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.

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

⋆ Corresponding author.

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.

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 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]

[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.

	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.

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 ϵ to denote the fraction of 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 $M = 2^{poly(\lambda)}$.

[2] Formal definition for this assumption can be found in the full version.

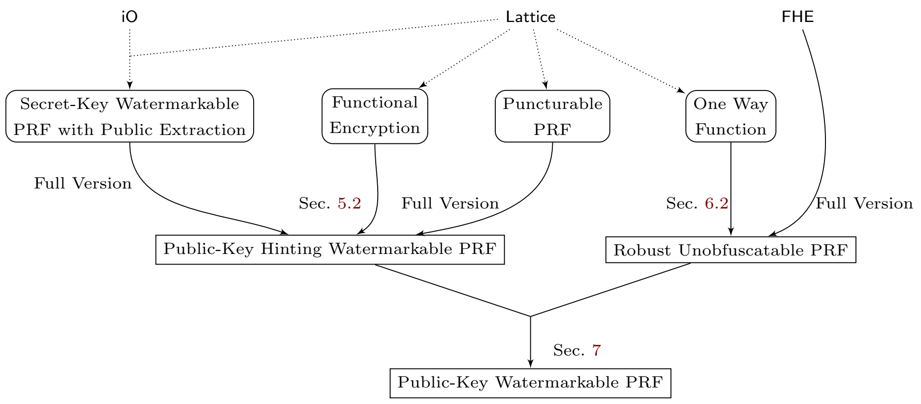
iO
Lattice
FHE
Secret-Key Watermarkable PRF with Public Extraction
Functional Encryption
Puncturable PRF
One Way Function
Full Version
Sec. 5.2
Full Version
Sec. 6.2
Full Version
Public-Key Hinting Watermarkable PRF
Robust Unobfuscatable PRF
Sec. 7
Public-Key Watermarkable PRF

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

- 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 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 $\texttt{hint} = ek$, 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 C and a hint $\texttt{hint} = ek$, the extraction algorithm runs the extraction algorithm of F on input (ek, 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 $\texttt{hint} = (x^*, ct^*)$, 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 C s.t. $\mathsf{C}(x) = \mathsf{F}_{k_{x^*}}(x)$. To test if a circuit C is watermarked, the extraction algorithm first recovers y^* by decrypting ct^* in the hint and outputs "marked" iff C is punctured (i.e., $\mathsf{C}(x^*) \neq y^*$).

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., $\texttt{hint} = (x^*, z^*)$), 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 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 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

[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].

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 $\mathtt{C}$ s.t. $\mathtt{C}(x) = (\mathsf{HF}_{k_{msg}}(x), \mathsf{UF}_{uk_{\mathtt{hint}}}(x))$. Finally, given a circuit $\mathtt{C}$, 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.

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

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

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

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 `Enc` and `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 $y'_3 = \mathsf{UF}_K(1\|x_3 \oplus \gamma\|ct^*_3) = \mathsf{F}_k(x_3 \oplus \gamma\|ct^*_3)$, 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 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 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

$$\Pr[\mathsf{C}(0\|x_1\|ct_1) = \mathsf{UF}_K(0\|x_1\|ct_1)] > 1/2$$
$$\Pr[\mathsf{C}(1\|x_2\|ct_2) = \mathsf{UF}_K(1\|x_2\|ct_2) \wedge \mathsf{C}(2\|x_2 \oplus \alpha\|ct_2) = \mathsf{UF}_K(2\|x_2 \oplus \alpha\|ct_2)] > 1/2$$
$$\Pr[\mathsf{C}(1\|x_3 \oplus \gamma\|ct_3^*) = \mathsf{UF}_K(1\|x_3 \oplus \gamma\|ct_3^*) \wedge \mathsf{C}(3\|x_3\|ct_3^*) = \mathsf{UF}_K(3\|x_3\|ct_3^*)] > 1/2$$

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 the full version 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 P gate by gate. Thus, it can still succeed in extracting the secret information from a circuit 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.

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

[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 $N^{O(|\mathsf{C}|)}$ test points, which is exponential in the size of the circuit 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.

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 $z \overset{\$}{\leftarrow} [0, \frac{q-1}{2}]$ 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 $s \overset{\$}{\leftarrow} \mathcal{S}$ 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 $\texttt{C}$ indexed by a few, say m, constants, we write $\texttt{C}[c_1, \ldots, c_m]$ to denote a circuit with constants c_1, $\ldots, c_m$. We use $\bar{\wedge}$ to denote the NAND gate and suppose that all circuits are composed exclusively by NAND gates unless otherwise specified. We provide more background knowledge and recall definitions of cryptographic primitives employed in this work in the full version.

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:

- $\texttt{Setup}(1^\lambda) \rightarrow PP$: On input the security parameter 1^λ, the setup algorithm outputs the public parameter PP.
- $\texttt{KeyGen}(PP) \rightarrow k$: On input the public parameter PP, the key generation algorithm outputs a PRF key $k \in \mathcal{K}$.
- $\texttt{Eval}(PP, k, x) \rightarrow y$: 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 $PP \leftarrow \texttt{Setup}(1^\lambda)$, $k \leftarrow \texttt{KeyGen}(PP)$, $\texttt{C} \leftarrow \texttt{Mark}(PP, k, msg)$, $x \overset{\$}{\leftarrow} \mathcal{X}$, 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* $\mathtt{C} : \mathcal{X} \to \mathcal{Y}$*, let* $PP \leftarrow \mathtt{Setup}(1^\lambda)$*, then we have* $\Pr[\mathtt{Extract}(PP, \mathtt{C}) \neq \perp] \leq negl(\lambda)$.

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* $PP \leftarrow \mathtt{Setup}(1^\lambda)$*,* $k \leftarrow \mathtt{KeyGen}(PP)$*, 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 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* $PP \leftarrow \mathtt{Setup}(1^\lambda)$ *and* $k \leftarrow \mathtt{KeyGen}(PP)$*. Also, let* $(msg, state) \leftarrow \mathcal{A}_1(PP)$*,* $\mathtt{C} \leftarrow \mathtt{Mark}(PP, k, msg)$*, 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* $\mathtt{C}(x)$*, 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, we construct it from functional encryption schemes. We provide more constructions with different properties in the full version.

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:

- $\texttt{Setup}(1^\lambda) \to PP$: On input the security parameter 1^λ, the setup algorithm outputs the public parameter PP.
- $\texttt{KeyGen}(PP) \to (k, \texttt{hint})$: On input the public parameter PP, the key generation algorithm outputs a PRF key $k \in \mathcal{K}$ and a hint $\texttt{hint}$.
- $\texttt{Eval}(PP, k, x) \to y$: 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 $y \in \mathcal{Y}$.
- $\texttt{Mark}(PP, k, msg) \to \texttt{C}$: 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 $\texttt{C} : \mathcal{X} \to \mathcal{Y}$.
- $\texttt{Extract}(PP, \texttt{C}, \texttt{hint}) \to msg$: 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 $PP \leftarrow \texttt{Setup}(1^\lambda)$, $(k, \texttt{hint}) \leftarrow \texttt{KeyGen}(PP)$, $\texttt{C} \leftarrow \texttt{Mark}(PP, k, msg)$, $x \overset{\$}{\leftarrow} \mathcal{X}$, 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 $\texttt{C} : \mathcal{X} \to \mathcal{Y}$ 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 $\texttt{C}^* \leftarrow \texttt{Mark}(PP, k^*, msg)$ to the adversary.*
3. *Finally, $\mathcal{A}$ submits a circuit $\tilde{\texttt{C}}$ and the experiment outputs 1 iff $\texttt{Extract}(PP, \tilde{\texttt{C}}, \texttt{hint}^*) \notin \mathcal{Q}^*$. 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 $\texttt{C}^* \in \mathcal{R}^*$ that $|\{x \in \mathcal{X} : \texttt{C}^*(x) \neq \tilde{\texttt{C}}(x)\}| \leq \epsilon \cdot |\mathcal{X}|$.*[8]

5.2 The Construction

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:

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

- An FE scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ 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 $\mathsf{FE.Dec}$ is a deterministic algorithm.
- A PKE scheme $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with message space $\{0,1\}^\lambda$ and ciphertext space $\{0,1\}^m$. Also, we use $\mathcal{R}_{\mathsf{PKE.Enc}}$ to denote the randomness space for the algorithm $\mathsf{PKE.Enc}$.
- A PRF $\mathsf{F} = (\mathsf{F.KeyGen}, \mathsf{F.Eval})$ with input space $\{0,1\}^n$ and output space $\mathcal{R}_{\mathsf{PKE.Enc}}$.

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:

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

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

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

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

Fig. 2 The circuit M.

```
Trace
Input: $ind_1, ind_2, p_1, p_2$
  1. $\Delta = |p_1 - p_2|$.
  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. $p_3 = \mathtt{Test}(ind_3)$.
  6. Return $\mathtt{Trace}(ind_1, ind_3, p_1, p_3) \cup$
     $\mathtt{Trace}(ind_3, ind_2, p_3, p_2)$.
```

```
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 $\frac{Acc}{T}$.
```

Fig. 3 The algorithms Trace and Test.

Theorem 5.1. *If* FE *is a secure functional encryption scheme with Q-adaptive indistinguishability and strong correctness*[10], 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.*

We present proof of Theorem 5.1 in the full version.

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, we construct it from one way function. We provide the construction from FHE in the full version.

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:

[10] The strong correctness requires that the decryption will always output $\perp$ given an invalid ciphertext and will always output the correct result given a valid ciphertext. Please see the full version for a formal definition.

- $\texttt{Setup}(1^\lambda) \to PP$: On input the security parameter 1^λ, the setup algorithm outputs the public parameter PP.
- $\texttt{KeyGen}(PP, msg) \to K$: On input the public parameter PP and a message $msg \in \mathcal{M}$, the key generation algorithm outputs a PRF key K.
- $\texttt{Eval}(PP, K, x) \to y$: On input the public parameter PP, a PRF key K, and an input $x \in \mathcal{X}$, the evaluation algorithm outputs an output $y \in \mathcal{Y}$.
- $\texttt{Extract}(PP, \texttt{C}) \to msg$: 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 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 $PP \leftarrow \texttt{Setup}(1^\lambda)$ and $K \leftarrow \texttt{KeyGen}(PP, msg)$, then we have* $\Pr[\texttt{Extract}(PP, \texttt{Eval}(PP, K, \cdot)) \neq msg] = 0$.

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 $PP \leftarrow \texttt{Setup}(1^\lambda)$. Also, let $(msg, state) \leftarrow \mathcal{A}_1(PP)$, $K \leftarrow \texttt{KeyGen}(PP, msg)$, 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 $\texttt{Eval}(PP, K, x)$, and let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \mathcal{X}$ and returns $f(x)$. 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*

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

Here, we say that an adversary $\mathcal{A}$ is ϵ-admissible *if* $|\{x \in \mathcal{X} : \texttt{C}(x) \neq \texttt{Eval}(PP, K, x)\}| \leq \epsilon \cdot |\mathcal{X}|$.[11]

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

6.2 The Construction

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 $\alpha \overset{\$}{\leftarrow} \{0,1\}^\lambda$ and $\beta \overset{\$}{\leftarrow} \{0,1\}^\lambda$. Then it generates PRF keys $k_{enc} \leftarrow \mathsf{F}_{enc}.\mathtt{KeyGen}(1^\lambda)$, $k_{IR} \leftarrow \mathsf{F}_{IR}.\mathtt{KeyGen}(1^\lambda)$, $k_{mask} \leftarrow \mathsf{F}_{mask}.\mathtt{KeyGen}(1^\lambda)$, and $k_{pad} \leftarrow \mathsf{F}_{pad}.\mathtt{KeyGen}(1^\lambda)$. Finally, it outputs the PRF key
$$K = (\alpha, \beta, k_{enc}, k_{IR}, k_{mask}, k_{pad}, msg)$$
- $\mathtt{Eval}$. 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. $ct = r_{\lambda+1} \| \mathsf{F}_{enc}.\texttt{Eval}(k_{enc}, r_{\lambda+1}) \oplus \mu$.
      6. $y_{pad} = \mathsf{F}_{pad}.\texttt{Eval}(k_{pad}, x)[1 : m - (\lambda + 1)]$.
      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. $\mu = \mathsf{F}_{enc}.\texttt{Eval}(k_{enc}, \bar{r}) \oplus \bar{c}$.
      3. $ct = r_{\lambda+2} \| \mathsf{F}_{enc}.\texttt{Eval}(k_{enc}, r_{\lambda+2}) \oplus \mu$.
      4. $y_{pad} = \mathsf{F}_{pad}.\texttt{Eval}(k_{pad}, x)[1 : m - (\lambda + 1)]$.
      5. Output $ct \| y_{pad}$.
   – If $u_0 = 3$:
      1. $z = (u_1, u_2, u_3, u_4, u_5)$.
      2. $y_{mask} = \mathsf{F}_{mask}.\texttt{Eval}(k_{mask}, z)$.
      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. $y_{mask} = \mathsf{F}_{mask}.\texttt{Eval}(k_{mask}, z)$.
      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) $\mu_i = \mathsf{F}_{enc}.\texttt{Eval}(k_{enc}, \bar{r}_i) \oplus \bar{c}_i$.
      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. $y_{mask} = \mathsf{F}_{mask}.\texttt{Eval}(k_{mask}, z)$.
      7. Output $msg \oplus y_{mask}$.
   – If $u_0 = 6$ or $u_0 = 7$:
      1. $y_{pad} = \mathsf{F}_{pad}.\texttt{Eval}(k_{pad}, x)$.
      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) $\bar{y}_{2,j} = \mathsf{C}(\bar{x}_{2,j})$.
   (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. $y_{2,j} = \mathsf{C}(x_{2,j})$.
8. $ct_j = y_{2,j}[1 : \lambda + 1]$.

After obtaining $ct_{|\mathsf{P}|-\lambda+1}, \ldots, ct_{|\mathsf{P}|}$, 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. $u_5 = (ct_{|\mathsf{P}|-\lambda+1}, \ldots, ct_{|\mathsf{P}|})$.
3. $u_0 = 101$.
4. $x_3 = u_0 \| u_1 \| u_2 \| u_3 \| u_4 \| u_5$.
5. $y_3 = \mathsf{C}(x_3)$.
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. $\tilde{y}_3 = \mathsf{C}(\tilde{x}_3)$.
10. $msg = \tilde{y}_3 \oplus y_3$.

Finally, it outputs msg.

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 the full version.

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.

```
P̄
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 $\bar{\mathtt{P}}$.

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. $\epsilon = min(\epsilon_1, \epsilon_2)$.[12]

Our construction is built on the following building blocks:

- A public-key hinting watermarkable PRF $\mathsf{HWF} = (\mathsf{HWF}.\mathtt{Setup}, \mathsf{HWF}.\mathtt{KeyGen}, \mathsf{HWF}.\mathtt{Eval}, \mathsf{HWF}.\mathtt{Mark}, \mathsf{HWF}.\mathtt{Extract})$ 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 $\mathsf{UOF} = (\mathsf{UOF}.\mathtt{Setup}, \mathsf{UOF}.\mathtt{KeyGen}, \mathsf{UOF}.\mathtt{Eval}, \mathsf{UOF}.\mathtt{Extract})$ with input space $\{0,1\}^n$, output space $\{0,1\}^{m_2}$, and message space $\{0,1\}^l$.
- A PRF $\mathsf{F} = (\mathsf{F}.\mathtt{KeyGen}, \mathsf{F}.\mathtt{Eval})$ 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 $pp_{hw} \leftarrow \mathsf{HWF}.\mathtt{Setup}(1^\lambda)$ and $pp_{uo} \leftarrow \mathsf{UOF}.\mathtt{Setup}(1^\lambda)$. 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 $(k_{hw}, \mathtt{hint}) \leftarrow \mathsf{HWF}.\mathtt{KeyGen}(pp_{hw})$, $k_f \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$, and $k_{uo} \leftarrow \mathsf{UOF}.\mathtt{KeyGen}(pp_{uo}, \mathtt{hint}\|k_f)$. Then, it 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. $y_{hw} = \mathsf{HWF}.\mathtt{Eval}(pp_{hw}, k_{hw}, x)$.
  2. $y_f = \mathsf{F}.\mathtt{Eval}(k_f, x)$.

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

3. $y_{uo} = \mathsf{UOF}.\mathtt{Eval}(pp_{uo}, k_{uo}, x)$.
4. Outputs $y = (y_{hw} \oplus y_f, y_{uo})$.

- `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 $\mathtt{C}_{hw} \leftarrow \mathsf{HWF}.\mathtt{Mark}(pp_{hw}, k_{hw}, msg)$. Then it outputs a circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^m$ s.t. for any $x \in \{0,1\}^n$:

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

- `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. $(\mathtt{hint}, k_f) \leftarrow \mathsf{UOF}.\mathtt{Extract}(pp_{uo}, \mathtt{C}_{uo})$.
  3. If $(\mathtt{hint}, k_f) = \perp$: output $\perp$.
  4. Set $\mathtt{C}_{hw}$ as a circuit that for any $x \in \{0,1\}^n$, $\mathtt{C}_{hw}(x) = \mathtt{C}(x)[1 : m_1] \oplus \mathsf{F}.\mathtt{Eval}(k_f, x)$.
  5. Output $msg \leftarrow \mathsf{HWF}.\mathtt{Extract}(pp_{hw}, \mathtt{C}_{hw}, \mathtt{hint})$.

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 the full version.

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

[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.

[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.

[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.

[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.

[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.

[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.

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

[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.

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

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

[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.

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

[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.

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

[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.