Watermarking PRFs from Lattices: Public Extract and Collusion Resistant

Yukun Wang[1] and Mingqiang Wang[2]

[1] School of Mathematics, Shandong University, Jinan, China
mathwangyukun@163.com
[2] School of Mathematics, Shandong University, Jinan, China
wangmingqiang@sdu.edu.cn

Abstract. A software watermarking scheme enables one to embed a "mark " (i.e., a message) into a program without significantly changing the functionality. Moreover, any removal of the watermark from a marked program is futile without significantly changing the functionality of the program. At present, the construction of software watermarking mainly focuses on watermarking pseudorandom functions (PRFs), watermarking public key encryption, watermarking signature, etc.
In this work, we construct new watermarking PRFs from lattices which provide collusion resistant and public extraction. Our schemes are the first to simultaneously achieve all of these properties. The key to the success of our new constructions lies in two parts. First, we relax the notion of functionality-preserving. In general, we require that a marked program (approximately) preserve the input/output behavior of the original program. For our scheme, the output circuit is divided into two parts, one for PRF output and the other for auxiliary functions. As a result, we only require the PRF output circuit to satisfy functionality-preserving. Second, the marking method we use is essentially different form the previous scheme. In general, the mark program will change the output of some special point. The extraction algorithm determines whether the circuit is marked by determining whether the output of some special points has been changed. In our schemes, we use the constrained signature to mark a PRF circuit.

Keywords: Watermarking PRF, Constrained Signature

1 Introduction

A software watermarking scheme allows one to embed a "mark" (i.e., a message) into a program without significantly changing the functionality. Moreover, it should be difficult to remove the watermark from a marked program or create a new programs that are considered to be watermarked without significantly changing the program's behavior. Watermarking schemes are widely used to traitor tracing, ownership protection, etc.

Barak et al. [BGI+01, BGI+12] and Hopper et al. [HMW07] proposed the first rigorous mathematical framework of watermarking schemes. These works are difficult to adapt to stronger security requirements. Early works in this

area [NSS99, YF11, Nis13] gave very partial results showing that certain cryptographic functions can be watermarked, but security only held against restricted adversaries with limited ability to modify the program.

The first positive result for watermarking scheme against arbitrary removal strategies is presented by Cohen et al. in [CHN+16]. They construct a watermarking pseudorandom function (PRF) based on the heavy hammer of indistinguishability obfuscation (IO). After Cohen et al. seminal work, many creative results have been proposed [BLW17, KW17, QWZ18, YAL+18, KW19, YAL+19, YAYX20]. Watermarkable PRFs are constructed form either indistinguishability obfuscation or standard (lattice) assumptions in these works. Even so, there is still a significant gap in security and capabilities between the schemes constructed from IO and those from standard assumption.

In [CHN+16], Cohen et al. also construct watermarking schemes for public encryption (PKE) and signature from their watermarking PRFs. Subsequently Baldimtsi et al. [BKS17] show how to watermark public-key cryptographic primitives with stateful setting. But their work are under a modified security model where a trusted watermarking authority generates both unmarked and marked keys[3]. Recently, Goyal et al. [GKM+19] construct watermarking public-key primitives with desired security properties from simple assumptions, such as the existence of one-way function, standard lattice assumptions, etc. The key enabler of their new constructions is a relaxed notion of functionality-preserving. In all of other works, a marked program (approximately) preserve the input/output behavior of the original program. But in [GKM+19], it is only required to preserve the "functionality" of the original program.[4] Unfortunately, this definition does not apply to watermarkable PRF, because in PRF functions correctness means the input/output behavior of marked program almost everywhere like the original program. **Watermarking PRFs.** A watermarking scheme for a

PRF family F consists of two main algorithm, the marking algorithm and the extraction algorithm. The mark algorithm takes as input a PRF F_k from the PRF family and output a program C which is the marked version of F_k. C and F_k should satisfy the approximate correctness, meaning that $F_k(x) = C(x)$ for all but a negligible fraction of inputs x and these inputs should be hard to find. The extraction algorithm takes as input a circuit C and the extraction key, and output whether the circuit is marked. Generally speaking, if a circuit has been marked, the extraction algorithm will output a symbol "marked" or a message. On the other hand, if a circuit is unmarked, the extraction algorithm will output a symbol $\perp$ or "unmarked".

The main security requirement of watermarking scheme is unremovability which requires that given a marked circuit C^* for a random PRF key, the adversary is not able to create a new circuit C' which has different result with C^* in extraction program without altering the outputs of C^* on a significant fraction

[3] In the standard watermarking model, anyone can generate keys

[4] For example, a marked signature program just need to output valid signatures which can be different from the output of original signature program.

of inputs (There are two cases, one is that different messages is embedded in the C′ and C*, and the other is that C′is unmarked.). Simply put, unremovability requires that it is difficult for the adversary to change the message marked in the circuit without destroying the circuit. Another very important security requirement is unforgeability, which means that anyone without the mark key is not able to generate a new watermarked circuit. **Prior works on watermarking**

PRFs. The first watermarkable PRF scheme was constructed by Cohen et al. in [CHN+16]. Their construction are based on IO and can be achieved secret marking and public extraction, where the unremovability property holds even if the adversary has access to the marking oracle. Later, Yang et al. [YAL+19] improve the scheme in [CHN+16] to achieve collusion resistance. The first watermarkable PRF scheme from standard assumptions was constructed by Kim and Wu in [KW17]. In their scheme, both the marking and the extraction procedures are secret, but the unremovability security property only holds if the adversary has access to the marking oracle. Subsequently, in [QWZ18,KW19], watermarkable PRFs with public marking and extraction queries have been constructed. None of the above-mentioned schemes (from standard assumption) meets the desirable security requirements such as public extraction and collusion resistance. In [YAYX20], Yang et al. provide a generic construction that upgrades a watermarkable PRF without collusion resistance to a collusion resistant one. In addition, the security properties of the original scheme can be preserved.

1.1 Our Results

In this work, we construct a public extract and collusion resistant watermarkable PRF from standard assumption and show that:

Scheme	Public Marking	Public Extraction	Extraction Oracle	PRF Security (Authority)	Collusion Resistant	Hardness Assumotion
[CHN+16]	✗	✓	✓	✓	✗	iO
[KW17]	✗	✗	✗	✓	✗	LWE
[QWZ18]	✓	✗	✓	✗	✗	LWE
[KW19]	✗	✗	✓	✓	✗	LWE
	✓	✗	✓	✓	✗	LWE+RO
[YAL+19]	✗	✓	✓	✓	✓	iO
[YAYX20]	✓	✗	✓	✓	✓	LWE
This Work	✗	✓	✓	✓	✓	LWE+SIS

Table 1. Comparatison

1.2 Technical Overview

In this section, we provide an overview of our technical. First we design a special method to judge watermarking points. Then we construct a watermarking PRFs scheme based on constrained signature. **The difficulty of public extraction.**

At present, all watermarkable PRFs schemes based on standard assumption adopt the following pattern:

- First, the setup procedure generater an unmarked circuit $F_k : \mathcal{X} \to \mathcal{Y}$.
- Second the marking procedure receives a message m and generate $h(m) = X$ where h is a special function and $X \in \mathcal{X}$. Then the marking procedure changes the function value of the points in X to get a marked circuit C where
$$\mathrm{C}(x) = \begin{cases} F_k(x) & \text{if } x \notin X, \\ random & \text{if } x \in X. \end{cases}$$
- Finally, the extraction procedure extracts the information marked in the circuit C by comparing the output of two circuits F_k and C.

It can be seen from the above that the extraction procedure need two key information, the function h and the value of $F_k(x)$ where $x \in X$ to run correctly. The function h is used to get a special set of points. The output of unmarked circuit F_k and marked circuit C are different in this set. The value of $F_k(x)$ where $x \in X$ is used to determine whether a given circuit is marked. It's easy to see that for a given circuit C, if $C(x) = F_k(x)$, $x \in X$, then C must be an unmarked circuit. On the other hand, if $C(x) \neq F_k(x)$ where $x \in X$, then C is a marked circuit.

Anyone who gets the function h and the value of $F_k(x)$ can break the unremovability of watermarkable PRFs which means that the adversary is able to remove the messages embedded in a watermarked program without significantly changing the functionality. When a marked circuit C is received , the adversary first extracts a message in circuit C and then compute the set X and the value of $F_k(x)$ where $x \in X$. After that, the adversary output a new circuit C' with the same output of circuit C except the points in X. When calculating the value of the point $x \in X$, the circuit C' will output $F_k(x)$. Generally speaking, the output of circuit C' is as follows:

$$\mathrm{C}'(x) = \begin{cases} C(x) & \text{if } x \notin X, \\ F_k(x) & \text{if } x \in X. \end{cases}$$

From the above results, it is not difficult to see that C is an unmarked circuit and the most of outputs are the same as circuit C. This break the unremovability of watermarkable PRFs.

In the framework of existing watermark scheme, it is very difficult to extract publicly. Anyone who has the ability to extract can break through unremovability. The main obstacle to achieving public extraction is that there is a huge contradiction between the existing extraction mode and the public extraction.

Our solution. Goyal et al. [GKM+19] construct a watermarkable signature scheme with public extraction. Their extraction procedure just need the verification algorithm of constrained signature. The verification algorithm could be public and does not break unremovability.

Relaxing functionality-preserving. Existing constructions of watermarkable PRFs [KW17, QWZ18, KW19] from standard assumptions do not support properties like collusion resistance (where the watermark remains unremovable even if a user sees multiple marked versions of the program) or public verifiability (where anyone is able to tell if a program is marked). Yang et al. [YAYX20] present a generic construction that upgrades a watermarkable PRF without collusion resistance to a collusion resistant one. Moreover, these constructions rely on heavy cryptographic machinery, such as fully homomorphic encryption, fingerprinting code to watermark a PRF.

Goyal et al. [GKM+19] take a step back and revisit some of the definitions underlying software watermarking. Much like Cohen et al. [CHN+16] started by relaxing perfect functionality-preserving to statistical functionality-preserving and used that as the basis for obtaining the first positive results on watermarking, they also identify another meaningful relaxation of the functionality-preserving requirement. In [CHN+16], functionality-preserving require that the input/output behavior of a marked circuit C' should be almost identical to that of the original circuit C. This is indeed the most natural notion of functionality-preserving when C implements a PRF. In [GKM+19], functionality-preserving require that a marked circuit C' should output a valid value just like the original circuit C. For instance, valid value in watermarkable signing algorithm mean that C and C' should output a valid signatures (even if the signatures output by C' might be different from the ones output by C). In watermarkable encryption algorithm, the valid value means that the decrypted message of C and C' is the same.

Unfortunately, their idea can't be directly applied to the watermarkable PRF, which is caused by the special function of PRF. For a PRF, the valid output means that the output value of C and C' must be exactly the same.

In order to solve this problem, we redefine a weak functionality-preserving. Under this definition, we divide the output of the function into two parts $C = C_1 \| C_2$, where for C_1 part, we need it to satisfy the function preserving property, the C_2 part is not required. When using the function, only the first part C_1 is used as the output of the pseudorandom function, and the second part C_2 is only used as an auxiliary tool in the process of watermark and extraction. With this definition, we can do more operations on C_2 to achieve public extraction.

Change of marking and extraction mode. We no longer use the previous watermark and extraction mode, that is, by comparing the existing circuit with the circuit without watermark to judge whether the circuit is watermarked or extract the watermark message. We creatively extract a new mode, by detecting the legitimacy of the signature to determine whether a circuit is watermarked. For the circuit without watermark, we require that most of the points can output

the legal signature, while for the circuit with watermark, we require that the legal signature can be output at some special points, which can be made public. Because the process of signature validity detection can be carried out publicly (because the verification key is public in the signature scheme), this characteristic makes public extraction possible.

The third output of the extraction algorithm. Generally speaking, the extraction algorithm has two outputs: unmark and msg, where unmark indicates that the circuit has not been watermarked, and msg indicates that the watermark message in the circuit has been extracted. In the previous scheme, the randomly selected circuit is generally classified as unmarked (which is also meaningful, because the randomly selected circuit has not experienced the watermark algorithm, which can be said to be unmarked). In our scheme, due to the particularity of our watermark and extraction mode, our extraction algorithm will have a third output $\perp$, this output indicates that the circuit is randomly selected or tampered by the opponent. In other words, we have the ability to detect whether the input circuit is legal or not. Here, the legal representation is generated according to the honest algorithm, and has not be tampered.

2 Notations

We use the hold upper-case letters (e.g.,$\mathbf{A}$,$\mathbf{B}$) to represent matrices and bold lower-case letters (e.g.,$\mathbf{a}$,$\mathbf{b}$) to represent column vectors.

Let $[\mathcal{A}\|\mathbf{B}]$ denote the concatenation of two matrices and $(\mathcal{A},\mathbf{B})$=$[\mathcal{A}^T\|\mathbf{B}^T]^T$. We use λ to denote the *security parameter* and $\mathrm{negl}(\lambda)$ to denote a negligible function that grows slower than λ^{-c} for any constant $c > 0$ and any large enough value of λ. For an integer N, we let $[N] \stackrel{def}{=} \{1, \ldots, N\}$.

3 Watermarkable PRFs

In this section, we formally propose the definition of watermarkable PRFs, which are adapted from those previous works [CHN+16,BLW17,KW17,QWZ18,KW19, YAL+19, YAYX20].

3.1 The definiton

Definition 3.1 (Watermarkable PRFs) *Fix a security parameter λ, key space $\mathcal{K}$, input space $\mathcal{X} \in \{0,1\}^n$, output space $\mathcal{Y} \in \{0,1\}^m$ and message space $\mathcal{M}$. A watermarkable PRFs* $\mathbf{WPRF} = \{\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Eval}, \mathrm{Mark}, \mathrm{Extract}\}$ *consists of following algorithms:*

- $\mathrm{Setup}\ (1^\lambda) \to (PP, MK, EK)$: *On input the security parameter λ, the setup algorithm outputs the public parameter PP, the watermarking mark key MK and the watermarking extract key EK.*

- KeyGen $(PP) \rightarrow k$: *On input the public parameter PP, the key generation algorithm outputs a PRF key* $k \in \mathcal{K}$.
- Eval $(PP, k, x) \rightarrow y$: *On input the public parameter PP, a PRF keys* $k \in \mathcal{K}$ *and an input* $x \in \mathcal{X}$*, the evaluation algorithm outputs an output* $y \in \mathcal{Y}$.
- Mark$(PP, MK, k, msg) \rightarrow \mathrm{C}$: *On input the public parameter PP, the watermarking mark key MK and a message* $msg \in \mathcal{M}$*, the mark algorithm outputs a marked circuit* $C : \mathcal{X} \rightarrow \mathcal{Y}$.
- Extract$(PP, EK, C) \rightarrow msg$: *On input the public parameter PP, the watermarking extract key EK and a circuit C, the extraction algorithm outputs a message* $m \in \mathcal{M} \cup \{unmarked\}$*, where unmarked denotes that the circuit is unmarked.*

Remark 3.2 (the third output of the extraction algorithm) *In addition to the output* $m \in \mathcal{M} \cup \{unmarked\}$ *in the standard definition, the extraction algorithm will have a third output* $\{\perp\}$ *in our scheme. When the extraction algorithm outputs* $\perp$*, it means that it is a illegal circuit. There are two kinds of illegal situations: one is that the circuit is a marked circuit, but the circuit has been tampered with by the adversary, so it can not extract effective information from the circuit; the other is that the circuit is not generated in a legal way. Therefore, no matter what the situation is, it is meaningless to extract the circuit.*

Correctness. The correctness requirements on a watermarking scheme are twofold. First, the output of the watermarked key should be the same as original key almost everywhere (i.e., the behavior of the watermarked key differs from the original key on only a negligible fraction of the domain). Second, the extraction algorithm should be able to extract the correct message from an honestly-watermarked key.

Definition 3.3 (Functionality Preserving) *For any* $msg \in \mathcal{M}$*, let* $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$*,* $k \leftarrow \mathrm{KeyGen}(PP)$*,* $\mathrm{C} \leftarrow \mathrm{Mark}(PP, MK, k, msg)$*,* $x \xleftarrow{\$} \mathcal{X}$*, then we have* $\Pr[\mathrm{C}(x) \neq \mathrm{Eval}(PP, k, x)] \leq negl(\lambda)$.

We propose a new definition of functionality preserving which called weak functionality preserving. The watermarking circuit C is divided into two parts: C_1 and C_2, where C_1 is used to output PRF values, C_2 is used to assist watermarking detection. C_1 is required to be functionality preserving, and C_2 is not required.

Definition 3.4 (Weak Functionality Preserving) *For any* $msg \in \mathcal{M}$*, let* $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$*,* $k \leftarrow \mathrm{KeyGen}(PP)$*,* $\mathrm{C} = \mathrm{C}_1 \times \mathrm{C}_2 \leftarrow \mathrm{Mark}(PP, MK, k, msg)$*,* $x \xleftarrow{\$} \mathcal{X}$*, then we have* $\Pr[\mathrm{C}_1(x) \neq \mathrm{Eval}(PP, k, x)] \leq negl(\lambda)$.

Definition 3.5 (Extraction Correctness) *For any* $msg \in \mathcal{M}$*, let* $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$*,* $k \leftarrow \mathrm{KeyGen}(PP)$*,* $\mathrm{C} \leftarrow \mathrm{Mark}(PP, MK, k, msg)$*,* $x \xleftarrow{\$} \mathcal{X}$*, then we have* $\Pr[\mathrm{Extract}(PP, EK, C) \neq msg] \leq negl(\lambda)$.

Definition 3.6 (Watermarking Meaningfulness) *For any circuit* $\mathrm{C} : \{0,1\}^n \to \{0,1\}^m$*, let* $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$*, then we have*:

$$\Pr[\mathrm{Extract}(PP, EK, \mathrm{C}) \neq \perp] \leq negl(\lambda)$$

Pseudorandomness. The second property we require on a watermarkable PRFs is the usual notion of pseudorandomness. First, it requires that the watermarkable PRF should be pseudorandomness against an external adversary. Second, the watermarkable PRF should be pseudorandom against the watermarking authority (i.e., the holder of watermarking mark key and extraction key).

Definition 3.7 (Pseudorandomness) *Let* $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$*,* $k \leftarrow \mathrm{KeyGen}(PP)$*, and f be a random function from* $\{0,1\}^n$ *to* $\{0,1\}^m$*. Also, let* $\mathcal{O}_1(\cdot)$ *be an oracle that takes as input a string* $x \in \{0,1\}^n$ *and returns* $\mathrm{Eval}(PP, k, x)$*, and let* $\mathcal{O}_2(\cdot)$ *be an oracle that takes as input a string* $x \in \{0,1\}^n$ *and returns* $f(x)$*. Then for all PPT adversary* $\mathcal{A}$*, we have:*

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

Definition 3.8 Pseudorandomness against the Watermarking Authority Let $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$, $k \leftarrow \mathrm{KeyGen}(PP)$, and f be a random function from $\{0,1\}^n$ to $\{0,1\}^m$. Also, let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $\mathrm{Eval}(PP, k, x)$, and let $\mathcal{O}_2(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $f(x)$. Then for all PPT adversary $\mathcal{A}$, we have:

$$|\Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(PP, ME, EK) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_2(\cdot)}(PP, MK, EK) = 1]| \leq negl(\lambda).$$

Definition 3.9 Weak Pseudorandomness against the Watermarking Authority Let $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$, $k \leftarrow \mathrm{KeyGen}(PP)$, and f be a random function from $\{0,1\}^n$ to $\{0,1\}^m$. Also, let $\mathcal{O}_1(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $(x, \mathrm{Eval}(PP, k, x))$, and let $\mathcal{O}_2(\cdot)$ be an oracle that takes as input a string $x \in \{0,1\}^n$ and returns $(x.f(x))$. Then for all PPT adversary $\mathcal{A}$, we have:

$$|\Pr[\mathcal{A}^{\mathcal{O}_1(\cdot)}(PP, ME, EK) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_2(\cdot)}(PP, MK, EK) = 1]| \leq negl(\lambda).$$

Unremovability The main security notions for a watermarking scheme is unremovability. Conceptually, unremovability says that an efficient adversary is not able to remove or modify the messages embedded in a watermarked program without significantly changing the functionality.

Definition 3.10 (ε -Unremovability) *Fix a security parameter* λ*. A watermarkable PRF* $\mathbf{WPRF} = \{\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Eval}, \mathrm{Mark}, \mathrm{Extract}\}$ *is* ε*-unremovable if for all PPT and* ε*-unremoving-admissible adversaries* $\mathcal{A}$*, we have*

$$\Pr[\mathrm{ExptUR}_{\mathcal{A}}(\lambda) = 1] \leq negl(\lambda),$$

where we define the experiment $\mathrm{ExptUR}_{\mathcal{A}}$ *and* ε*-unremoving-admissible adversaries* $\mathcal{A}$ *as follows:*

1. *The challenger begins by sampling* $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^\lambda)$ *and returns* PP *to the adversary* $\mathcal{A}$.
2. *Then the adversary* $\mathcal{A}$ *is given access to following oracles (but it may be restricted in querying them as discussed below):*
   - ***Mark Key Oracle.*** *The mark key oracle returns* MK *to the adversary* $\mathcal{A}$.
   - ***Extraction Key Oracle.*** *The extraction key oracle returns* EK *to the adversary* $\mathcal{A}$.
   - ***Marking Oracle.*** *On input a message* $msg \in \mathcal{M}$ *and a PRF key* $k \in \mathcal{K}$, *the challenger returns the circuit* $\mathrm{C} \leftarrow \mathrm{Mark}(PP, MK, k, msg)$.
   - ***Extraction Oracle.*** *On input a circuit* C, *the extraction oracle returns a message* $msg \leftarrow \mathrm{Extract}(PP, EK, \mathrm{C})$.
   - ***Challenge Oracle.*** *On input a message* $msg \in \mathcal{M}$, *the challenger samples* $k^* \leftarrow \mathrm{KeyGen}(PP)$ *and returns the circuit* $\mathrm{C}^* \leftarrow \mathrm{Mark}(PP, MK, k^*, msg)$ *to the adversary* $\mathcal{A}$.
3. *Finally,* $\mathcal{A}$ *outputs a circuit* $\widetilde{\mathrm{C}}$ *and the experiment* $\mathrm{ExptUR}_{\mathcal{A}}$ *output 1 iff* $\mathrm{Extract}(PP, EK, \widetilde{\mathrm{C}}) \notin \mathbf{M}^*$. *Here, we use* $\mathbf{M}^*$ *to denote all messages submitted to the challenge oracle and use* $\hat{\mathrm{C}}$ *to denote all circuits returned by challenge oracle.*

We say that a adversary $\mathcal{A}$ *is* ε*-unremoving-admissible if there exists circuit* $\mathrm{C}^* \in \hat{\mathrm{C}}$ *satisfies* $\mathrm{C}^* \sim_\varepsilon \widetilde{\mathrm{C}}$ *which means that* $|\{x \in \{0,1\}^n : \mathrm{C}^*(x) \neq \widetilde{\mathrm{C}}(x)\}| \leq \varepsilon \cdot 2^n$.

We can achieve different security requirements by limiting the adversary capabilities in querying oracles. In a nutshell, we write unremovability as $\mathbb{C}$-$(\mathbb{M}, \mathbb{E})$-ε-unremovability, where $\mathbb{C} \in$ {single key, bounded collusion resistant, unbounded collusion resistant}, $\mathbb{M} \in \{-, \mathrm{MO}, \mathrm{PM}\}$, and $\mathbb{E} \in \{-$,bounded EO, EO, PE$\}$. The above three security notions are explained in more detail below:

- **Constraints on Challenge Oracle.** The unremovability can be defined against an adversary that can:
  - **single key.** make only one query to the challenge oracle.
  - **bounded collusion resistant.** make queries to the challenge oracle for priori bounded times.
  - **unbounded collusion resistant.** make queries to the challenge oracle for unbounded times.
- **Constraints on Mark Key** The unremovability can be defined against an adversary that can:
  - $-$ make no query to the mark key oracle and the marking oracle.
  - **MO** make queries to the marking oracle for unbounded times but make no query to the mark key oracle.
  - **PM** make query to the mark key oracle.
- **Constrains on Extraction Key** The unremovability can be defined against an adversary that can:
  - $-$ make no query to the extraction key oracle and the extraction oracle

- **EO** make queries to the extraction oracle for unbounded times but make no query to the extraction key oracle.
- **PE** make query to the extraction key oracle.

Unforgeability. The another security notion for a cryptographic watermarking scheme is unforgeability. Roughly, it says that an adversary should not be able to construct a marked program without the mark key.

Definition 3.11 (δ-Unforgeability.) *Fix a security parameter λ. A watermarkable PRF* $\mathbf{WPRF} = \{\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Eval}, \mathrm{Mark}, \mathrm{Extract}\}$ *is δ-unforgeability if for all PPT and δ-unforging-admissible adversaries $\mathcal{A}$, we have*

$$\Pr[\mathrm{ExptUF}_{\mathcal{A}}(\lambda) = 1] \leq negl(\lambda),$$

where we define the experiment $\mathrm{ExptUF}_{\mathcal{A}}$ *and δ-unforging-admissible adversaries $\mathcal{A}$ as follows:*

1. *The challenger begins by sampling* $(PP, MK, EK) \leftarrow \mathrm{Setup}(1^{\lambda})$ *and returns PP to the adversary $\mathcal{A}$.*
2. *Then the adversary $\mathcal{A}$ is given access to following oracles (but it may be restricted in querying them as discussed below):*
   - ***Marking Oracle.*** *On input a message $msg \in \mathcal{M}$ and a PRF key $k \in \mathcal{K}$, the challenger returns the circuit* $\mathrm{C} \leftarrow \mathrm{Mark}(PP, MK, k, msg)$.
   - ***Extraction Oracle.*** *On input a circuit* C*, the extraction oracle returns a message* $msg \leftarrow \mathrm{Extract}(PP, EK, \mathrm{C})$.
   - ***Extraction Key Oracle.*** *The extraction key oracle returns EK to the adversary $\mathcal{A}$.*
3. *Finally, $\mathcal{A}$ outputs a circuit* $\widetilde{\mathrm{C}}$ *and the experiment* $\mathrm{ExptUF}_{\mathcal{A}}$ *output 1 iff* $\mathrm{Extract}(PP, EK, \widetilde{\mathrm{C}}) \neq \{\perp, unmark\}$.

We say that a adversary $\mathcal{A}$ is δ-unforging-admissible if for every circuit C_i^* *returned by marking oracle satisfies* $\mathrm{C}_i^* \nsim_{\delta} \widetilde{\mathrm{C}}$.

We can achieve different security requirements by limiting the adversary capabilities in querying oracles. In a nutshell, we write unforgeability as $(\mathbb{M}, \mathbb{E})$-δ-unforgeability, where $\mathbb{M} \in \{-, \mathrm{MO}\}$, and $\mathbb{E} \in \{-, \mathrm{EO}, \mathrm{PE}\}$. The above two security notions are explained in more detail below:

– **Constraints on Mark Key** The unforgeability can be defined against an adversary that can:
  - $-$ make no query to the mark key oracle and the marking oracle.
  - **MO** make queries to the marking oracle for unbounded times but make no query to the mark key oracle.
– **Constrains on Extraction Key** The unforgeability can be defined against an adversary that can:
  - $-$ make no query to the extraction key oracle and the extraction oracle
  - **EO** make queries to the extraction oracle for unbounded times but make no query to the extraction key oracle.
  - **PE** make query to the extraction key oracle.

3.2 Building Block: Constrained Signature

The main building block of our watermarking PRFs is a prefix-constrained signature (which can be built generically from any signature scheme, or more generally, any one-way function). We give the formal definition below:

Definition 3.12 (Constrained Signatures [BF14, Tsa17, GKM+19].) *A constrained signature scheme with message space $\mathcal{M}$ and constraint family $\mathcal{F} \subseteq \mathbf{Funs}[\mathcal{M}, \{0,1\}]$ is a tuple of algorithm $\mathbf{\Pi_{CSig}} = (\mathrm{Setup}, \mathrm{Sign}, \mathrm{Verify}, \mathrm{Constrain}, \mathrm{ConstrainSign})$. with the following properties:*

- $\mathrm{Setup}(1^\lambda) \to (VK, MSK)$: *On input the security parameter λ, the setup algorithm outputs the verification key VK and the master secret key MSK.*
- $\mathrm{Sign}(MSK, m) \to \sigma$: *On input the master secret key MSK and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.*
- $\mathrm{Verify}(VK, m, \sigma) \to b$: *On input the verification key VK, a message $m \in \mathcal{M}$ and a signature σ, the verification algorithm outputs a bit $b \in \{0,1\}$.*
- $\mathrm{Constrain}(MSK, f) \to sk_f$: *On input the master secret key MSK and a function $f \in \mathcal{F}$, the constrain algorithm outputs a constrained key sk_f.*
- $\mathrm{ConstrainSign}(sk_f, m) \to \sigma$: *Om input a constrained key sk_f and a message $m \in \mathcal{M}$, the constrain signing algorithm outputs a signature σ.*

Definition 3.13 (Correctness) *For any message $m \in \mathcal{M}$ and constraint family $\mathcal{F}$, let $(VK, MSK) \leftarrow \mathrm{Setup}(1^\lambda)$ then a constrained signature scheme $\mathbf{\Pi_{CSig}}$ is correct if*

$$\Pr[\mathrm{Verify}(VK, m, \mathrm{Sign}(MSK, m)) = 1] = 1.$$

In addition, for all constraints $f \in \mathcal{F}$ where $f(m) = 1$, let $sk_f \leftarrow \mathrm{Constrain}(MSK, f)$,

$$\Pr[\mathrm{Verify}(VK, m, \mathrm{ConstrainSign}(sk_f, m)) = 1] = 1.$$

Definition 3.14 (Constrained Unforgeabiity) *Fix a security parameter λ. A constrained signature scheme $\mathbf{\Pi_{CSig}} = (\mathrm{Setup}, \mathrm{Sign}, \mathrm{Verify}, \mathrm{Constrain}, \mathrm{ConstrainSign})$ with message space $\mathcal{M}$ and constraint family $\mathcal{F}$ is constrained unforgeability if for all PPT and unforging-admissible adversaries $\mathcal{A}$, we have*

$$\Pr[\mathrm{ExptCSign}_{\mathcal{A}}(\lambda) = 1] \leq negl(\lambda),$$

where we define the experiment $\mathrm{ExptCSign}_{\mathcal{A}}$ and unforging-admissible adversaries $\mathcal{A}$ as follows:

1. *The challenger begins by sampling $(VK, MSK) \leftarrow \mathrm{Setup}(1^\lambda)$ and returns VK to the adversary $\mathcal{A}$.*
2. *Then the adversary $\mathcal{A}$ is given access to following oracles:*
   - ***Constrain Oracle.*** *On input a function $f \in \mathcal{F}$, the challenger returns $sk_f \leftarrow \mathrm{Constrain}(MSK, f)$*
   - ***Signing Oracle.*** *On input a message $m \in \mathcal{M}$, the challenger returns a signature $\sigma \leftarrow \mathrm{Sign}(msk, m)$.*

3. *Finally, $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*) and the experiment* $\text{ExptCSign}_{\mathcal{A}}$ *output 1 iff* $\text{Verify}(VK, m^*, \sigma^*) = 1$.

We say that a adversary $\mathcal{A}$ is unforging-admissible if following conditions hold:

- *The adversary did not make a signing query on message m^*.*
- *The adversary did not make a constrain query on any function $f \in \mathcal{F}$ where $f(m^*) = 1$.*

We say that $\mathbf{\Pi_{CSig}}$ *is secure if for all efficient adversaries $\mathcal{A}$,* $\Pr[\text{ExptCSign}_{\mathcal{A}}(\lambda) = 1] = negl(\lambda)$

3.3 The Construction

In this section, we present our basic construction of a public extraction and collusion resistant watermarkable PRFs
Let λ be the security parameter, $\bar{\varepsilon} = 1 - \varepsilon$
Our construction is built on the following building blocks:

- A Constrained signature scheme $\mathbf{\Pi_{CSig}}$ = (CS.Setup, CS.Sign, CS.Verify, CS.Constrain, CS.ConstrainSign) with message space $\{0,1\}^{n+m}$, signature space $\{0,1\}^m$.
- A pseudorandom function family F = (F.KeyGen, F.Eval) with input space $\{0,1\}^n$ key space $\{0,1\}^\lambda$ and output space $\{0,1\}^m$.

We construct WPRF = (Setup, KeyGen, Eval, Mark, Extract), which has input space $\{0,1\}^n$, output space $\{0,1\}^{2m}$, and message space [1,N] as follows:

- Setup $\rightarrow (PP, MK, EK)$: On input the security parameter λ, the setup algorithm first generates $(VK, MSK) \leftarrow$ CS.Setup and random pick m_i from $\{0,1\}^n$ for all $i \in [N]$ ($m_i \xleftarrow{\$} \{0,1\}^n$ for $i \in [N]$), let the mark set $M = \{m_1, m_2, \cdots, m_N\}$. Then, it outputs the public parameter $PP = (M, VK)$, The mark key $MK = (MSK)$, and the extraction key $EK = (VK)$.
- KeyGen$(PP) \rightarrow k$: On input the public parameter PP, the key generation algorithm generates $k \leftarrow$ F.KeyGen and outputs the PRF key $k \in \{0,1\}^\lambda$.
- Eval $\rightarrow (PP, k, x)$: On input the public parameter $PP = (M, VK)$, a PRF key $k \in \{0,1\}^\lambda$ and an input $x \in \{0,1\}^n$, the evaluation algorithm proceeds as follows:
  1. $sk_f \leftarrow$ CS.Constrain(MSK, f) where

$$f(x) = \begin{cases} 0 & \text{if } x \in M, \\ 1 & \text{otherwise.} \end{cases}$$

  2. Output $\big(\text{F.Eval}(k, x),\ \text{CS.constrainSign}(sk_f, x, \text{F.Eval}(k, x))\big)$
- Mark$(PP, MK, k, msg) \rightarrow$ C : On input the public parameter $PP = (M, VK)$, the mark key $MK = (MSK)$, the PRF key k and a $msg \in [1, N]$, the marking algorithm proceeds as follows:

1. pick m_{msg} from the mark set M
2. $sk_{f_{msg}} \leftarrow \text{CS.Constrain}(MSK, f_{msg})$ where

$$f_{msg}(x) = \begin{cases} 1 & \text{if } x = m_{msg}, \\ 0 & \text{otherwise.} \end{cases}$$

3. Output $\text{C} = \big(\text{F.Eval}(k, x),\ \text{CS.constrainSign}(sk_{f_{msg}}, x, \text{F.Eval}(k, x))\big)$.

- $\text{Extract}(PP, EK, C) \rightarrow msg$: On input the public parameter $PP = (M, VK)$, the extraction key $EK = (VK)$, and a circuit C, the extraction algorithm proceeds as follows:
  1. Set the circuit $\text{C}(x) = \text{C}_1(x)||\text{C}_2(x)$ where $\text{C}_1(x) = \text{C}(x)[1 : m]$ and $\text{C}_2(x) = \text{C}(x)[m + 1 : 2m]$.
  2. $A = 0$.
  3. For $i \in [t]$:
     (a) Sample $x_i \xleftarrow{\$} \{0, 1\}^n$.
     (b) Let $y_i = \text{C}_1(x_i)$, $\pi_i = \text{C}_2(x_i)$.
     (c) If $\text{CS.Verify}(VK, x_i, y_i, \pi_i) = 1$, $A = A + 1$
  4. If $A > t \times \bar{\varepsilon}$, **output** unmark.
  5. Otherwise For $j \in [N]$:
     (a) Let $y_i = \text{C}_1(m_j)$, $\pi_i = \text{C}_2(m_j)$ where $m_j \in M$, and $B = \emptyset$.
     (b) If $\text{CS.Verify}(VK, m_j, y_j, \pi_j) = 1$, put m_j into B.
  6. If there is only one element in B which is $m_{j'}$, let $msg = j'$, **output** msg.
  7. otherwise **output** $\perp$.

Theorem 3.1 *Suppose F is a secure pseudorandom function and* $\mathbf{\Pi_{CSig}}$ *is a constrained unforgeability signature scheme, then WPRF is a secure watermarkable PRF family with collusion resistant and public extraction security.*

We present proof the Theorem 3.1 later in this section, which includes proof of the correctness and pseudorandomness, the unremoveability, and the unforgeability of WPRF.

3.4 Correctness and Pseudorandomness of WPRF

Weak Functionality Preserving. This feature requires that part of the output of the function before and after the watermark is the same. In our scheme, the final output of the circuit is divided into two parts $\text{C}(x) = \text{C}_1(x)||\text{C}_2(x)$where $\text{C}_1(x) = \text{F.Eval}(k, x)$, we use this part as the output of the real pseudo-random function, and $\text{C}_2(x) = \text{CS.constrainSign}(sk_f, x, \text{F.Eval}(k, x))$ is used as auxiliary watermark detection function, not as pseudo-random output. Therefore, as long as there is no significant change in the output of $\text{C}_1(x)$ after the watermark, then the circuit after the watermark can perform the same function as the circuit before the watermark, that is, the two circuits can be approximately regarded as the same function.

In our scheme, the watermark process only changes the output of $\mathrm{C}_2(x)$, for $\mathrm{C}_1(x)$ part doesn't make any changes, so the circuit after the watermark and before the watermark in $\mathrm{C}_1(x)$ must be exactly the same, and the weak functionality preserving can be satisfied.

Extraction Correctness. This feature requires that the watermark message can be extracted correctly from a circuit after being watermarked. When circuit C is watermarked, we can get $\mathrm{C} = \big(\mathrm{F.Eval}(k,x),\ \mathrm{CS.constrainSign}(sk_{f_{msg}}, x, \mathrm{F.Eval}(k,x))\big)$. If and only if $x = m_{msg}$, we can get a valid signature form $\mathrm{C}_2(x)$, when $x \neq m_{msg}$, the signature output by $\mathrm{C}_2(x)$ cannot be verified. In addition, all possible messages are in the set M. Therefore, we only need to check the elements in Min turn until there is a m_{msg} can satisfy $\mathrm{CS.Verify}(VK, m_{msg}, \mathrm{C}_1(m_{msg}), \mathrm{C}_2(m_{msg})) = 1$, in this time msg is the message watermarked in the circuit.

Watermarking Meaningfulness. This characteristic requires that the output of the extraction algorithm should be $\perp$ for the randomly selected circuit. When we randomly select a circuit C, the part of $\mathrm{C}_2(x)$ can hardly output a valid signature, that is, CS.Verify = 0. At this time, the fourth and sixth steps of the extraction algorithm will not stop, but will output $\perp$ in step 7.

Pseudorandomness. This feature requires that the output of the watermark function is pseudo-random, which can be directly guaranteed by the pseudo randomness of the selected function F, Because it's only $\mathrm{C}_1(x)$ part is the real output of the function.

3.5 Unremovability of WPRF

In this section, we prove the fully collusion resistant$-(MO, PE)-\varepsilon-$unremovability of WPRF, assuming that $\mathbf{\Pi_{CSig}}$ is a constrained unforgeability signature scheme and F is a secure pseudorandom function

Theorem 3.2 *Suppose* $\mathbf{\Pi_{CSig}}$ *satisfies constrained unforgeability and F is a secure pseudorandom function, then the watermarkable PRF scheme satisfies fully collusion resistant*$-(MO, PE) - \varepsilon-$*unremovability.*

Proof. Assume that there is a PPT adversary $\mathcal{A}$ can win the security experiment $\mathrm{ExptUR}_{\mathcal{A}}$ with non-negligible probability β. We construct a PPT simulator $\mathcal{S}$ that breaks the constrained unforgeability for $\mathbf{\Pi_{CSig}}$.

- First the simulator receives $VK \leftarrow \mathrm{CS.Setup}(1^\lambda)$ from the challenger and generates the mark set $M = \{m_1, m_2, \cdots, m_N\}$. Then, it output the parameter PP=(M,VK), the extraction key $EK = (VK)$ to the adversary $\mathcal{A}$.
- Then the simulator $\mathcal{S}$ samples the challenge key $k^* \leftarrow \{0,1\}^\lambda$ which are used in answering the challenge oracle.
- Next the simulator $\mathcal{S}$ answers $\mathcal{A}$'s oracle queries, include marking oracle and challenge oracle. Once $\mathcal{A}$ submits a message $i \in [N]$ and a PRF key k for marking Oracle, the simulator $\mathcal{S}$ proceeds as follows:

1. generate an empty set M^* and put i into M^*
2. send $(m_i, \text{F.Eval}(k, m_i)$ to the challenger for signing oracle and receive a signature σ_i.
3. Generate a random function $\text{R} : \{0,1\}^n \to \{0,1\}^m$ and output the circuit $\text{C}_i = \text{C}_1 || \text{C}_2$ to $\mathcal{A}$ where $\text{C}_1(x) = \text{F.Eval}(k, x)$ and

$$C_2(x) = \begin{cases} \sigma & \text{if } x = m_i, \\ R(x) & \text{otherwise.} \end{cases}$$

Once $\mathcal{A}$ submits a message $i \in [N]$ for challenge oracle, the simulator $\mathcal{S}$ proceeds as follows:

1. generate an empty set M^* and put i into M^*
2. send $(m_i, \text{F.Eval}(k^*, m_i)$ to the challenger for signing oracle and receive a signature σ_i^*.
3. Generate a random function $\text{R} : \{0,1\}^n \to \{0,1\}^m$ and output the circuit $\text{C}_i^* = \text{C}_{i1}^* || \text{C}_{i2}^*$ to $\mathcal{A}$ where $\text{C}_{i1}^*(x) = \text{F.Eval}(k^*, x)$ and

$$C_{i2}^*(x) = \begin{cases} \sigma_i^* & \text{if } x = m_i, \\ R(x) & \text{otherwise.} \end{cases}$$

– Finally, $\mathcal{A}$ submits a circuit $\widetilde{\text{C}} = \widetilde{\text{C}}_1 || \widetilde{\text{C}}_2$ to the simulator $\mathcal{S}$ and the simulator $\mathcal{S}$ proceeds as follows
  1. Use extraction algorithm to $\widetilde{\text{C}}$
  2. If $\text{Extract}(PP, EK, \widetilde{\text{C}})$ outputs unmark, then sample $x \xleftarrow{\$} \{0,1\} \setminus M$, let $y = \widetilde{\text{C}}_1(x)$, and $\sigma^* = \widetilde{\text{C}}_2(x)$, outputs a message-signature pair (x, y, σ^*) to the challenge
  3. If $\text{Extract}(PP, EK, \widetilde{\text{C}})$ outputs a message $j \in [N]$, if $j \in M^*$, output **abort**. Otherwise let $y = \widetilde{\text{C}}_1(m_j)$, and $\sigma^* = \widetilde{\text{C}}_2(m_j)$, outputs a message-signature pair (m_j, y, σ^*) to the challenge
  4. If $\text{Extract}(PP, EK, \widetilde{\text{C}})$ outputs $\perp$, output **abort**.

Now we show that the adversary $\mathcal{A}$ who wins the security experiment $\text{ExptUR}_{\mathcal{A}}$ can be used to breaks the constrained unforgeability for $\mathbf{\Pi_{CSig}}$. Because $\mathcal{A}$ is ε-unremoving-admissible, that means $\widetilde{\text{C}}_1 \sim_\varepsilon \widetilde{\text{C}}_{i1}^*$ for all $i \in M^*$. Then according to the different output of the extraction algorithm, there are the following three results:

– **Unmark.** If $\text{Extract}(PP, EK, \widetilde{\text{C}})$ outputs unmark, for $x \xleftarrow{\$} \{0,1\} \setminus M$, $\Pr[\text{CS.Verify}(VK, x, \widetilde{\text{C}}_1(x), \widetilde{\text{C}}_2(x)) = 1] \geq \bar{\varepsilon}$. That means we found a valid message-signature pair $(x, \widetilde{\text{C}}_1(x), \widetilde{\text{C}}_2(x))$ for $\mathbf{\Pi_{CSig}}$ and $(x, \widetilde{\text{C}}_1(x))$ has not been queried. Therefor we can get $\Pr[\text{ExptCSign}_{\mathcal{A}}(\lambda) = 1] \geq \bar{\varepsilon}$
– **j.** If $\text{Extract}(PP, EK, \widetilde{\text{C}})$ outputs a message $j \in [N]$ and $j \in M^*$, that means $\mathcal{A}$ is not ε-unremoving-admissible. So we can get $\Pr[j \notin M^*] = \beta$. Furthermore, $(m_j, \widetilde{\text{C}}_1(m_j))$ has not been queried and $(m_j, \widetilde{\text{C}}_1(m_j), \widetilde{\text{C}}_2(m_j))$ is a valid message-signature pair for $\mathbf{\Pi_{CSig}}$ because of $\text{CS.Verify}(VK, x, \widetilde{\text{C}}_1(m_j), \widetilde{\text{C}}_2(m_j)) = 1$.

– $\perp$ If Extract$(PP, EK, \widetilde{\mathrm{C}})$ outputs $\perp$ then $\mathcal{A}$ is not ε-unremoving-admissible.

In conclusion, if there is a PPT adversary $\mathcal{A}$ can win the security experiment $\mathrm{ExptUR}_{\mathcal{A}}$ with non-negligible probability β, then the simulator $\mathcal{S}$ wins the security experiment $\mathrm{ExptCSign}_{\mathcal{A}}$ for $\mathbf{\Pi_{CSig}}$ with probability at least $\beta\bar{\varepsilon}$.

Collusion Resistant. The collusion resistant feature mainly considers whether the scheme can remain secure when different messages are inquired in the marking oracle with the same watermark key. The reason why we need to consider this feature is that in the previous scheme, if we get the watermark circuit for different messages with the same key, we can forge a new watermark circuit naturally, and only need to take part of the watermark points to recombine. This is determined by the watermark and extraction mode. But in our scheme, because our watermark and extraction process operate on the part of constrained signature, the unforgeability of signature can help us achieve the effect of collusion resistant, that is, if the adversary can collude, it means that the adversary can forge the signature.

We don't restrict adversaries to query the same key and different messages many times in the marking oracle instruction, so our scheme is also collusion resistant.

Public Extraction. In our game, the extraction key is directly sent to the adversary, that is, we allow the adversary to extract messages in any circuit by using the extraction key, so our scheme can meet the public extraction characteristics.

3.6 Unforgeability of WPRF

In this section, we prove the fully collusion resistant$-(MO, PE)-\delta-$unforgeability of WPRF, assuming that $\mathbf{\Pi_{CSig}}$ is a constrained unforgeability signature scheme and F is a secure pseudorandom function

Theorem 3.3 *Suppose* $\mathbf{\Pi_{CSig}}$ *satisfies constrained unforgeability and F is a secure pseudorandom function, then the watermarkable PRF scheme satisfies fully collusion resistang*$-(MO, PE) - \delta-$*unforgeability.*

Proof. Assume that there is a PPT adversary $\mathcal{A}$ can win the security experiment $\mathrm{ExptUF}_{\mathcal{A}}$ with non-negligible probability β. We construct a PPT simulator $\mathcal{S}$ that breaks the constrained unforgeability for $\mathbf{\Pi_{CSig}}$.

– First the simulator receives $VK \leftarrow \mathrm{CS.Setup}(1^{\lambda})$ from the challenger and generates the mark set $M = \{m_1, m_2, \cdots, m_N\}$. Then, it output the parameter PP=(M,VK), the extraction key $EK = (VK)$ to the adversary $\mathcal{A}$.
– Then the simulator $\mathcal{S}$ answers $\mathcal{A}$'s oracle queries, include marking oracle. Once $\mathcal{A}$ submits a message $i \in [N]$ and a PRF key k for marking Oracle, the simulator $\mathcal{S}$ proceeds as follows:

1. generate an empty set M^* and put i into M^*.
2. send $(m_i, \mathrm{F.Eval}(k, m_i)$ to the challenger for signing oracle and receive a signature σ_i.
3. Generate a random function $\mathrm{R} : \{0,1\}^n \to \{0,1\}^m$ and output the circuit $\mathrm{C}_i = \mathrm{C}_1 || \mathrm{C}_2$ to $\mathcal{A}$ where $\mathrm{C}_1(x) = \mathrm{F.Eval}(k, x)$ and

$$C_2(x) = \begin{cases} \sigma & \text{if } x = m_i, \\ R(x) & \text{otherwise.} \end{cases}$$

– Finally, $\mathcal{A}$ submits a circuit $\widetilde{\mathrm{C}} = \widetilde{\mathrm{C}}_1 || \widetilde{\mathrm{C}}_2$ to the simulator $\mathcal{S}$ and the simulator $\mathcal{S}$ proceeds as follows
  1. Use extraction algorithm to $\widetilde{\mathrm{C}}$.
  2. If $\mathrm{Extract}(PP, EK, \widetilde{\mathrm{C}})$ outputs unmark, output **abort**.
  3. If $\mathrm{Extract}(PP, EK, \widetilde{\mathrm{C}})$ outputs a message $j \in [N]$, if $j \in M^*$, output **abort**. Otherwise let $y = \widetilde{\mathrm{C}}_1(m_j)$, and $\sigma^* = \widetilde{\mathrm{C}}_2(m_j)$, outputs a message-signature pair (m_j, y, σ^*) to the challenge.
  4. If $\mathrm{Extract}(PP, EK, \widetilde{\mathrm{C}})$ outputs $\perp$, output **abort**.

Now we show that the adversary $\mathcal{A}$ who wins the security experiment $\mathrm{ExptUF}_{\mathcal{A}}$ can be used to breaks the constrained unforgeability for $\mathbf{\Pi_{CSig}}$.

If $\mathrm{Extract}(PP, EK, \widetilde{\mathrm{C}})$ output a message $j \in [N]$ for $j \notin M^*$, this means that the extraction algorithm detects a valid signature. Then we can get $\mathrm{CS.Verify}(VK, x, \widetilde{\mathrm{C}}_1(m_j), \widetilde{\mathrm{C}}_2(m_j)) = 1$ and $(m_j, \widetilde{\mathrm{C}}_1(m_j), \widetilde{\mathrm{C}}_2(m_j))$ is a valid message signature pair. This breaks the constrained unforgeability for $\mathbf{\Pi_{CSig}}$.

Collusion Resistant. We don't limit adversaries to quire the consent key and different messages many times in the marking oracle, so our scheme is collusion resistant.

Public Extraction. In our game, the extraction key is directly sent to the adversary, that is, we allow the adversary to extract messages in any circuit by using the extraction key, so our scheme can meet the public extraction characteristics.

Acknowledgements

The authors are supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant No. 61832012) and the National Cryptography Development Fund (Grant No. MMJJ20180210).

References

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 520–537. Springer, 2014.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

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

[BKS17] Foteini Baldimtsi, Aggelos Kiayias, and Katerina Samari. Watermarking public-key cryptographic functionalities and implementations. In *ISC*, volume 10599 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2017.

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

[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, 2019.

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

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

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

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

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

[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, 2018.

[Tsa17] Rotem Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In *TCC*, pages 489–518, 2017.

[YAL+18] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Unforgeable watermarking schemes with public extraction. In *SCN*, pages 63–80, 2018.

[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, 2019.

[YAYX20] Rupeng Yang, Man Ho Au, Zuoxia Yu, and Qiuliang Xu. Collusion resistant watermarkable prfs from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2020:695, 2020.

[YF11] Maki Yoshida and Toru Fujiwara. Toward digital watermarking for cryptographic data. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 94-A(1):270–272, 2011.