\mathtt{C}}_{hw}(x) = \tilde{\mathtt{C}}(x)[1 : m_1] \oplus \mathsf{F}.\texttt{Eval}(k_f, x)$.
    5. $msg \leftarrow \mathsf{HWF}.\texttt{Extract}(pp_{hw}, \tilde{\mathtt{C}}_{hw}, \texttt{hint})$.
    6. **Output** 1 if $msg \notin \mathcal{Q}^*$ and **output** 0 otherwise.
- **Game 1.** This is identical to Game 0 except that in Phased III, after obtaining $(\texttt{hint}, k_f)$, the challenger aborts and outputs 2 if $(\texttt{hint}, k_f) \neq (\texttt{hint}^*, k^*_f)$.

Let $\mathcal{E}_i$ be the output of Game i and we next show that $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$ via proving the following lemmas.

Lemma E.42. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. Game 0 and Game 1 are identical unless $(\texttt{hint}, k_f) \neq (\texttt{hint}^*, k^*_f)$, thus, it is sufficient to show that the inequality occurs with a negligible probability. This comes from the ϵ_2-robust learnability of UOF. More precisely, assuming $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]|$ is non-negligible (i.e., the probability that $(\texttt{hint}, k_f) \neq (\texttt{hint}^*, k^*_f)$ is non-negligible), then we can construct an adversary $\mathcal{B}$ that breaks the ϵ_2-robust learnability of UOF as follows:

- On receiving the public parameter pp_{uo} from its challenger, the adversary $\mathcal{B}$ samples $pp_{hw} \leftarrow \mathsf{HWF}.\mathtt{Setup}(1^\lambda)$, $(k^*_{hw}, \mathtt{hint}^*) \leftarrow \mathsf{HWF}.\mathtt{KeyGen}(pp_{hw})$, and $k^*_f \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then it submits $\mathtt{hint}^* \| k^*_f$ as its challenge message to its challenger and receives k^*_{uo}.
- Next, $\mathcal{B}$ sends $PP = (pp_{hw}, pp_{uo})$ to the adversary $\mathcal{A}$ and answers $\mathcal{A}$'s challenge oracle queries for at most Q times. More precisely, each time the adversary $\mathcal{A}$ submits a message msg, $\mathcal{B}$ computes $\mathtt{C}_{hw} \leftarrow \mathsf{HWF}.\mathtt{Mark}(pp_{hw}, k^*_{hw}, msg)$ and returns a circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ s.t. for any $x \in \{0,1\}^n$:

  $$\mathtt{C}(x) = (\mathtt{C}_{hw}(x) \oplus \mathsf{F}.\mathtt{Eval}(k^*_f, x), \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k^*_{uo}, x))$$

  Here, we also use $\mathcal{Q}^*$ to denote the set of all messages submitted by $\mathcal{A}$ and use $\mathcal{R}^*$ to denote the set of all circuits returned to $\mathcal{A}$.
- Finally, after $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$, $\mathcal{B}$ sets $\tilde{\mathtt{C}}_{uo}$ as a circuit that for any $x \in \{0,1\}^n$, $\tilde{\mathtt{C}}_{uo}(x) = \tilde{\mathtt{C}}(x)[m_1 + 1 : m]$ and submits $\tilde{\mathtt{C}}_{uo}$ to its challenger.

First, it is easy to see, the view of $\mathcal{A}$ in the environment simulated by $\mathcal{B}$ is identical to its view in Game 0 and Game 1. Thus, with a non-negligible probability

$$\mathsf{UOF}.\mathtt{Extract}(pp_{uo}, \tilde{\mathtt{C}}_{uo}) \neq \mathtt{hint}^* \| k^*_f$$

Also, as $\mathcal{A}$ is ϵ-unremoving-admissible, there exists $\mathtt{C}^* \in \mathcal{R}^*$ s.t. $|\{x \in \{0,1\}^n : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq \epsilon \cdot 2^n$. This implies that

$$|\{x \in \{0,1\}^n : \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k^*_{uo}, x) \neq \tilde{\mathtt{C}}_{uo}(x)\}| \leq \epsilon \cdot 2^n \leq \epsilon_2 \cdot 2^n$$

i.e., $\mathcal{B}$ is ϵ_2-admissible. Thus, $\mathcal{B}$ will succeed in breaking the ϵ_2-robust learnability of UOF.

This completes the proof of Lemma E.42. □

Lemma E.43. $\Pr[\mathcal{E}_1 = 1] \leq negl(\lambda)$.

Proof. This comes from the Q-bounded ϵ_1-unremovability of HWF. More precisely, assuming $\Pr[\mathcal{E}_1 = 1]$ is non-negligible, then we can construct an adversary $\mathcal{B}$ that breaks the ϵ_1-unremovability of HWF as follows:

- On receiving the public parameter and hint $(pp_{hw}, \mathtt{hint}^*)$ from its challenger, the adversary $\mathcal{B}$ samples $pp_{uo} \leftarrow \mathsf{UOF}.\mathtt{Setup}(1^\lambda)$, $k^*_f \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$, and $k^*_{uo} \leftarrow \mathsf{UOF}.\mathtt{KeyGen}(pp_{uo}, \mathtt{hint}^* \| k^*_f)$.
- Then it sends $PP = (pp_{hw}, pp_{uo})$ to the adversary $\mathcal{A}$ and answers $\mathcal{A}$'s challenge oracle queries for at most Q times. More precisely, each time the adversary $\mathcal{A}$ submits a message msg, $\mathcal{B}$ submits msg to its own challenge oracle and on receiving the response $\mathtt{C}_{hw}$ from its challenge oracle, it returns a circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ s.t. for any $x \in \{0,1\}^n$:

  $$\mathtt{C}(x) = (\mathtt{C}_{hw}(x) \oplus \mathsf{F}.\mathtt{Eval}(k^*_f, x), \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k^*_{uo}, x))$$

  Here, we also let $\mathcal{Q}^*$ to be the set of all messages submitted by $\mathcal{A}$ and let $\mathcal{R}^*$ to be the set of all circuits returned to $\mathcal{A}$. In addition, we let $\bar{\mathcal{R}}^*$ to be the set of all circuits returned to $\mathcal{B}$.

- Finally, after $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$, $\mathcal{B}$ sets $\tilde{\mathtt{C}}_{hw}$ as a circuit that for any $x \in \{0,1\}^n$, $\tilde{\mathtt{C}}_{hw}(x) = \tilde{\mathtt{C}}(x)[1 : m_1] \oplus \mathsf{F}.\mathtt{Eval}(k_f^*, x)$ and submits $\tilde{\mathtt{C}}_{hw}$ to its challenger.

First, it is easy to see, the view of $\mathcal{A}$ in the environment simulated by $\mathcal{B}$ is identical to its view in Game 1. Thus, we have

$$\mathsf{HWF}.\mathtt{Extract}(pp_{hw}, \tilde{\mathtt{C}}_{hw}, \mathtt{hint}^*) \notin \mathcal{Q}^*$$

with a non-negligible probability.[32] Also, as $\mathcal{A}$ is ϵ-unremoving-admissible[33], there exists $\mathtt{C}^* \in \mathcal{R}^*$ s.t. $|\{x \in \{0,1\}^n : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq \epsilon \cdot 2^n$, i.e., $|\{x \in \{0,1\}^n : \mathtt{C}^*(x)[1 : m_1] \oplus \mathsf{F}.\mathtt{Eval}(k_f^*, x) \neq \tilde{\mathtt{C}}(x)[1 : m_1] \oplus \mathsf{F}.\mathtt{Eval}(k_f^*, x)\}| \leq \epsilon \cdot 2^n$. This implies that there exists $\mathtt{C}_{hw} \in \bar{\mathcal{R}}^*$ s.t.

$$|\{x \in \{0,1\}^n : \mathtt{C}_{hw} \neq \tilde{\mathtt{C}}_{hw}(x)\}| \leq \epsilon \cdot 2^n \leq \epsilon_1 \cdot 2^n$$

i.e., $\mathcal{B}$ is ϵ_1-unremoving-admissible. Thus, $\mathcal{B}$ will succeed in breaking the Q-bounded ϵ_1-unremovability of HWF. This completes proof of Lemma E.43. □

Combining Lemma E.42 and Lemma E.43, we have $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$, i.e., the probability that $\mathcal{A}$ wins in the real experiment $\mathtt{ExptUR}$ is negligible. This completes the proof of unremovability.

[32] Recall that in Game 1, the challenger will abort and output 2 if the extracted ($\mathtt{hint}$, k_f) is not equal to ($\mathtt{hint}^*, k_f^*$).

[33] Since $\mathcal{A}$'s views are identical in Game 0 and Game 1, $\mathcal{A}$ is still ϵ-unremoving-admissible in Game 1.