Collusion Resistant Watermarkable PRFs from Standard Assumptions

Rupeng Yang[1⋆†], Man Ho Au[1⋆], Zuoxia Yu[1†], and Qiuliang Xu[2]

[1] Department of Computer Science, The University of Hong Kong, Hong Kong, China
orbbyrp@gmail.com, allenau@cs.hku.hk, zuoxia.yu@gmail.com
[2] School of Software, Shandong University, Jinan, 250101, China
xql@sdu.edu.cn

Abstract. A software watermarking scheme can embed a message into a program without significantly changing its functionality. Moreover, any attempt to remove the embedded message in a marked program will substantially change the functionality of the program. Prior constructions of watermarking schemes focus on watermarking cryptographic functions, such as pseudorandom function (PRF), public key encryption, etc.

A natural security requirement for watermarking schemes is collusion resistance, where the adversary's goal is to remove the embedded messages given multiple marked versions of the same program. Currently, this strong security guarantee has been achieved by watermarking schemes for public key cryptographic primitives from standard assumptions (Goyal et al., CRYPTO 2019) and by watermarking schemes for PRFs from indistinguishability obfuscation (Yang et al., ASIACRYPT 2019). However, no collusion resistant watermarking scheme for PRF from standard assumption is known.

In this work, we solve this problem by presenting a generic construction that upgrades a watermarkable PRF without collusion resistance to a collusion resistant one. One appealing feature of our construction is that it can preserve the security properties of the original scheme. For example, if the original scheme has security with extraction queries, the new scheme is also secure with extraction queries. Besides, the new scheme can achieve unforgeability even if the original scheme does not provide this security property. Instantiating our construction with existing watermarking schemes for PRF, we obtain collusion resistant watermarkable PRFs from standard assumptions, offering various security properties.

1 Introduction

A watermarking scheme allows one to embed some information into a program while preserving its functionality. Moreover, it should be difficult for an adversary to remove the embedded information without destroying the marked

⋆ Corresponding author.

† Part of the work was done while the author was with the Department of Computing, The Hong Kong Polytechnic University.

program. Watermarking schemes are widely employed in many applications, including ownership protection, traitor tracing, etc.

The theoretical study of watermarking schemes was initiated by Barak et al. [BGI+01] and Hopper et al. [HMW07]. However, no concrete construction is provided in both works. It is extremely difficult to construct provably secure watermarking schemes and early works in this area [NSS99, YF11, Nis13] only consider restricted adversaries, which are not allowed to change the format of the watermarked object.

The first watermarking scheme with provable security against arbitrary removal strategies is presented by Cohen et al. in [CHN+16]. Specifically, they construct a watermarkable pseudorandom function (PRF) from indistinguishability obfuscation. In subsequent works [BLW17,KW17,QWZ18,YAL+18,KW19, YAL+19], watermarkable PRFs are constructed from either indistinguishability obfuscation or standard (lattice) assumptions. However, there is still a significant gap in security between the schemes constructed from indistinguishability obfuscation and those from standard assumptions.

In [CHN+16], Cohen et al. also construct watermarking schemes for public key encryption (PKE) and signature from their watermarkable PRFs. Subsequently, (stateful) watermarking schemes for PKE are constructed from any PKE scheme [BKS17]. Recently, in [GKM+19], Goyal et al. construct watermarking schemes for various public key cryptographic primitives with nearly all desired security properties from simple assumptions, such as the existence of one-way function, standard lattice assumptions, etc. This is achieved by a slight relaxation on the correctness of the watermarking scheme. More precisely, in their definition, a marked program is not required to approximately preserve the input/output behaviors of the original program, and instead, it is only required to preserve the "functionality" of the original program.[1] Unfortunately, such relaxation is not applicable to watermarkable PRF, whose functionality is exactly specified by its input/output behaviors.

Watermarking PRFs. A watermarking scheme for a PRF family F consists of two main 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 evaluates $F_k(\cdot)$ correctly on almost all inputs. The extraction algorithm takes as input the extraction key and a circuit, and outputs either a message or a symbol $\perp$, which indicates that the circuit is unmarked.

The main security property of a watermarking scheme is *unremovability*, which requires that given a marked circuit $\mathtt{C}^*$ for a random PRF key (namely, the challenge key), the adversary is not able to remove or modify the embedded message[2] without altering the outputs of $\mathtt{C}^*$ on a significant fraction of inputs. An additional security property is *unforgeability*, which prevents anyone without the

[1] For example, to mark a signing algorithm, it is sufficient that the marked program can still output valid signatures.

[2] That is, the extraction algorithm should still output the original message when extracting a circuit created by the adversary.

mark key from generating a new watermarked circuit. Besides, for watermarkable PRF, it is usually required to have *pseudorandomness against the watermarking authority*, i.e., the pseudorandomness holds against an adversary who possesses the mark key and the extraction key.

When defining security (either unremovability or unforgeability) for watermarking schemes, adversaries with different capabilities are considered. For example, if the adversary is allowed to access more than one marked circuit of the challenge key, the scheme is *collusion resistant*. Moreover, we say that the scheme has security with *marking (oracle) queries* if the security is defined against an adversary who can obtain marked circuits of its generated keys and we say that the scheme has security with *public marking* if the adversary can obtain the mark key. Besides, we say that the scheme has security with *extraction (oracle) queries* if the security is defined against an adversary who can obtain extraction results of its generated circuits and we say that the scheme has security with *public extraction* if the adversary can obtain the extraction key.

Prior works on watermarkable PRFs. Watermarkable PRF is first constructed by Cohen et al. in [CHN$^+$16]. The scheme is constructed from indistinguishability obfuscation and has unremovability with public extraction. Later, in [YAL$^+$19], Yang et al. improve Cohen et al.'s scheme to achieve collusion resistance. Both constructions rely on the full power of indistinguishability obfuscation and it seems infeasible to instantiate them from standard assumptions.

Towards constructing watermarkable PRF from standard assumptions, Boneh et al. [BLW17] propose a new approach that builds watermarkable PRF from variants of constrained PRFs [BW13, KPTZ13, BGI14]. The schemes provided in [BLW17] still rely on the existence of indistinguishability obfuscation. Then, building on Boneh et al.'s framework, watermarkable PRFs from standard assumptions are developed. In [KW17], Kim and Wu present the first watermarkable PRF from standard assumptions. The scheme only achieves security with marking queries. Subsequently, in [QWZ18, KW19], watermarkable PRFs that have security with public marking and extraction queries are constructed. However, all of these constructions (from standard assumptions) fail to provide desirable security properties such as security with public extraction and collusion resistance.

The goal of this work is to narrow the gap in security between indistinguishability obfuscation based watermarkable PRFs and standard assumptions based ones. We note that security with extraction queries, which is a natural stepping stone towards security with public extraction, is already achieved by previous watermarkable PRFs from standard assumptions [QWZ18, KW19]. In contrast, no positive result on collusion resistant watermarkable PRF from standard assumptions is known. Therefore, our main objective is to design *collusion resistant* watermarkable PRF that can be instantiated from *standard assumptions*.

1.1 Our Results

In this work, we explore the possibility to build collusion resistant watermarkable PRF from standard assumption and show that:

	Collusion Resistance	Unremovability with PM	Unremovability with EO	Unforgeability	Unforgeability with EO	Unforgeability with PE	Pseudorandomness against Authority
[KW17]	✗	✗	✗	✓	✗	✗	fully
[QWZ18]	✗	✓	✓	✗	–	–	✗
[KW19]	✗	✗	✓	✓	✓	✗	weak†
	✗	✓	✓	✗	–	–	weak†
Ours + [KW17]	✓	✗	✗	✓	✓	✓	fully
Ours + [QWZ18]	✓	✓	✓*	✓	✓	✓	✗
Ours + [KW19]	✓	✓	✓*	✓	✓	✓	weak†

∗: The adversary can only query the extraction oracle for a prior bounded number of times.
†: Actually, a stronger T-restricted pseudorandomness (see [KW19]) can be achieved.

Table 1: Security properties achieved by watermarkable PRFs from standard assumption. The default setting for unremovability, which is achieved by all constructions, is unremovability with marking queries. We use unremovability with PM to denote unremovability with public marking and use unremovability with EO to denote unremovability with extraction (oracle) queries. The default setting for unforgeability is unforgeability with marking queries. We use unforgeability with EO to denote unforgeability with extraction (oracle) queries and use unforgeability with PE to denote unforgeability with public extraction. We refer the reader to Sec. 4.1 for a more detailed discussion on different levels of unremovability and unforgeability.

Theorem 1.1 (Informal). *Assuming the existence of secure watermarkable PRF, there exist* collusion resistant *watermarkable PRFs. Especially, collusion resistant watermarkable PRFs exist assuming the worst-case hardness of appropriately parameterized GapSVP problems.*

We prove Theorem 1.1 by presenting a generic transformation from watermarkable PRF without collusion resistance to collusion resistant watermarkable PRF. Our transformation can approximately preserve the security of the original scheme. For example, if the original scheme has security with public marking, then so does the new scheme. Besides, by using our transformation, the new scheme has very strong unforgeability even if the original scheme is not unforgeable. This is achieved by a novel technique that adds unforgeability to a large class of watermarkable PRFs, which may be of independent interest.

By applying our transformation to existing watermarkable PRFs from standard assumptions [KW17, QWZ18, KW19], we obtain lattice based collusion resistant watermarkable PRFs with various features. The results are summarized in Table 1.

The key component of our transformation is a fingerprinting code with enhanced security, where the adversary can query an extraction oracle that outputs the decoding of its submitted word. Surprisingly, this natural security requirement has not been considered in previous works. In this work, we change this situation by defining and constructing fingerprinting code that has security with extraction queries. The new primitive is also potentially useful for copyright protection in practical applications.

One caveat is that our constructions of fingerprinting code (and thus collusion resistant watermarking schemes) are only secure against an adversary that can make at most q queries to the extraction oracle, where q is a priori bounded polynomial. Also, the message spaces of our fingerprinting code and watermarkable PRFs are of polynomial-size.[3] It is an interesting open problem to design fingerprinting codes and standard assumption based collusion resistant watermarkable PRFs without these restrictions.

1.2 Technical Overview

In this section, we provide an overview of our techniques. We first recall current approach for constructing (single key secure) watermarkable PRF from standard assumption and identify the difficulty for achieving collusion resistance via this approach. Then we show our ideas to overcome the difficulty.

The difficulty. Existing constructions of watermarkable PRF from standard assumptions [KW17, QWZ18, KW19] are all built on (variants of) constrained PRFs, following the blueprint proposed by Boneh et al. in [BLW17]. A constrained PRF F is a family of PRF that allows one to derive a constrained key ck from a PRF key k, where $\mathsf{F}_{ck}(\cdot)$ and $\mathsf{F}_k(\cdot)$ evaluate identically on almost all inputs except at some "punctured" points[4]. Its security requires that given the constrained key ck, $\mathsf{F}_k(x)$ is still pseudorandom if x is a punctured point. A constrained PRF is constraint-hiding if the constrained key does not reveal the punctured points. A (constraint-hiding) constrained PRF is collusion resistant if the security remains even if the adversary can obtain multiple constrained keys derived from a PRF key. Next, we briefly review how to watermark a constrained PRF family F.

To watermark a PRF key k of F, the marking algorithm first generates an input x^* and produces a constrained key ck that is punctured on x^* (i.e., $\mathsf{F}_{ck}(x^*) \neq \mathsf{F}_k(x^*)$ and $\mathsf{F}_{ck}(x) = \mathsf{F}_k(x)$ for all $x \neq x^*$). The marked version of k is just a circuit that evaluates $\mathsf{F}_{ck}(\cdot)$. To test if a circuit $\mathtt{C}$ is marked, the extraction algorithm recovers x^* by using the extraction key and checks if $\mathtt{C}$ is a constrained key punctured on x^*. This is accomplished via either checking if $\mathtt{C}(x^*)$ is in a specific set ([KW17]) or checking if $\mathtt{C}(x^*) \neq \mathsf{F}_k(x^*)$ ([QWZ18, KW19]). The variants of constrained PRF used in these works support such checks. Security of the watermarking schemes relies on the fact that the punctured point x^* (or the output $\mathsf{F}_k(x^*)$) is hidden from the adversary. Based on this, to embed a message $msg \in \{0,1\}^l$ (instead of a mark) into a PRF key, the marking algorithm will encode the message into the punctured points. One simple method is to generate $2l$ inputs $(x^*_{1,0}, x^*_{1,1}, \dots, x^*_{l,0}, x^*_{l,1})$ and puncture the PRF key on $\{x^*_{i,msg[i]}\}_{i\in[l]}$. Then, the extraction algorithm can recover the i-th bit of the embedded message via checking if the circuit is punctured on $x^*_{i,0}$ or if it is punctured on $x^*_{i,1}$.

[3] In contrast, existing watermarkable PRFs without collusion resistance have exponential message spaces.

[4] The punctured points may be selected by a general constraint, e.g. a circuit.

The main obstacle to achieving collusion resistance via the above approach is that the underlying (variants of) constrained PRFs are not collusion resistant. Specifically, for the instantiations provided in [KW17, KW19], one can recover the PRF key k and thus compromise security of the watermarking scheme if it is given two different constrained keys derived from k. For the scheme constructed in [QWZ18], it can be instantiated from any constraint-hiding constrained PRF for general constraint. However, to the best of our knowledge, no constraint-hiding constrained PRF from standard assumption [BKM17, CC17, BTVW17, PS18, CVW18, AMN+18, DKN+20] is known to achieve collusion resistance. Moreover, as proved in [CC17], collusion resistant constraint-hiding constrained PRF for general constraint implies indistinguishability obfuscation.

Our solution. To get around this obstacle, our key idea is to encode bits of a message into different "keys" instead of encoding them into different inputs. Next, we first illustrate how the idea works with a *failed* attempt, then we show how to correct it. We also discuss some barriers to achieving other desirable security properties and explain how to solve them.

The first attempt. The watermarking object of our initial attempt is a new PRF family $\tilde{\mathsf{F}}$ that is a "repetition" of l constrained PRFs, i.e., $\tilde{\mathsf{F}}_{k_1,\ldots,k_l}(x) = (\mathsf{F}_{k_1}(x), \ldots, \mathsf{F}_{k_l}(x))$. To embed a message $msg \in \{0,1\}^l$ into a key $\boldsymbol{k} = (k_1, \ldots, k_l)$ of $\tilde{\mathsf{F}}$, the marking algorithm first generates $2l$ inputs $(x^*_{1,0}, x^*_{1,1}, \ldots, x^*_{l,0}, x^*_{l,1})$, then it punctures k_i on $x^*_{i,msg[i]}$ to obtain a constrained key ck_i. The marked version of $\boldsymbol{k}$ is a circuit that computes $(\mathsf{F}_{ck_1}(\cdot), \ldots, \mathsf{F}_{ck_l}(\cdot))$. Then, on input a circuit, the extraction algorithm can recover the i-th bit of the embedded message via checking if the i-th part of the circuit is punctured on $x^*_{i,0}$ or if it is punctured on $x^*_{i,1}$.

Now, we examine what is guaranteed from this construction. For simplicity, we consider the simplified case that $l = 3$ and that the adversary only obtains two marked circuits $\mathsf{C}^{(1)} = (\mathsf{F}_{ck^{(1)}_1}(\cdot), \mathsf{F}_{ck^{(1)}_2}(\cdot), \mathsf{F}_{ck^{(1)}_3}(\cdot))$ and $\mathsf{C}^{(2)} = (\mathsf{F}_{ck^{(2)}_1}(\cdot), \mathsf{F}_{ck^{(2)}_2}(\cdot), \mathsf{F}_{ck^{(2)}_3}(\cdot))$ of $\boldsymbol{k} = (k_1, k_2, k_3)$, embedded with messages

$$msg^{(1)} = 101 \quad \text{and} \quad msg^{(2)} = 110$$

respectively. First, we have $ck^{(1)}_1 = ck^{(2)}_1$ since both of them are generated by puncturing k_1 on $x^*_{1,1}$ (we derive the randomness for the puncturing algorithm from $\boldsymbol{k}$). Thus, by the *single key* security of the underlying (constraint-hiding) constrained PRF, the adversary is not able to modify the mark 1 in k_1. However, as $ck^{(1)}_2$ and $ck^{(2)}_2$ (also, $ck^{(1)}_3$ and $ck^{(2)}_3$) are generated by puncturing k_2 (resp. k_3) on different points, we have $ck^{(1)}_2 \neq ck^{(2)}_2$ (resp. $ck^{(1)}_3 \neq ck^{(2)}_3$). So, the adversary is able to obtain different constrained versions of k_2 and k_3, and thus it has the capability to remove or modify the marks in them. As a result, when extracting a circuit produced by the adversary, the extraction algorithm may obtain a message in $\{1\} \times \{?, 0, 1\} \times \{?, 0, 1\}$,[5] which contains new messages

[5] We use ? to denote that no mark is detected for this position.

such as 111 and 100. That is, the adversary still has the ability to modify the embedded messages even if it fails at some position.

A secure solution using fingerprinting code. To solve this problem, we employ a fingerprinting code to amplify the robustness of our initial construction, from extracting some bits of the embedded messages to extracting one of the embedded messages, when dealing with adversarially-generated circuits. A fingerprinting code scheme consists of two algorithms, namely, the generation algorithm and the decoding algorithm. The generation algorithm generates a codebook and a trapdoor, where the codebook assigns a unique codeword to each message and the trapdoor is used for decoding. The decoding algorithm decodes a word (not necessarily in the codebook) using the trapdoor. Its security ensures that given a few codewords for messages in a specific set $\mathcal{C}$, no one could produce a word that is decoded to a message outside $\mathcal{C}$. This security is defined assuming the "marking assumption", where the adversary is not allowed to modify the bit at a position if all given codewords agree at this position. For example, if the given codewords are 101 and 110, then a word w such that $w[1] = 0$ is invalid.

Now, we integrate the fingerprinting code into our construction. More precisely, the marking algorithm first gets the codeword for the given message from the codebook. Then it embeds the codeword into the PRF keys via invoking the marking algorithm provided in our initial construction. Then, on input a circuit, the extraction algorithm first invokes the extraction algorithm of our initial construction. It replaces all "?" in the returned string with "0", and decodes the string using the decoding algorithm of the fingerprinting code. The marking assumption is guaranteed by security of our initial construction and thus the new extraction algorithm can succeed in extracting one of the embedded messages.

It is worth noting that our construction does not rely on concrete properties of the underlying constrained PRF, thus it is safe to replace it with any secure watermarkable PRF. In other words, our idea can be seen as a compiler that compiles a single key secure watermarkable PRF into a collusion resistant one.

Achieving security with marking queries/public marking. We have shown how to achieve collusion resistant watermarkable PRF in a setting that the adversary is only allowed to obtain some "challenge circuits", which are produced by embedding messages in a set $\mathcal{C}$ into a random PRF key. However, in previous works, the adversary is always allowed to further learn marked circuits for its selected keys. The above solution is not secure with such marking queries. This is because the marking query will provide codewords of messages outside $\mathcal{C}$, which can help the adversary to alter the embedded messages in the challenge circuits.

We fix this issue by forcing the marking algorithm to use different codebooks for different keys. In particular, the marking algorithm will first generate a codebook and the associated trapdoor (using randomness derived from the input PRF key), and then produce the marked circuit with this fresh codebook. As the codewords acquired from the marking queries are from different codebooks, they will not help the adversary in modifying the embedded messages in the challenge circuits.

The next issue is how to send the trapdoor to the extraction algorithm. Here we need to guarantee that the extraction algorithm can always receive the correct trapdoor and that the trapdoor is hidden to the adversary[6]. Note that, however, the only communication channel between the marking algorithm and the extraction algorithm is the watermarked circuits, which can be arbitrarily modified by the adversary.

We complete this task by embedding an encryption of the trapdoor into a new watermarkable PRF. More precisely, the watermarking object now is $l+1$ single key secure watermarkable PRF, where each of the first l parts is embedded with one bit of the codeword and the last part is embedded with the ciphertext. For the same PRF key, we use the same randomness to generate the trapdoor and its encryption, thus single key security of a watermarkable PRF is sufficient to guarantee a reliable transmission of the trapdoor. Also, confidentiality of the trapdoor is guaranteed by security of the encryption scheme.

By applying above tweaks, security with marking queries of the new construction can be based on the security with marking queries of the underlying (single key secure) watermarkable PRF. Besides, we can also show that if the underlying watermarkable PRF is secure with public marking (i.e., the mark key is public), the new scheme also supports public marking.

Achieving security with extraction queries. Another desirable security property for watermarking schemes is security with extraction queries, which allows the adversary to learn what can be extracted from its generated circuits. Note that the extraction algorithm of our scheme consists of three steps. First, it extracts a word and a ciphertext from the marked circuit; then it decrypts the ciphertext to get the trapdoor; finally, it uses the trapdoor to retrieve the message from the word. Therefore, security with extraction queries of our scheme can be guaranteed if the underlying watermarkable PRF has security with extraction queries (achieved in [QWZ18, KW19]), the underlying encryption scheme has security with decryption queries (i.e., CCA-security, which is achieved by numerous previous works), and the underlying fingerprinting code has security with extraction queries. However, no fingerprinting code that is provable secure with extraction queries is known. To solve this problem, in this work, we construct the first fingerprinting code that is secure with extraction queries. We provide an overview of this construction later in this section.

Achieving unforgeability. One drawback of the current construction is that it cannot achieve unforgeability even if the underlying watermarking scheme is unforgeable. To see this, recall that given a marked circuit, which is a combination of $l+1$ marked circuits, the extraction algorithm will extract one bit from each of the first l circuits. The bit is set to be 1 if it gets 1 from the circuit and the bit is set to be 0 either if it gets 0 from the circuit or if it gets an unmarked symbol $\perp$. That is to say, the extraction algorithm could still output some message even if part of the circuit is unmarked. Thus, an adversary may break the unforgeability of our construction by replacing part of a marked circuit with a random

[6] This is because current fingerprinting code is not secure if the trapdoor is revealed.

circuit. The new circuit and the original marked circuit should behave differently on nearly all inputs, yet the extraction algorithm will probably extract some message from it.

We solve this problem by presenting a general approach to adding strong unforgeability to watermarkable PRFs.[7] Let G be a secure watermarkable PRF (without unforgeability). We show how to construct a secure watermarkable PRF with unforgeability from G. The construction employs a signature scheme and an encryption scheme. In more detail, the revised marking algorithm first signs on the PRF key and encrypts the PRF key and the signature. Then, it embeds the ciphertext as well as the message into the PRF key using the original marking algorithm of G. The new extraction algorithm will first extract the ciphertext and the message from the circuit. Then, it decrypts the ciphertext to obtain the PRF key k and the signature. Next, the extraction algorithm checks if the signature is valid and if the circuit behaves almost identically to G_k. It outputs the message only if both checks are passed, and it outputs $\perp$ otherwise.

Unforgeability of the watermarking scheme comes from unforgeability of the signature scheme. In particular, due to the unforgeability of the signature scheme, the adversary is not able to generate valid signatures for a new PRF key. Therefore, if the adversary wishes to create a circuit that can pass the extraction algorithm, the circuit must be close to one of previously marked PRF keys, which is exactly what the unforgeability requires. We stress that the claim holds even if the extraction key of the scheme is revealed. Thus, we provide the first watermarkable PRF achieving unforgeability with public extraction from standard assumption (yet, it does not have unremovability with public extraction).

Next, we argue why the new construction still has unremovability. There are two main concerns. Firstly, the original PRF key is included in the marked circuit, but as only an encryption of the key is embedded, this will not leak additional information to the adversary.[8] Secondly, an additional check is performed in the extraction algorithm to test if the circuit preserves the functionality of the original key. Since the adversary (for unremovability) is not allowed to significantly change the functionality of the challenge circuit(s), its submitted circuit should pass the check.

Putting it all together. Piecing together all ideas and techniques proposed above, we obtain a generic construction of collusion resistant watermarkable PRF from any single key secure watermarkable PRF. The construction preserves the security with marking queries/public marking/extraction queries of the underlying single key secure watermarking scheme. Also, it achieves unforgeability for free. We provide a detailed description of the construction in Sec. 4.

[7] The technique only works for watermarkable PRFs with exponential message space, which is not achieved by our collusion resistant watermarkable PRF (due to the polynomially sized message space of the underlying fingerprinting code). Nonetheless, we can still apply it in our construction specifically since the underlying single key secure schemes do support exponential message space.

[8] This only holds when the decryption key of the encryption scheme is kept private, so, the upgrading does not preserve the unremovability in the public extraction setting.

Fingerprinting code secure with extraction queries. It remains to show how to construct a fingerprinting code that is secure with extraction queries. We start by briefly reviewing the well-known Boneh-Shaw code [BS95], which is widely used in cryptography.

Let N be the size of the message space and let L be a polynomial in security parameter and N. The code generation algorithm first samples N disjoint subsets $\mathcal{P}_1, \ldots, \mathcal{P}_N$ of $[NL]$, where $|\mathcal{P}_i| = L$, and sets them as the trapdoor. Then it sets the codeword for a message $\mathtt{m} \in [N]$ to be an NL-bit binary string $\bar{w}_{\mathtt{m}}$, where $\bar{w}_{\mathtt{m}}[j] = 1$ iff $j \in \mathcal{P}_i$ for some $i \leq \mathtt{m}$. To decode a word w, the decoding algorithm sets $A_0 = 1$ and $A_{N+1} = 0$, then it computes $A_i = (\sum_{j \in \mathcal{P}_i} w[j])/L$ and outputs the first i s.t. $A_i - A_{i+1}$ is large.

To see security of the Boneh-Shaw code, considering a simple example where $N = 4$ and the adversary is given two codewords $\bar{w}_1$ and $\bar{w}_3$, let w be the word output by the adversary. Then, the decoding algorithm will not output a message outside $\{1, 3\}$ on input w, because:

1. For any $j \in \mathcal{P}_1$, $\bar{w}_1[j] = \bar{w}_3[j] = 1$ and for any $j \in \mathcal{P}_4$, $\bar{w}_1[j] = \bar{w}_3[j] = 0$, then from the marking assumption, the adversary is not allowed to modify the bit at a position in $\mathcal{P}_1$ and $\mathcal{P}_4$. Thus, we still have $A_1 = 1$ and $A_4 = 0$. Therefore, the decoding algorithm will not output 0 or 4.
2. For bits at positions in $\mathcal{P}_2$ and $\mathcal{P}_3$, the adversary can modify them arbitrarily. But, since the trapdoor is kept *hidden* to the adversary, it cannot distinguish positions in $\mathcal{P}_2$ and that in $\mathcal{P}_3$. So, the adversary cannot make $A_2 - A_3$ large and thus the decoding algorithm will not output 2.

However, if the adversary is allowed to make queries to an extraction oracle, it can learn some information about the trapdoor from each query. Thus, the second claim above will be invalidated in this case.[9]

We deal with this issue by using *part* of the trapdoor in each invocation of the decoding algorithm. In particular, the decoding algorithm randomly picks a fixed size subset $\mathcal{S}_i \subseteq \mathcal{P}_i$ for $i \in [N]$. Then it computes $A'_i = (\sum_{j \in \mathcal{S}_i} w[j])/|\mathcal{S}_i|$ and finds the large gap between A'_i and A'_{i+1}. The fraction A'_i can be viewed as an estimation of A_i and the two numbers are close, so the modification here will not compromise security of the Boneh-Shaw code.

To see why the revised decoding algorithm can provide security with extraction queries, let $\mathcal{S}^*_1, \ldots, \mathcal{S}^*_N$ be the partial trapdoor used when decoding a word w from the adversary, who has seen codewords for messages in a set $\mathcal{C}$. Due to the security of the original Boneh-Shaw code, the decoding algorithm should output a message in $\mathcal{C}$, if all $\mathcal{S}_i$ used in previous extraction queries are sampled from $\mathcal{P}_i - \mathcal{S}^*_i$. Thus, it is sufficient to show that the output of the extraction oracle will not change (significantly) if the decoding algorithm uses a random subset of $\mathcal{P}_i - \mathcal{S}^*_i$ instead of a random subset of $\mathcal{P}_i$. Unfortunately, it seems that there is a non-negligible gap between the oracle outputs in these two cases and the conventional statistical distance is not applicable here to bound the adversary's advantage. To overcome this hurdle, we use the Rényi divergence to measure

[9] In fact, the adversary could find positions in $\mathcal{P}_2$ via altering bits of $\bar{w}_3$ one by one and observe when the extraction oracle outputs 1 instead of 3.

the distribution closeness and limit the number of extraction queries from the adversary. See Sec. 3 for a more detailed description of our construction.

1.3 Related Works

The notion of fingerprinting code is first studied in [Wag83, BMP85]. Considering the adversary's ability in altering the codewords, many different models for fingerprinting code are studied. In this work, we consider the model presented in [BS98]. Boneh and Shaw [BS98] construct the first fingerprinting code that is secure in this model. Then, in [Tar03], Tardos presents a shorter code and shows that the code length is optimal in the asymptotic sense. Some subsequent works (see e.g., [NFH$^+$09, AT09, LdW14] and references therein) aim at improving the concrete efficiency of the scheme. However, to the best of our knowledge, no work has considered an adversary that can ask for the decoding of its created words.

One important application of fingerprinting code is to build traitor tracing schemes [CFN94], which aims at tracing secret key leakers in a broadcast encryption setting. The notion of traitor tracing is somewhat similar to the notion of collusion resistant watermarking. But our construction has several differences from previous fingerprinting code based traitor tracing schemes [BN08]. First, we embed each bit of the codeword into the underlying single key watermarkable PRF directly, while in [BN08], codewords are used to select secret keys for users. Besides, we need to additionally send the trapdoor from the marking algorithm to the extraction algorithm. In addition, we require a stronger fingerprinting code that has adaptive security with extraction queries, and provide an instantiation.

2 Notations

Let s be a string, we use $|s|$ to denote the length of s. For integers $a \leq |s|$, we use $s[a]$ to denote the i-th character of s and for integers $a \leq b \leq |s|$, we use $s[a:b]$ to denote the substring $(s[a], s[a+1], \ldots, s[b])$. Let $\mathcal{S}$ be a finite set, we use $|\mathcal{S}|$ to denote the size of $\mathcal{S}$, and use $s \xleftarrow{\$} \mathcal{S}$ to denote sampling an element s uniformly from set $\mathcal{S}$. Let $\mathcal{D}$ be a distribution, we use $d \leftarrow \mathcal{D}$ to denote sampling d according to $\mathcal{D}$ and use $Supp(\mathcal{D})$ to denote the support of $\mathcal{D}$.

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 and use $[b]$ to denote all integers from 1 to b. For natural numbers $a \leq b$, we use $\binom{b}{a}$ to denote the binomial coefficient, i.e., $\binom{b}{a} = \frac{b \cdot (b-1) \cdot \ldots \cdot (b-a+1)}{a \cdot (a-1) \cdot \ldots \cdot 1}$.

For more background knowledge and definitions of cryptographic primitives employed, we refer the readers to the full version of this paper.

3 Fingerprinting Code with Enhanced Security

3.1 The Definition

In this section, we provide the definition of fingerprinting code. Compared to previous definitions [BS95, Tar03, BN08], we require a stronger security, where the adversary is allowed to 1) make queries to an extraction oracle that outputs the decoding of a given word and 2) make challenge oracle queries adaptively.

Definition 3.1 (Fingerprinting Code). *A fingerprinting code* $\mathsf{FC} = (\mathtt{Gen}, \mathtt{Dec})$ *with message space* $[1, N]$ *and code length* l *consists of the following algorithms:*

- $\mathtt{Gen}(1^\lambda) \to (td, \Gamma = (\bar{w}_{\mathtt{m}})_{\mathtt{m} \in [N]})$: *On input the security parameter* λ*, the code generation algorithm outputs the trapdoor* td *and* N *codewords* $\bar{w}_1, \ldots \bar{w}_N$ *(for messages* $1, \ldots, N$ *respectively) in* $\{0,1\}^l$*.*
- $\mathtt{Dec}(td, w) \to \mathtt{m}$: *On input the trapdoor* td *and a word* $w \in \{0,1\}^l$ *(*w *is not necessarily in* Γ*), the decoding algorithm outputs a message* $\mathtt{m} \in [1, N] \cup \{\bot\}$*.*

The correctness property requires that the the decoding algorithm will decode codewords in Γ correctly.

Definition 3.2 (Correctness). *Let* $(td, (\bar{w}_{\mathtt{m}})_{\mathtt{m} \in [N]}) \leftarrow \mathtt{Gen}(1^\lambda)$*, then for any* $\mathtt{m}$*, we have:*

$$\Pr[\mathtt{Dec}(td, \bar{w}_{\mathtt{m}}) \neq \mathtt{m}] = 0$$

The security property requires that given a few codewords $\{\bar{w}_{\mathtt{m}}\}_{\mathtt{m} \in \mathcal{C}^*} \subseteq \Gamma$ for messages in a set $\mathcal{C}^*$, no adversary can generate a "feasible" word that decodes to a new message outside $\mathcal{C}^*$. Here, we say that a word w is **feasible** if

$$\forall j \in [l], \exists \mathtt{m} \in \mathcal{C}^*, \bar{w}_{\mathtt{m}}[j] = w[j]$$

In this work, we consider a strong security, where the adversary is allowed to learn the decoding of q feasible words for an a priori bounded q. Also, we allow the adversary to make challenge oracle queries adaptively, i.e., it can request codewords for its selected messages after viewing some codewords and the decoding of some words.

Definition 3.3 (Security with q Extraction Queries). *A fingerprinting code is secure with* q *extraction queries if for all polynomial-time (PPT) adversaries* $\mathcal{A}$*, we have* $\Pr[\mathtt{Expt}_{\mathcal{A},q}(\lambda) = 1] \leq negl(\lambda)$*, where we define the experiment* $\mathtt{Expt}$ *as follows:*

1. *The challenger samples* $(td, (\bar{w}_{\mathtt{m}})_{\mathtt{m} \in [N]}) \leftarrow \mathtt{Gen}(1^\lambda)$ *and initializes the set* $\mathcal{C}^* = \emptyset$*.*
2. *Then, the adversary is allowed to make a priori unbounded number of queries to the challenge oracle and make up to* q *queries to the extraction oracle, which are defined as follows:*
   - ***Challenge Oracle.*** *On input a message* $\mathtt{m} \in [1, N]$*, the challenger returns* $\bar{w}_{\mathtt{m}}$ *to the adversary and sets* $\mathcal{C}^* = \mathcal{C}^* \cup \{\mathtt{m}\}$*.*
   - ***Extraction Oracle.*** *On input a word* w*, the challenger does not return anything to* $\mathcal{A}$ *if* w *is not feasible (according to current* $\mathcal{C}^*$*). Otherwise, it computes* $\mathtt{m} \leftarrow \mathtt{Dec}(td, w)$*. The challenger returns* $\mathtt{m}$ *to* $\mathcal{A}$ *if* $\mathtt{m} \in \mathcal{C}^*$*. Otherwise, the experiment aborts and outputs 1.*
3. *The experiment outputs 0 if it does not abort in Step 2.*

3.2 The Construction

In this section, we present our construction of fingerprinting code that has adaptive security with extraction queries.

Let λ be the security parameter. Let N, L, l, q be positive integers that are polynomial in λ and satisfy $l = 8\lambda(N+1)^2$, $L = 8l - 4 + 4l^2 Nq$. Let $\theta = 1/(2(N+1))$. Let $\mathfrak{S} = \{\mathcal{S} \subseteq [1, L] : |\mathcal{S}| = l\}$ be the set of all subsets of $[1, L]$ that contain l elements.

We construct the fingerprinting code $\mathsf{FC} = (\texttt{Gen}, \texttt{Dec})$ with message space $[1, N]$ and code length NL as follows:

- $\texttt{Gen}$. On input a security parameter λ, the code generation algorithm first samples a random permutation $\texttt{P}$ over $[NL]$. Then for $\texttt{m} \in [1, N]$, and $h \in [NL]$, it sets

$$\bar{w}_{\texttt{m}}[h] = \begin{cases} 1 & \text{if } \lceil \texttt{P}(h)/L \rceil \leq \texttt{m} \\ 0 & \text{otherwise} \end{cases}$$

  Finally, it outputs the trapdoor $td = \texttt{P}$ and the codewords $(\bar{w}_{\texttt{m}})_{\texttt{m} \in [N]}$.
- $\texttt{Dec}$. On input the trapdoor $td = \texttt{P}$ and a word $w \in \{0,1\}^{NL}$, the decoding algorithm proceeds as follows:
  1. For $i \in [N]$:
     (a) Sample $\mathcal{S}_i \xleftarrow{\$} \mathfrak{S}$
     (b) $A_i = 0$
     (c) For $j \in \mathcal{S}_i$:
        i. $A_i = A_i + w[\texttt{P}^{-1}((i-1)L + j)]$
     (d) If $\frac{A_i}{l} \leq \frac{3}{4} - i\theta$:
        i. If $i = 1$: Output $\perp$
        ii. Output $i - 1$
  2. Output N

Theorem 3.1. FC *is a secure fingerprinting code that has correctness and adaptive security with q extraction queries.*

We give proof of Theorem 3.1 in the full version.

4 Collusion Resistant Watermarkable PRF

4.1 The Definition

In this section, we provide the definition of watermarkable PRF, which is adapted and generalized from definitions in previous works [CHN$^+$16, BLW17, KW17, QWZ18, KW19, YAL$^+$19].

Definition 4.1 (Watermarkable PRFs). *A watermarkable PRF* $\mathsf{WPRF} = (\texttt{Setup}, \texttt{KeyGen}, \texttt{Eval}, \texttt{Mark}, \texttt{Extract})$ *with key space $\mathcal{K}$, input space $\{0,1\}^n$, output space $\{0,1\}^m$, and message space $\mathcal{M}$ consists of the following algorithms:*

- $\texttt{Setup}(1^\lambda) \to (PP, MK, EK)$: *On input the security parameter λ, the setup algorithm outputs the public parameter PP, the mark key MK and the extraction key EK.*
- $\texttt{KeyGen}(PP) \to k$: *On input the public parameter PP, the key generation algorithm outputs a PRF key $k \in \mathcal{K}$.*
- $\texttt{Eval}(PP, k, x) \to y$: *On input the public parameter PP, a PRF key $k \in \mathcal{K}$, and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs an output $y \in \{0,1\}^m$.*
- $\texttt{Mark}(PP, MK, k, msg) \to \texttt{C}$: *On input the public parameter PP, the mark key MK, a PRF key $k \in \mathcal{K}$, and a message $msg \in \mathcal{M}$, the marking algorithm outputs a marked circuit $\texttt{C} : \{0,1\}^n \to \{0,1\}^m$.*
- $\texttt{Extract}(PP, EK, \texttt{C}) \to msg$: *On input the public parameter PP, the extraction key EK, and a circuit $\texttt{C}$, the extraction algorithm outputs a message $m \in \mathcal{M} \cup \{\perp\}$, 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 4.2 (Functionality Preserving). *For any $msg \in \mathcal{M}$, let $(PP, MK, EK) \leftarrow \texttt{Setup}(1^\lambda)$, $k \leftarrow \texttt{KeyGen}(PP)$, $\texttt{C} \leftarrow \texttt{Mark}(PP, MK, k, msg)$, $x \xleftarrow{\$} \{0,1\}^n$, 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.

Definition 4.3 (Extraction Correctness). *For any $msg \in \mathcal{M}$, let $(PP, MK, EK) \leftarrow \texttt{Setup}(1^\lambda)$, $k \leftarrow \texttt{KeyGen}(PP)$, $\texttt{C} \leftarrow \texttt{Mark}(PP, MK, k, msg)$, then we have* $\Pr[\texttt{Extract}(PP, EK, \texttt{C}) \neq msg] \leq negl(\lambda)$.

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

Definition 4.4 (Watermarking Meaningfulness). *For any circuit $\texttt{C} : \{0,1\}^n \to \{0,1\}^m$, let $(PP, MK, EK) \leftarrow \texttt{Setup}(1^\lambda)$, then we have:*

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

Remark 4.1. In Definition 4.2 and Definition 4.3, correctness properties are defined for honestly-generated PRF keys only. A stronger notion of correctness consider adversarially-chosen keys, where k is chosen by the adversary. See [KW17, KW19] for more detailed discussions on different notions of correctness.

Pseudorandomness. The pseudorandomness property of a watermarkable PRF is twofold. First, it requires that the watermarkable PRF should be pseudorandom against an external adversary.

Definition 4.5 (Pseudorandomness). *Let* $(PP, MK, EK) \leftarrow \texttt{Setup}(1^\lambda)$*,* $k \leftarrow \texttt{KeyGen}(PP)$*, 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* $\texttt{Eval}(PP, k, x)$*, 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:*

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

Moreover, the watermarkable PRF should be (weak) pseudorandom against the watermarking authority, who holds the mark key and the extraction key.

Definition 4.6 (Pseudorandomness against the Watermarking Authority). *Let* $(PP, MK, EK) \leftarrow \texttt{Setup}(1^\lambda)$*,* $k \leftarrow \texttt{KeyGen}(PP)$*, 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* $\texttt{Eval}(PP, k, x)$*, 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:*

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

Definition 4.7 (Weak Pseudorandomness against the Watermarking Authority). *Let* $(PP, MK, EK) \leftarrow \texttt{Setup}(1^\lambda)$*,* $k \leftarrow \texttt{KeyGen}(PP)$*, and* f *be a random function from* $\{0,1\}^n$ *to* $\{0,1\}^m$*. Also, let* $\mathcal{O}_1$ *be an oracle that samples* $x \xleftarrow{\$} \{0,1\}^n$ *and returns* $(x, \texttt{Eval}(PP, k, x))$ *on each query, and let* $\mathcal{O}_2$ *be an oracle that samples* $x \xleftarrow{\$} \{0,1\}^n$ *and returns* $(x, f(x))$ *on each query. Then for all PPT adversary* $\mathcal{A}$*, we have:*

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

Unremovability. This is the main security requirement for a watermarking scheme. Roughly, it requires that the adversary is not able to remove or modify the messages embedded in a random PRF key without significantly changing the functionality.

Definition 4.8 (ϵ-Unremovability). *A watermarkable PRF is* ϵ*-unremovable if for all PPT and* ϵ*-unremoving-admissible adversaries* $\mathcal{A}$*, we have* $\Pr[\texttt{ExptUR}_{\mathcal{A}}(\lambda) = 1] \leq negl(\lambda)$*, where we define the experiment* $\texttt{ExptUR}$ *as follows:*

1. *The challenger samples* $(PP, MK, EK) \leftarrow \texttt{Setup}(1^\lambda)$ *and returns* PP *to* $\mathcal{A}$*. Also, it samples a challenge key* $k^* \leftarrow \texttt{KeyGen}(PP)$*, which is used in answering the adversary's challenge oracle queries.*
2. *Then,* $\mathcal{A}$ *is given access to the 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.*
   - ***Extraction Key Oracle.*** *The extraction key oracle returns* EK *to the adversary.*

- ***Marking Oracle.*** *On input a PRF key $k \in \mathcal{K}$ and a message $msg \in \mathcal{M}$, the marking oracle returns a circuit $\mathtt{C} \leftarrow \mathtt{Mark}(PP, MK, k, msg)$.*
- ***Extraction Oracle.*** *On input a circuit $\mathtt{C}$, the extraction oracle returns a message $msg \leftarrow \mathtt{Extract}(PP, EK, \mathtt{C})$.*
- ***Challenge Oracle.*** *On input a message msg, the challenge oracle returns a circuit $\mathtt{C}^* \leftarrow \mathtt{Mark}(PP, MK, k^*, msg)$ to the adversary.*

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

We say that an adversary $\mathcal{A}$ is ϵ-unremoving-admissible *if there exists circuit $\mathtt{C}^* \in \mathsf{C}^*$ that $|\{x \in \{0,1\}^n : \mathtt{C}^*(x) \neq \tilde{\mathtt{C}}(x)\}| \leq \epsilon \cdot 2^n$.*

We can get different levels of unremovability by restricting the adversary's ability in querying oracles. In a nutshell, we write unremovability as $\mathcal{C}$-$(\mathcal{M}, \mathcal{E})$-$\epsilon$-unremovability, where $\mathcal{C} \in \{$single key, bounded collusion resistant, fully collusion resistant$\}$, $\mathcal{M} \in \{-, \mathrm{MO}, \mathrm{PM}\}$, and $\mathcal{E} \in \{-, \text{bounded EO}, \mathrm{EO}, \mathrm{PE}\}$. In more detail, the security notions are organized along the following three dimensions:

- **Ability to Query the Challenge Oracle.** The unremovability can be defined against an adversary that can:
  - make only one query to the challenge oracle (single key).
  - make queries to the challenge oracle for a priori bounded number of times (bounded collusion resistant).
  - make queries to the challenge oracle for a priori unbounded number of times (fully collusion resistant).
- **Ability in Obtaining Information about MK.** The unremovability can be defined against an adversary that can:
  - make query to neither the mark key oracle nor the marking oracle $(-)$.
  - make a priori unbounded number of queries to the marking oracle but make no query to the mark key oracle (MO).
  - make query to the mark key oracle (PM).
- **Ability in Obtaining Information about EK.** The unremovability can be defined against an adversary that can:
  - make query to neither the extraction key oracle nor the extraction oracle $(-)$.
  - make at most q queries to the extraction oracle but make no query to the extraction key oracle, where q is a priori bounded (bounded EO or q-EO).
  - make a priori unbounded number of queries to the extraction oracle but make no query to the extraction key oracle (EO).
  - make query to the extraction key oracle (PE).

Remark 4.2. In our definition of collusion resistant unremovability, the adversary is allowed to make challenge oracle queries adaptively. Such adaptive security is not defined (and achieved) in previous works about collusion resistant watermarkable PRF [YAL+19].

Unforgeability. This property is dual to the unremovability. Roughly, it prevents one from embedding messages to PRF keys without the mark key.

Definition 4.9 (δ-Unforgeability). *A watermarkable PRF is δ-unforgeable if for all PPT and δ-unforging-admissible adversaries $\mathcal{A}$, we have* $\Pr[\texttt{ExptUF}_{\mathcal{A}}(\lambda) = 1] \leq negl(\lambda)$*, where we define the experiment* $\texttt{ExptUF}$ *as follows:*

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

*Here, an adversary $\mathcal{A}$ is δ-*unforging-admissible *if for every circuit $\texttt{C}_i$ returned by the marking oracle, $|\{x \in \{0,1\}^n : \texttt{C}_i(x) \neq \tilde{\texttt{C}}(x)\}| \geq \delta \cdot 2^n$.*[10]

We can get different levels of unforgeability by restricting the adversary's ability in querying oracles. In a nutshell, we write unforgeability as $(\mathscr{M}, \mathscr{E})$-$\delta$-unforgeability, where $\mathscr{M} \in \{-, \text{MO}\}$, and $\mathscr{E} \in \{-, \text{EO}, \text{PE}\}$. In more detail, the security notions are organized along the following two dimensions:

- **Ability in Obtaining Information about MK.** The unforgeability can be defined against an adversary that can:
  - make no query to the marking oracle ($-$).
  - make a priori unbounded number of queries to the marking oracle (MO).
- **Ability in Obtaining Information about EK.** The unforgeability can be defined against an adversary that can:
  - make query to neither the extraction key oracle nor the extraction oracle ($-$).
  - make a priori unbounded number of queries to the extraction oracle but make no query to the extraction key oracle (EO).
  - make query to the extraction key oracle (PE).

4.2 The Construction

In this section, we show our main construction, which upgrades single key secure watermarkable PRF families into fully collusion resistant ones.

Let λ be the security parameter. Let n, m, N, l, s, κ, q be positive integers that are polynomial in λ. Let $\epsilon, \epsilon', \bar{\epsilon}$ be positive real values s.t. $1/\bar{\epsilon}$ is polynomial in λ, $\bar{\epsilon} = (1 + 1/\lambda) \cdot \epsilon, \epsilon' = (1 + 2/\lambda) \cdot \epsilon$. Also, let $t = \lambda^3/\epsilon$.

Our construction is built on the following building blocks:

[10] An alternative definition of δ-unforging-admissibility, which is used in [KW17], additionally requires that for every PRF key k_i submitted to the marking oracle, $|\{x \in \{0,1\}^n : \texttt{C}_i(x) \neq \texttt{Eval}(PP, k_i, x)\}| \geq \delta \cdot 2^n$.

- A watermarkable PRF family $\mathsf{WPRF}_0 = (\mathsf{WPRF}_0.\mathtt{Setup}, \mathsf{WPRF}_0.\mathtt{KeyGen}, \mathsf{WPRF}_0.\mathtt{Eval}, \mathsf{WPRF}_0.\mathtt{Mark}, \mathsf{WPRF}_0.\mathtt{Extract})$ with input space $\{0,1\}^n$, output space $\{0,1\}^m$, and message space $\{0,1\}^\kappa$. Also, we use $\mathcal{R}_0$ and $\mathcal{R}'_0$ to denote the randomness space for the algorithm $\mathsf{WPRF}_0.\mathtt{KeyGen}$ and the algorithm $\mathsf{WPRF}_0.\mathtt{Mark}$ respectively.
- A fingerprinting code $\mathsf{FC} = (\mathsf{FC}.\mathtt{Gen}, \mathsf{FC}.\mathtt{Dec})$ with message space $[1, N]$ and code length l. Also, we use $\mathcal{T}$ and $\mathcal{R}_{\mathsf{FC}}$ to denote the key space (i.e., the set of all trapdoors for FC) and the randomness space for the algorithm $\mathsf{FC}.\mathtt{Gen}$ respectively.
- A signature scheme $\mathsf{SIG} = (\mathsf{SIG}.\mathtt{KeyGen}, \mathsf{SIG}.\mathtt{Sign}, \mathsf{SIG}.\mathtt{Verify})$ with message space $\{0,1\}^\lambda$, signature space $\{0,1\}^s$ and signing randomness space $\mathcal{R}_{\mathsf{SIG}}$.
- A PKE scheme $\mathsf{PKE} = (\mathsf{PKE}.\mathtt{KeyGen}, \mathsf{PKE}.\mathtt{Enc}, \mathsf{PKE}.\mathtt{Dec})$ with message space $\mathcal{T} \times \{0,1\}^\lambda \times \{0,1\}^s$, ciphertext space $\{0,1\}^\kappa$, and encryption randomness space $\mathcal{R}_{\mathsf{PKE}}$.
- Pseudorandom generators:

$$\mathtt{G} : \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^\lambda \times \mathcal{R}_{\mathsf{FC}} \times \mathcal{R}_{\mathsf{SIG}} \times \mathcal{R}_{\mathsf{PKE}}$$
$$\mathtt{G}' : \{0,1\}^\lambda \to \mathcal{R}_0^{l+1} \qquad \mathtt{G}'' : \{0,1\}^\lambda \to \mathcal{R}_0'^{2l+1}$$

- A pseudorandom function family $\mathsf{F} = (\mathsf{F}.\mathtt{KeyGen}, \mathsf{F}.\mathtt{Eval})$ with input space $\mathcal{R}_0'^{2l+1}$ and output space $\mathcal{R}_0'^{2l+1}$.

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

- $\mathtt{Setup}$. On input a security parameter λ, the setup algorithm first generates $(PP_0, MK_0, EK_0) \leftarrow \mathsf{WPRF}_0.\mathtt{Setup}(1^\lambda)$, $(VK, SK) \leftarrow \mathsf{SIG}.\mathtt{KeyGen}(1^\lambda)$, $(PK, DK) \leftarrow \mathsf{PKE}.\mathtt{KeyGen}(1^\lambda)$, and $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then, it outputs the public parameter $PP = (PP_0, VK, PK)$, the mark key $MK = (MK_0, SK, K)$, and the extraction key $EK = (EK_0, DK)$.
- $\mathtt{KeyGen}$. On input the public parameter PP, the key generation algorithm outputs the PRF key $s \xleftarrow{\$} \{0,1\}^\lambda$.
- $\mathtt{Eval}$. On input the public parameter $PP = (PP_0, VK, PK)$, a PRF key $s \in \{0,1\}^\lambda$ and an input $x \in \{0,1\}^n$, the evaluation algorithm proceeds as follows:
  1. $(\check{r}, \hat{r}, R_{\mathsf{FC}}, R_{\mathsf{SIG}}, R_{\mathsf{PKE}}) = \mathtt{G}(s)$.
  2. $(r_0, r_1, \ldots, r_l) = \mathtt{G}'(\check{r})$.
  3. For $i \in [0, l]$, $k_i = \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0; r_i)$.
  4. Output $(\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i, x))_{i \in [0,l]}$.
- $\mathtt{Mark}$. On input the public parameter $PP = (PP_0, VK, PK)$, the mark key $MK = (MK_0, SK, K)$, a PRF key $s \in \{0,1\}^\lambda$ and a message $msg \in [1, N]$, the marking algorithm proceeds as follows:
  1. $(\check{r}, \hat{r}, R_{\mathsf{FC}}, R_{\mathsf{SIG}}, R_{\mathsf{PKE}}) = \mathtt{G}(s)$.
  2. $(r_0, r_1, \ldots, r_l) = \mathtt{G}'(\check{r})$.
  3. For $i \in [0, l]$, $k_i = \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0; r_i)$.

4. $(td, (\bar{w}_i)_{i\in[N]}) = \mathsf{FC}.\mathtt{Gen}(1^\lambda; R_{\mathsf{FC}})$.
5. $\sigma = \mathsf{SIG}.\mathtt{Sign}(SK, \check{r}; R_{\mathsf{SIG}})$.
6. $ct = \mathsf{PKE}.\mathtt{Enc}(PK, td\|\check{r}\|\sigma; R_{\mathsf{PKE}})$.
7. $(r'_0, (r'_{i,\iota})_{i\in[l],\iota\in\{0,1\}}) = \mathsf{F}.\mathtt{Eval}(K, \mathtt{G}''(\hat{r}))$.
8. $\mathtt{W}_0 = \mathsf{WPRF}_0.\mathtt{Mark}(PP_0, MK_0, k_0, ct; r'_0)$.
9. For $i \in [l]$:
   (a) $b_i = \bar{w}_{msg}[i]$.
   (b) $\mathtt{W}_i = \mathsf{WPRF}_0.\mathtt{Mark}(PP_0, MK_0, k_i, b_i; r'_{i,b_i})$.
10. Outputs a circuit $\mathtt{C} : \{0,1\}^n \to \{0,1\}^{(l+1)m}$ s.t. $\mathtt{C}(x) = (\mathtt{W}_i(x))_{i\in[0,l]}$.

- $\mathtt{Extract}$. On input the public parameter $PP = (PP_0, VK, PK)$, the extraction key $EK = (EK_0, DK)$, and a circuit $\mathtt{C}$, the extraction algorithm proceeds as follows:
  1. Set the circuit $\mathtt{W}_0 : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_0(x) = \mathtt{C}(x)[1 : m]$.
  2. $ct = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_0)$.
  3. If $ct = \perp$, **output** $\perp$.
  4. $(td\|\check{r}\|\sigma) = \mathsf{PKE}.\mathtt{Dec}(DK, ct)$.
  5. If $(td\|\check{r}\|\sigma) = \perp$, **output** $\perp$.
  6. If $\mathsf{SIG}.\mathtt{Verify}(VK, \check{r}, \sigma) = 0$, **output** $\perp$.
  7. $(r_0, r_1, \ldots, r_l) = \mathtt{G}'(\check{r})$.
  8. For $i \in [0, l]$, $k_i = \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0; r_i)$.
  9. $A = 0$.
  10. For $j \in [t]$:
      (a) Sample $x \xleftarrow{\$} \{0,1\}^n$.
      (b) If $\mathtt{C}(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i, x))_{i\in[0,l]}$, $A = A + 1$.
  11. If $A > t \cdot \bar{\epsilon}$, **output** $\perp$.
  12. For $i \in [l]$:
      (a) $a = im + 1$, $b = (i + 1)m$.
      (b) Set the circuit $\mathtt{W}_i : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_i(x) = \mathtt{C}(x)[a : b]$.
      (c) $w[i] = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_i)$.
      (d) If $w[i] \notin [0, 1]$, $w[i] = 0$.
  13. $msg \leftarrow \mathsf{FC}.\mathtt{Dec}(td, w)$.
  14. **Output** msg.

Theorem 4.1. *If* WPRF_0 *is a single key secure watermarkable PRF family,* FC *is a secure fingerprinting code that is adaptively secure with* $q + 1$ *extraction queries as defined in Sec. 3,* PKE *is a CCA secure PKE scheme,* SIG *is a secure signature scheme,* $\mathtt{G}, \mathtt{G}', \mathtt{G}''$ *are secure pseudorandom generators, and* F *is secure pseudorandom function, then* WPRF *is a secure watermarkable PRF family with collusion resistant security. In particular:*

- *If* WPRF_0 *has (weak) pseudorandomness against the watermarking authority, then* WPRF *also has (weak) pseudorandomness against the watermarking authority.*
- *If* WPRF_0 *is single key-*$(\mathcal{M}, \mathcal{E})$*-*$\epsilon'$*-unremovable, then* WPRF *is fully collusion resistant-*$(\mathcal{M}, \mathcal{E})$*-*$\epsilon$*-unremovable, where* $\mathcal{M} \in \{MO, PM\}$*, and* $\mathcal{E} \in \{-,$ *bounded EO*$\}$*. In more detail, if* WPRF_0 *is single key-*$(\mathcal{M}, (l + 1)q\text{-}EO)$*-*$\epsilon'$*-unremovable, then* WPRF *is fully collusion resistant-*$(\mathcal{M}, q\text{-}EO)$*-*$\epsilon$*-unremovable.*

- WPRF *is* (MO, PE)-ϵ'-*unforgeable.*

We present proof of Theorem 4.1 later in this section, which includes proof of the correctness and pseudorandomness (Sec. 4.4), the unremoveability (Sec. 4.5), and the unforgeability (Sec. 4.6) of WPRF.

4.3 The Instantiations

In this section, we show how to instantiate our construction via employing existing watermarkable PRFs from standard assumptions [KW17, QWZ18, KW19]. Note that, all of them can be instantiated from some standard lattice assumptions, which can be further reduced to the worst-case hardness of appropriately parameterized GapSVP problem. Therefore, the watermarking schemes provided in this work also rely on the worst-case hardness of the GapSVP problem.

Instantiating from [KW17]. The scheme in [KW17] can achieve a single key-$(\mathrm{MO}, -)$-ϵ'-unremovability and a $(\mathrm{MO}, -)$-δ'-unforgeability, where ϵ' is negligible in λ and $\delta' = 1/poly(\lambda)$. Besides, the scheme has pseudorandomness against the watermarking authority.

Unfortunately, the scheme can not be used in our general construction directly. This is because in our construction, ϵ' is required to be significantly larger than $\bar{\epsilon}$, where $1/\bar{\epsilon} = poly(\lambda)$. Nonetheless, the requirement (i.e., $\epsilon' - \bar{\epsilon}$ is large) is only desired when proving unremovability against an adversary that can query the extraction oracle. Since the scheme in [KW17] does not achieve security with extraction queries, we do not need to argue it during the upgrading. So, we can still instantiate WPRF_0 with the scheme. Formally, we have:

Corollary 4.1. *Assuming the worst-case hardness of appropriately parameterized GapSVP problem, there exist watermarkable PRF families with fully collusion resistant-*$(MO, -)$*-*ϵ*-unremovability,* (MO, PE)*-*δ*-unforgeability, and pseudorandomness against the watermarking authority, where* $\epsilon = negl(\lambda)$ *and* $\delta = 1/poly(\lambda)$.

Instantiating from [QWZ18]. The scheme in [QWZ18] can achieve a single key-$(\mathrm{PM}, \mathrm{EO})$-$\epsilon'$-unremovability, where $\epsilon' = 1/2 - 1/poly(\lambda)$. When instantiating our construction with this scheme, we have:

Corollary 4.2. *Assuming the worst-case hardness of appropriately parameterized GapSVP problem, there exist watermarkable PRF families with fully collusion resistant-*$(PM, \text{bounded } EO)$*-*ϵ*-unremovability and* (MO, PE)*-*δ*-unforgeability, where* $\epsilon = \delta - 1/poly(\lambda)$ *and* $\delta = 1/2 - 1/poly(\lambda)$.

Instantiating from [KW19]. The scheme provided in [KW19] has single key-$(\mathrm{PM}, \mathrm{EO})$-$\epsilon'$-unremovability and weak pseudorandomness against the watermarking authority[11], where $\epsilon' = 1/2 - 1/poly(\lambda)$. When instantiating our construction with this scheme, we have

[11] In fact, the scheme can achieve a T-restricted pseudorandomness against the watermarking authority, which guarantees security as long as the authority does not query the PRF on some pre-defined T inputs.

Corollary 4.3. *Assuming the worst-case hardness of appropriately parameterized GapSVP problem, there exist watermarkable PRF families with fully collusion resistant-(PM, bounded EO)-ϵ-unremovability, (MO, PE)-δ-unforgeability, and weak pseudorandomness against the watermarking authority, where* $\epsilon = \delta - 1/poly(\lambda)$ *and* $\delta = 1/2 - 1/poly(\lambda)$.

4.4 Correctness and Pseudorandomness of WPRF

Functionality Preserving. The functionality preserving property comes from the functionality preserving property of WPRF_0 and the pseudorandomness of $\mathtt{G}, \mathtt{G}', \mathtt{G}'', \mathsf{F}$ directly.

Note that if WPRF_0 has functionality preserving against adversarially-chosen keys (achieved in [QWZ18, KW19]), WPRF also has this stronger correctness property. Besides, even if WPRF_0 does not satisfy it, WPRF can still achieve functionality preserving against adversarially-chosen keys if the outputs of $\mathtt{G}$ are "random" enough (e.g., if $\mathtt{G}$ is modeled as a random oracle).

Extraction Correctness. The extraction correctness comes from the extraction correctness of WPRF_0, the correctness of PKE, the correctness of SIG, the functionality preserving property of WPRF_0, the correctness of FC, and the pseudorandomness of $\mathtt{G}, \mathtt{G}', \mathtt{G}'', \mathsf{F}$ directly.

Watermarking Meaningfulness. The watermarking meaningfulness comes from the watermarking meaningfulness of WPRF_0 directly.

Pseudorandomness. The pseudorandomness comes from the pseudorandomness of WPRF_0 and the pseudorandomness of $\mathtt{G}, \mathtt{G}'$ by a direct reduction.

(Weak) Pseudorandomness against the Watermarking Authority. The (weak) pseudorandomness against the watermarking authority comes from the (weak) pseudorandomness against the watermarking authority of WPRF_0 and the pseudorandomness of $\mathtt{G}, \mathtt{G}'$ by a direct reduction.

4.5 Unremovability of WPRF

In this section, we prove the fully collusion resistant-$(\mathcal{M}, \mathcal{E})$-$\epsilon$-unremovability of WPRF, assuming that WPRF_0 is single key-$(\mathcal{M}, \mathcal{E})$-$\epsilon'$-unremovable, where $\mathcal{M} \in \{\text{MO}, \text{PM}\}$ and $\mathcal{E} \in \{-, \text{bounded EO}\}$. For simplicity, here we only provide the detailed proof for $\mathcal{M} = \text{PM}$ and $\mathcal{E} = \text{bounded EO}$. The proofs are similar in cases that $\mathcal{M} \in \{\text{PM}\}$ and $\mathcal{E} \in \{-, \text{bounded EO}\}$, and at the end of this section, we also discuss how to deal with a few subtle issues in the proofs when $\mathcal{M} = \text{MO}$.

First, we define the following games between a challenger and a PPT ϵ-unremoving-admissible adversary $\mathcal{A}$:

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

I. First, the challenger generates $(PP_0, MK_0, EK_0) \leftarrow \mathsf{WPRF}_0.\mathtt{Setup}(1^\lambda)$, $(VK, SK) \leftarrow \mathsf{SIG}.\mathtt{KeyGen}(1^\lambda)$, $(PK, DK) \leftarrow \mathsf{PKE}.\mathtt{KeyGen}(1^\lambda)$, and $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then, it returns the public parameter $PP = (PP_0, VK, PK)$ and the mark key $MK = (MK_0, SK, K)$ to $\mathcal{A}$.

II. Then the challenger samples the challenge key $s^* \xleftarrow{\$} \{0,1\}^\lambda$ and generates some variables (determined by s^*), which are used in answering the challenge oracle:

1. $(\check{r}^*, \hat{r}^*, R^*_{\mathsf{FC}}, R^*_{\mathsf{SIG}}, R^*_{\mathsf{PKE}}) = \mathtt{G}(s^*)$.
2. $(r_0^*, r_1^*, \ldots, r_l^*) = \mathtt{G}'(\check{r}^*)$.
3. For $i \in [0, l]$, $k_i^* = \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0; r_i^*)$.
4. $(td^*, (\bar{w}_i^*)_{i\in[N]}) = \mathsf{FC}.\mathtt{Gen}(1^\lambda; R^*_{\mathsf{FC}})$.
5. $\sigma^* = \mathsf{SIG}.\mathtt{Sign}(SK, \check{r}^*; R^*_{\mathsf{SIG}})$.
6. $ct^* = \mathsf{PKE}.\mathtt{Enc}(PK, td^* \| \check{r}^* \| \sigma^*; R^*_{\mathsf{PKE}})$.
7. $(r_0'^*, (r_{i,\iota}'^*)_{i\in[l],\iota\in\{0,1\}}) = \mathsf{F}.\mathtt{Eval}(K, \mathtt{G}''(\hat{r}^*))$.

III. Next the challenger answers $\mathcal{A}$'s oracle queries, including the extraction oracle queries and the challenge oracle queries. Once $\mathcal{A}$ submits a circuit $\mathtt{C}$ to the extraction oracle, the challenger proceeds as follows:

1. Set the circuit $\mathtt{W}_0 : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_0(x) = \mathtt{C}(x)[1 : m]$.
2. $ct = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_0)$.
3. If $ct = \perp$, **return** $\perp$ to $\mathcal{A}$.
4. $(td \| \check{r} \| \sigma) = \mathsf{PKE}.\mathtt{Dec}(DK, ct)$.
5. If $(td \| \check{r} \| \sigma) = \perp$, **return** $\perp$ to $\mathcal{A}$.
6. If $\mathsf{SIG}.\mathtt{Verify}(VK, \check{r}, \sigma) = 0$, **return** $\perp$ to $\mathcal{A}$.
7. $(r_0, r_1, \ldots, r_l) = \mathtt{G}'(\check{r})$.
8. For $i \in [0, l]$, $k_i = \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0; r_i)$.
9. $A = 0$.
10. For $j \in [t]$:
    (a) Sample $x \xleftarrow{\$} \{0,1\}^n$.
    (b) If $\mathtt{C}(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i, x))_{i\in[0,l]}$, $A = A + 1$.
11. If $A > t \cdot \bar{\epsilon}$, **return** $\perp$ to $\mathcal{A}$.
12. For $i \in [l]$:
    (a) $a = im + 1$, $b = (i+1)m$.
    (b) Set the circuit $\mathtt{W}_i : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_i(x) = \mathtt{C}(x)[a : b]$.
    (c) $w[i] = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_i)$.
    (d) If $w[i] \notin [0, 1]$, $w[i] = 0$.
13. $msg \leftarrow \mathsf{FC}.\mathtt{Dec}(td, w)$.
14. **Return** msg to $\mathcal{A}$.

Also, for the h-th challenge oracle query with message msg_h^*, the challenger generates the circuit $\mathtt{C}_h^*$ as follows and returns it back to the adversary.

1. $\mathtt{W}^*_{h,0} = \mathsf{WPRF}_0.\mathtt{Mark}(PP_0, MK_0, k_0^*, ct^*; r_0'^*)$.
2. For $i \in [l]$:
    (a) $b^*_{h,i} = \bar{w}^*_{msg_h^*}[i]$.
    (b) $\mathtt{W}^*_{h,i} = \mathsf{WPRF}_0.\mathtt{Mark}(PP_0, MK_0, k_i^*, b^*_{h,i}; r'^*_{i,b_i})$.
3. Set the circuit $\mathtt{C}_h^*$ as $\mathtt{C}_h^*(x) = (\mathtt{W}^*_{h,i}(x))_{i\in[0,l]}$.

IV. Finally, $\mathcal{A}$ submits a circuit $\tilde{C}$ to the challenge oracle and the challenger checks if $\mathcal{A}$ succeeds in attacking the unremovability of WPRF as follows. Here, we use M to denote the set of all messages submitted to the challenge oracle.

1. Set the circuit $\mathtt{W}_0 : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_0(x) = \tilde{\mathtt{C}}(x)[1:m]$.
2. $ct = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_0)$.
3. If $ct = \perp$, **output** 1.
4. $(td \| \check{r} \| \sigma) = \mathsf{PKE}.\mathtt{Dec}(DK, ct)$.
5. If $(td \| \check{r} \| \sigma) = \perp$, **output** 1.
6. If $\mathsf{SIG}.\mathtt{Verify}(VK, \check{r}, \sigma) = 0$, **output** 1.
7. $(r_0, r_1, \ldots, r_l) = \mathtt{G}'(\check{r})$.
8. For $i \in [0, l]$, $k_i = \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0; r_i)$.
9. $A = 0$.
10. For $j \in [t]$:
    (a) Sample $x \xleftarrow{\$} \{0,1\}^n$.
    (b) If $\tilde{\mathtt{C}}(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i, x))_{i \in [0,l]}$, $A = A + 1$.
11. If $A > t \cdot \bar{\epsilon}$, **output** 1.
12. For $i \in [l]$:
    (a) $a = im + 1$, $b = (i+1)m$.
    (b) Set the circuit $\mathtt{W}_i : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_i(x) = \tilde{\mathtt{C}}(x)[a:b]$.
    (c) $w[i] = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_i)$.
    (d) If $w[i] \notin [0,1]$, $w[i] = 0$.
13. $msg \leftarrow \mathsf{FC}.\mathtt{Dec}(td, w)$.
14. If $msg \notin \mathsf{M}$, **output** 1.
15. **Output** 0.

- **Game 1.** This is identical to Game 0 except that in Step II, the challenger samples $(\check{r}^*, R^*_{\mathsf{FC}}, R^*_{\mathsf{SIG}}, R^*_{\mathsf{PKE}}, r'^*_0, (r'^*_{i,\iota})_{i \in [l], \iota \in \{0,1\}})$ uniformly at random instead of computing them using pseudorandom generators and pseudorandom functions.
- **Game 2.** This is identical to Game 1 except that the challenger changes the way to answer extraction oracle queries. In particular, after receiving a circuit $\mathtt{C}$ and extracting the ciphertext ct from the first part of $\mathtt{C}$, the challenger works as follows (instead of continuing the extraction procedure defined above) if $ct = ct^*$:
  1. $A = 0$.
  2. For $j \in [t]$:
     (a) Sample $x \xleftarrow{\$} \{0,1\}^n$.
     (b) If $\mathtt{C}(x) \neq \mathtt{C}^*_1(x)$, $A = A + 1$.
  3. If $A > t \cdot \bar{\epsilon}$, **return** $\perp$ to $\mathcal{A}$.
  4. For $i \in [l]$:
     (a) $a = im + 1$, $b = (i+1)m$.
     (b) Set the circuit $\mathtt{W}_i : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_i(x) = \mathtt{C}(x)[a:b]$.
     (c) $w[i] = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_i)$.
     (d) If $w[i] \notin [0,1]$, $w[i] = 0$.
  5. $msg \leftarrow \mathsf{FC}.\mathtt{Dec}(td^*, w)$.

6. **Return** msg to $\mathcal{A}$.

- **Game 3.** This is identical to Game 2 except that in Step IV, after extracting the ciphertext ct, it works as follows (instead of continuing the extraction procedure defined above) if $ct = ct^*$:
  1. For $i \in [l]$:
     (a) $a = im + 1$, $b = (i+1)m$.
     (b) Set the circuit $\mathtt{W}_i : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_i(x) = \tilde{\mathtt{C}}(x)[a:b]$.
     (c) $w[i] = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_i)$.
     (d) If $w[i] \notin [0,1]$, $w[i] = 0$.
  2. $msg \leftarrow \mathsf{FC}.\mathtt{Dec}(td^*, w)$.
  3. If $msg \notin \mathsf{M}$, **output** 1.
  4. **Output** 0.
- **Game 4.** This is identical to Game 3 except that the challenger changes the way to generate ct^*. In particular, it computes $ct^* \leftarrow \mathsf{PKE}.\mathtt{Enc}(PK, 0)$.
- **Game 5.** This is identical to Game 4 except that the challenger samples $(r_0^*, r_1^*, \ldots, r_l^*)$ uniformly at random instead of setting them as $(r_0^*, r_1^*, \ldots, r_l^*) = \mathtt{G}'(\check{r}^*)$. Note that in Game 5, each $\mathtt{C}_h^*$ is set as

$$\mathtt{C}_h^*(x) = \mathtt{W}_0^*(x) \| (\mathtt{W}^*_{i, \bar{w}^*_{msg_h^*}[i]}(x))_{i \in [1,l]}$$

  where $\mathtt{W}_0^*, \{\mathtt{W}_{i,j}^*\}_{i \in [l], j \in \{0,1\}}$ are generated as follows in Step II:
  1. For $i \in [0,l]$, $k_i^* \leftarrow \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0)$.
  2. $(td^*, (\bar{w}_i^*)_{i \in [N]}) \leftarrow \mathsf{FC}.\mathtt{Gen}(1^\lambda)$.
  3. $ct^* \leftarrow \mathsf{PKE}.\mathtt{Enc}(PK, 0)$.
  4. $\mathtt{W}_0^* \leftarrow \mathsf{WPRF}_0.\mathtt{Mark}(PP_0, MK_0, k_0^*, ct^*)$.
  5. For $i \in [l]$:
     (a) $\mathtt{W}_{i,0}^* \leftarrow \mathsf{WPRF}_0.\mathtt{Mark}(PP_0, MK_0, k_i^*, 0)$.
     (b) $\mathtt{W}_{i,1}^* \leftarrow \mathsf{WPRF}_0.\mathtt{Mark}(PP_0, MK_0, k_i^*, 1)$.
- **Game 6.** This is identical to Game 5 except that in Step IV, after extracting the ciphertext ct, the challenger aborts the experiment and outputs 2 if $ct \neq ct^*$.
- **Game 7.** This is identical to Game 6 except that when
  - answering an extraction oracle query with extracted ciphertext $ct = ct^*$,
  - performing the final check in Step IV,

  the challenger aborts and outputs 2 if the extracted word w satisfies

$$\exists i \in [l], b \in \{0,1\} : w[i] \neq b \wedge \forall msg \in \mathsf{M}, \bar{w}^*_{msg}[i] = b$$

  We call this event as Bad. Here, we abuse the notion M as the set of all messages submitted to the challenge oracle before the event occurs.

Next, we prove the indistinguishability of each consecutive pair of games defined above and show that the adversary $\mathcal{A}$ will win in the final game (Game 7) with a negligible probability. For simplicity of notation, we use $\mathcal{E}_i$ to denote the output of Game i.

Lemma 4.1. $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. In Game 1, some random variables are sampled uniformly instead of being set as output of pseudorandom generators. As the PRG seed $s^*, \hat{r}^*$ does not appear in the view of $\mathcal{A}$ directly, indistinguishability between Game 0 and Game 1 comes from the pseudorandomness of $\mathtt{G}, \mathtt{G}''$. $\square$

Lemma 4.2. $|\Pr[\mathcal{E}_1 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$.

Proof. Game 1 and Game 2 are identical as long as

1. $(td^* \| \check{r}^* \| \sigma^*) = \mathsf{PKE}.\mathtt{Dec}(DK, ct^*)$.
2. $\mathsf{SIG}.\mathtt{Verify}(VK, \check{r}^*, \sigma^*) = 1$.
3. For all tested input x, $\mathtt{C}_1^*(x) = (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i^*, x))_{i \in [0,l]}$.

The first two conditions are satisfied (with all but negligible probability) due to the correctness of PKE and the correctness of SIG respectively. The last condition comes from the functionality preserving property (against an honest key) of WPRF, which guarantees that the probability that $\mathtt{C}_1^*(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i^*, x))_{i \in [0,l]}$ is negligible for a uniform x. $\square$

Lemma 4.3. $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| \leq negl(\lambda)$.

Proof. Proof of Lemma 4.3 is similar to the proof of Lemma 4.2. Note that as $\mathcal{A}$ is ϵ-unremoving-admissible, there exists $\tilde{i} \in [Q]$ s.t.

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

Also, by the functionality preserving property (against an honest key) of WPRF,

$$|\{x \in \{0,1\}^n : \mathtt{C}_{\tilde{i}}^*(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i^*, x))_{i \in [0,l]}\}| \leq negl(\lambda) \cdot 2^n$$

So, we have

$$|\{x \in \{0,1\}^n : \tilde{\mathtt{C}}(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i^*, x))_{i \in [0,l]}\}| \leq (\epsilon + negl(\lambda)) \cdot 2^n$$

By the Chernoff bounds,

$$\Pr[A \geq t \cdot \bar{\epsilon}] \leq e^{-\frac{\lambda}{60}}$$

Therefore, it will not affect the output even if the challenger does not check whether $\tilde{\mathtt{C}}$ is close to $(\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i^*, x))_{i \in [0,l]}$. $\square$

Lemma 4.4. $|\Pr[\mathcal{E}_3 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 3 and Game 4 comes from the CCA-security of PKE by a direct reduction. Note that the reduction can answer the extraction oracle queries and perform the check in Step IV by querying its decryption oracle, and in both cases it is not required to decrypt the challenge ciphertext ct^*. $\square$

Lemma 4.5. $|\Pr[\mathcal{E}_4 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$.

Proof. As the PRG seed $\check{r}^*$ is not used in any other part of the experiment, indistinguishability between Game 4 and Game 5 comes from the pseudorandomness of $\mathtt{G}'$ directly. □

Lemma 4.6. $|\Pr[\mathcal{E}_5 = 1] - \Pr[\mathcal{E}_6 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 5 and Game 6 comes from the single key-(PM, $(l+1)q$-EO)-ϵ'-unremovability of WPRF_0.

More precisely, if the adversary is able to generate a circuit $\tilde{\mathtt{C}}$ such that $\mathtt{W}_0(\cdot) = \tilde{\mathtt{C}}[1:m]$ is marked with a message not equal to ct^* (with a non-negligible probability), then we can construct an adversary $\mathcal{B}$ that breaks the single key-(PM, $(l+1)q$-EO)-ϵ'-unremovability of WPRF_0.

In particular, the adversary $\mathcal{B}$ sets the circuit $\mathtt{W}_0^*$ as its challenge, which is obtained by submitting ct^* to its challenge oracle. Moreover, $\mathcal{B}$ can answer extraction oracle queries via querying its own extraction oracle. Also, using the mark key returned from its mark key oracle, it can answer the mark key oracle query from $\mathcal{A}$ and to generate $\{\mathtt{W}_{i,b}^*\}_{i\in[l],j\in\{0,1\}}$ when answering the challenge oracle. Finally, $\mathcal{B}$ submits $\mathtt{W}_0(\cdot) = \tilde{\mathtt{C}}[1:m]$ to its challenger. Note that, $\mathcal{B}$ is ϵ'-unremoving-admissible since $\mathcal{A}$ is ϵ-unremoving-admissible, which ensures that $|\{x \in \{0,1\}^n : \mathtt{W}_0(x) \neq \mathtt{W}_0^*(x)\}| \leq \epsilon \cdot 2^n$. □

Lemma 4.7. $|\Pr[\mathcal{E}_6 = 1] - \Pr[\mathcal{E}_7 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 6 and Game 7 comes from the single key-(PM, $(l+1)q$-EO)-ϵ'-unremovability of WPRF_0 by a hybrid argument and the reductions are similar to the reduction provided in the proof of Lemma 4.6.

Note that for $i \in [l]$, if $\bar{w}^*_{msg_1^*}[i] = \bar{w}^*_{msg_2^*}[i]$ for all $msg_1^*, msg_2^* \in \mathsf{M}$, then the adversary $\mathcal{A}$ can only obtain one marked circuit for k_i^*, thus, single key security for WPRF_0 is enough. Also, for $i \in [l]$,

- If *Bad* occurs at Step IV: Let $\mathtt{W}_i(\cdot) = \tilde{\mathtt{C}}(\cdot)[il+1,(i+1)l]$. Then $|\{x \in \{0,1\}^n : \mathtt{W}_i(x) \neq \mathtt{W}_i^*(x)\}| \leq \epsilon \cdot 2^n$ due to the fact that $\mathcal{A}$ is ϵ-unremoving-admissible.
- If *Bad* occurs at an extraction oracle query: Let $\mathtt{W}_i(\cdot) = \mathtt{C}(\cdot)[il+1,(i+1)l]$, where $\mathtt{C}$ is the circuit submitted to the extraction oracle. Assuming that $|\{x \in \{0,1\}^n : \mathtt{W}_i(x) \neq \mathtt{W}_i^*(x)\}| \geq \epsilon' \cdot 2^n$, then by the chernoff bound, the probability that $\mathtt{C}$ can pass the check in Step 3 (in the new extraction procedure defined in Game 2) is negligible, i.e., the challenger is not able to recover a word w in this case.

Thus, the adversary $\mathcal{B}$ is ϵ'-unremoving-admissible.

□

Lemma 4.8. $\Pr[\mathcal{E}_7 = 1] \leq negl(\lambda)$.

Proof. Lemma 4.8 comes from adaptive security with $(q+1)$ extraction queries of FC by a direction reduction. □

Combining Lemma 4.1 to Lemma 4.8, we have $\Pr[\mathcal{E}_0 = 1] \leq negl(\lambda)$, i.e., the probability that $\mathcal{A}$ wins in the real experiment $\mathtt{ExptUR}$ is negligible. This completes the proof of unremovability.

The proofs in cases that $\mathcal{M}$ = MO. The above proof strategies (almost) work perfectly in cases that $\mathcal{M}$ = MO. One subtle issue is that in the proof of Lemma 4.6 and that of Lemma 4.7, the adversary $\mathcal{B}$ for single key security of WPRF_0 needs to simulate the marking oracle for $\mathcal{A}$ via its own marking oracle. However, as the seed (WPRF's PRF key) is chosen by $\mathcal{A}$, security of pseudorandom generator is not enough to ensure that the simulated marking oracle (which runs $\mathsf{WPRF}_0.\mathtt{Mark}$ on fresh randomness) is indistinguishable from an honest marking oracle (which runs $\mathsf{WPRF}_0.\mathtt{Mark}$ on randomness output by some pseudorandomness generation component). To solve this subtle issue, we employ a pseudorandom function F to generate the randomness for $\mathsf{WPRF}_0.\mathtt{Mark}$. Since the secret key K of F is put in the mark key, which is not given to $\mathcal{A}$ in this case, we can argue the indistinguishability of these two modes for answering marking oracle queries.

4.6 Unforgeability of WPRF

Next, we prove the unforgeability of WPRF. First, we define the following games between a challenger and a PPT ϵ'-unforging-admissible adversary $\mathcal{A}$:

- **Game 0.** This is the real experiment $\mathtt{ExptUF}$. More precisely, the challenger proceeds as follows.
  I. First, the challenger generates $(PP_0, MK_0, EK_0) \leftarrow \mathsf{WPRF}_0.\mathtt{Setup}(1^\lambda)$, $(VK, SK) \leftarrow \mathsf{SIG}.\mathtt{KeyGen}(1^\lambda)$, $(PK, DK) \leftarrow \mathsf{PKE}.\mathtt{KeyGen}(1^\lambda)$, and $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then, it returns the public parameter $PP = (PP_0, VK, PK)$ to $\mathcal{A}$.
  II. Next, it answers $\mathcal{A}$'s oracle queries:
    - If $\mathcal{A}$ submits a query to the extraction key oracle, the challenger returns $EK = (EK_0, DK)$ to $\mathcal{A}$.
    - If $\mathcal{A}$ submits the h-th marking oracle query $(s^h, msg^h) \in \{0,1\}^\lambda \times [N]$, the challenger returns $\mathtt{C}^h \leftarrow \mathtt{Mark}(PP, MK, s^h, msg^h)$ to $\mathcal{A}$.

  III. Finally, $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$ and the challenger proceeds as follows:
    1. Set the circuit $\mathtt{W}_0 : \{0,1\}^n \to \{0,1\}^m$ as $\mathtt{W}_0(x) = \tilde{\mathtt{C}}(x)[1:m]$.
    2. $ct = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_0)$.
    3. If $ct = \perp$, **output 0**.
    4. $(td \| \check{r} \| \sigma) = \mathsf{PKE}.\mathtt{Dec}(DK, ct)$.
    5. If $(td \| \check{r} \| \sigma) = \perp$, **output 0**.
    6. If $\mathsf{SIG}.\mathtt{Verify}(VK, \check{r}, \sigma) = 0$, **output 0**.
    7. $(r_0, r_1, \ldots, r_l) = \mathtt{G}'(\check{r})$.
    8. For $i \in [0, l]$, $k_i = \mathsf{WPRF}_0.\mathtt{KeyGen}(PP_0; r_i)$.
    9. $A = 0$.
    10. For $j \in [t]$:
        (a) Sample $x \xleftarrow{\$} \{0,1\}^n$.
        (b) If $\tilde{\mathtt{C}}(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i, x))_{i \in [0,l]}$, $A = A + 1$.
    11. If $A > t \cdot \bar{\epsilon}$, **output 0**.
    12. For $i \in [l]$:
        (a) $a = im + 1$, $b = (i+1)m$.

   (b) Set the circuit $\mathtt{W}_i : \{0,1\}^n \rightarrow \{0,1\}^m$ as $\mathtt{W}_i(x) = \tilde{\mathtt{C}}(x)[a:b]$.
   (c) $w[i] = \mathsf{WPRF}_0.\mathtt{Extract}(PP_0, EK_0, \mathtt{W}_i)$.
   (d) If $w[i] \notin [0,1]$, $w[i] = 0$.
13. $msg \leftarrow \mathsf{FC}.\mathtt{Dec}(td, w)$.
14. If $msg = \perp$, **output 0**.
15. **Output 1**.

- **Game 1.** This is identical to Game 0 except that in Step III.6, after checking if σ is a valid signature for $\check{r}$, the challenger further checks if $\check{r}$ has appeared. In particular, let Q be the number of marking oracle queries the adversary made and for $h \in [Q]$, let $(\check{r}^h, \hat{r}^h, R^h_{\mathsf{FC}}, R^h_{\mathsf{SIG}}, R^h_{\mathsf{PKE}}) = \mathtt{G}(s^h)$, then the challenger outputs 0 if $\forall h \in [Q], \check{r} \neq \check{r}^h$.

Game 0 and Game 1 are identical unless $\mathsf{SIG}.\mathtt{Verify}(VK, \check{r}, \sigma) = 1$ but $\forall h \in [Q], \check{r} \neq \check{r}^h$, i.e., the adversary generates a valid signature σ for a new message $\check{r}$ after viewing signatures for messages $\check{r}^1, \ldots, \check{r}^Q$. This occurs with only a negligible probability due to the existentially unforgeable of SIG. Thus, the probability that $\mathcal{A}$ succeeds in Game 0 and that in Game 1 are close.

Next, we argue that Game 1 outputs 1 with only a negligible probability. First, due to the new checking rule in Game 1, $\check{r} = \check{r}^h$ for some $h \in [Q]$ (otherwise, the experiment outputs 0 directly). Then, by the functionality preserving property (against adversarially-chosen PRF keys) of WPRF, with all but negligible probability,

$$|\{x \in \{0,1\}^n : (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i, x))_{i\in[0,l]} \neq \mathtt{C}^h(x)\}| \leq negl(\lambda) \cdot 2^n$$

Since $\mathcal{A}$ is ϵ'-unforging-admissible,

$$|\{x \in \{0,1\}^n : \tilde{C}(x) \neq \mathtt{C}^h(x)\}| \geq \epsilon' \cdot 2^n$$

So, we have[12]

$$|\{x \in \{0,1\}^n : \tilde{C}(x) \neq (\mathsf{WPRF}_0.\mathtt{Eval}(PP_0, k_i, x))_{i\in[0,l]}\}| \geq (\epsilon' - negl(\lambda)) \cdot 2^n \quad (1)$$

Finally, by the Chernoff bounds,

$$\Pr[A \leq t \cdot \bar{\epsilon}] \leq e^{-\frac{\lambda-2}{8}}$$

i.e., $\tilde{C}$ can pass the check in Step III.11 with only negligible probability.

This completes the proof of unforgeability.

Acknowledgement. We appreciate the anonymous reviewers for their valuable comments and especially to one reviewer for suggesting defining collusion resistant unremovability against adversaries that can adaptively make challenge queries. Part of this work was supported by the National Natural Science Foundation of China (Grant No. 61972332, U1636205, 61572294, 61632020), and the Research Grant Council of Hong Kong (Grant No. 25206317).

[12] If we use the alternative definition of unforging-admissibility (see Footnote 10), then ϵ'-unforging-admissibility implies Equation (1) directly and we do not need functionality preserving against adversarially-chosen PRF keys for WPRF.

References

[AMN+18] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In *CRYPTO*, pages 543–574. Springer, 2018.

[AT09] Ehsan Amiri and Gábor Tardos. High rate fingerprinting codes and the fingerprinting capacity. In *SODA*, pages 336–345. SIAM, 2009.

[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.

[BKM17] Dan Boneh, Sam Kim, and Hart Montgomery. Private puncturable PRFs from standard lattice assumptions. In *EUROCRYPT*, pages 415–445. Springer, 2017.

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

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

[BMP85] GR Blakley, Catherine Meadows, and George B Purdy. Fingerprinting long forgiving messages. In *CRYPTO*, pages 180–189. Springer, 1985.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *CCS*, pages 501–510. ACM, 2008.

[BS95] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. In *CRYPTO*, pages 452–465. Springer, 1995.

[BS98] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In *TCC*, pages 264–302. Springer, 2017.

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

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In *EUROCRYPT*, pages 446–476. Springer, 2017.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, pages 257–270. Springer, 1994.

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

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In *CRYPTO*, pages 577–607. Springer, 2018.

[DKN+20] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. Cryptology ePrint Archive, Report 2020/111, 2020. `https://eprint.iacr.org/2020/111`.

[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.

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

[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.

[LdW14] Thijs Laarhoven and Benne de Weger. Optimal symmetric tardos traitor tracing schemes. *Designs, Codes and Cryptography*, 71(1):83–103, 2014.

[NFH+09] Koji Nuida, Satoshi Fujitsu, Manabu Hagiwara, Takashi Kitagawa, Hajime Watanabe, Kazuto Ogawa, and Hideki Imai. An improvement of discrete tardos fingerprinting codes. *Designs, Codes and Cryptography*, 52(3):339–362, 2009.

[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.

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

[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.

[Tar03] Gábor Tardos. Optimal probabilistic fingerprint codes. In *STOC*, pages 116–125. ACM, 2003.

[Wag83] Neal R Wagner. Fingerprinting. In *S& P*, pages 18–18. IEEE, 1983.

[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. Springer, 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. Springer, 2019.

[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.