Collusion Resistant Watermarking Schemes for Cryptographic Functionalities

Rupeng Yang[1,2] ⋆, Man Ho Au[2] ⋆⋆, Junzuo Lai[3], Qiuliang Xu[1] ⋆⋆, and Zuoxia Yu[2]
⋆ ⋆ ⋆

[1] School of Computer Science and Technology, Shandong University,
Jinan, 250101, China
orbbyrp@gmail.com, xql@sdu.edu.cn
[2] Department of Computing, The Hong Kong Polytechnic University,
Hung Hom, Hong Kong
csallen@comp.polyu.edu.hk, zuoxia.yu@gmail.com
[3] College of Information Science and Technology, Jinan University,
Guangzhou, 510632, China
laijunzuo@gmail.com

Abstract. A cryptographic watermarking scheme embeds message into a program while preserving its functionality. Essential security of the watermarking schemes requires that no one could remove the marking message of a marked program without substantially changing its functionality. In practical applications, it is common to mark a program with multiple different messages, e.g. in the secret-leaker tracing scenarios. Thus, it is usually required that the watermarking scheme should be secure against the "collusion attacks", where the adversary can obtain multiple watermarked programs embedded with different messages for the same functionality. However, current works in this area have not formally considered this requirement.

In this paper, we formally address the problem and give new security definition for watermarking schemes that captures the collusion attacks. Then we explore the existence of watermarking schemes secure under our new security definition:

- On the negative side, we observe that all current watermarking schemes either do not support multi-message embedding inherently or are vulnerable to the collusion attacks.
- On the positive side, we construct watermarking scheme secure against the collusion attacks for pseudorandom function (PRF). This is achieved by introducing a new message-embedding technique in the watermarking settings and is built on a newly presented primitive, namely, private multi-programmable PRF. Based on our watermarking scheme for PRF, we also construct watermarking schemes for various other cryptographic functionalities.

⋆ This work was mainly done when doing the internship at The Hong Kong Polytechnic University.
⋆⋆ Corresponding author.
⋆ ⋆ ⋆ The second to the fifth authors are sorted in the alphabetical order.

1 Introduction

A watermarking scheme allows one to embed some information into a program[4] without significantly changing its functionality. There are many natural applications of watermarking schemes, most notably, it can be used for the ownership protection and for the information leaker tracing. The formal definition of watermarking schemes for programs is first presented by Barak et al. in [BGI+01]. A number of properties are also defined in it and subsequent works, including

- **Unremovability:** This is the essential security property for the watermarking scheme, which requires that it is hard to remove the embedded information in a marked program without destroying the program.
- **Unforgeability:** The unforgeability requires that anyone without the marking secret key cannot generate marked programs that are not functionally similar to marked programs he/she gets.
- **Public Extraction:** In a watermarking scheme supporting public extraction, the extraction key can be made public to enable anyone to extract the embedded message in a marked program.
- **Message-Embedding:** The message-embedding property allows one to embed a given string (instead of merely a mark symbol) into the watermarked object.
- **Decentralized Key Generation vs Centralized Key Generation:** In a watermarking scheme with centralized key generation, a user cannot generate the key (pair) of original scheme by himself/herself, instead, users need to obtain their keys from a "watermarking center" possessing the marking secret key in practice.
- **Stateless vs Stateful:** In a stateful watermarking scheme, the mark algorithm needs to maintain a state. The state is shared with the extraction algorithm and is updated each time when the mark algorithm is invoked.

Despite its natural concept and its wide applications, watermarking schemes for programs secure against arbitrary removal strategies is not given until 2015. In two concurrent works [NW15] and [CHV15] (which are merged into [CHN+16]), watermarking schemes for the evaluation algorithm of pseudorandom functions (PRF) are constructed from indistinguishability obfuscators. Both constructions can support public extraction, but none of them is proved to have the standard unforgeability property: in [NW15], the unforgeability is not considered; while in [CHV15], the construction is only proved to have a relaxed unforgeability.

Then in [BLW17], based on their proposed private programmable PRF, which can be instantiated from indistinguishability obfuscator, watermarkable PRF (PRF admitting a watermarking scheme for its evaluation algorithm) with standard unforgeability is constructed. Subsequently, in [KW17], based on a relaxed variant of the private programmable PRF, which is denoted as translucent puncturable PRF, watermarkable PRF from the standard lattice assumptions are presented.

Beyond watermarkable PRF, watermarkable public key encryption (PKE) schemes (PKE scheme admitting a watermarking scheme for its decryption algorithm) and watermarkable signature schemes (signature scheme admitting a watermarking scheme

[4] In this paper, we concentrate on watermarking schemes for programs and only consider those with a provable security against arbitrary removal strategies.

for its sign algorithm) are also constructed. In [NW15, CHN+16], the authors show that assuming the indistinguishability obfuscator exists, one can construct watermarkable PKE schemes and watermarkable signature schemes from a watermarkable PRF that allows one to generate a punctured marked key. However, watermarkable PKE/signature schemes constructed in this way do not have the decentralized key generation property, since the key generation algorithm in their construction needs to additionally take the marking secret key as an input.

Recently, a very simple yet elegant construction of watermarking scheme for any PKE schemes is constructed in [BKS17] (recall that in previous works, only watermarking schemes for specifically constructed schemes are given). However, its mark algorithm and extraction algorithm are stateful and its extraction algorithm has a running time linear to the number of times the mark algorithm has been invoked. Besides, it does not support multiple-message-embedding inherently and do not support decentralized key generation.

Collusion Resistance of Watermarking schemes. Generally, in practical applications, it is usually required that unremovability of the watermarking schemes holds under "collusion attacks", where the attacker could access several copies embedded with different information of the same program. To demonstrate this, consider the following scenario. A software development company wants to outsource the test of its recently developed software to several different organizations. To prevent these organizations from leaking the software, before sending a copy of the software to an organization, the company will employ a watermarking scheme to embed the name of the target organization into the copy. Here, the used watermarking scheme should enable the company to trace the software leaker even when a few target organizations collude.

Unfortunately, to the best of our knowledge, there is still no (provably) collusion resistant watermarking scheme constructed in previous works. In particular, current watermarking schemes either do not support multiple-message embedding inherently [BKS17], or vulnerable to the collusion attacks [NW15, CHN+16, BLW17, KW17].

To see this, recall that in current watermarking schemes with message-embedding, programs are marked by first puncturing the program at some inputs and then embedding the message into these punctured inputs. To extract the embedded message from a given circuit, one first recover (parts of) these punctured inputs, evaluate the circuit on these recovered punctured input, and compute the message from the outputs. Security of these watermarking schemes relies on the fact that the punctured inputs (either the pattern of these inputs [NW15, CHV15] or the inputs themselves [BLW17, KW17]) are hidden given only the marked circuits. Our observation is that for current watermarking schemes with message embedding, if two circuits embedded with different messages for the same functionality are given, then one can find (parts of) the hidden punctured inputs and are able to modify or remove the marks.

To better explain this observation, we takes the watermarking scheme from [BLW17] as an example. For simplicity of description, we simplify their construction here. Roughly speaking, the watermarking scheme is for a private programmable PRF PF, which is a punctured PRF that additionally has the "reprogrammability" and the "constraint privacy", where the reprogrammability allows one to specify the output on the punctured input when generating a punctured key and the constraint privacy requires the punc-

tured key to hide its punctured input. The marking secret key here is a secret key K of a normal PRF F and a random string z in the domain of PF. To embed a message $\mathtt{m}$ into a key k of PF, the mark algorithm computes

$$(x, y_1, y_2) = \mathsf{F}(K, \mathsf{PF}(k, z))$$
$$ck \leftarrow \mathit{Puncture}(k, x, (y_1, y_2 \oplus \mathtt{m}))$$

and outputs a circuit C that evaluates PF with ck. Note that here $C(\cdot)$ and $\mathsf{PF}(k, \cdot)$ evaluates identically on all inputs except that $C(x) = (y_1, y_2 \oplus \mathtt{m})$. To extract the message given a circuit C, the extraction algorithm first computes

$$(x, y_1, y_2) = \mathsf{F}(K, C(z))$$
$$(y_1', y_2') = C(x)$$

then it outputs $\perp$ if $y_1' \neq y_1$ and outputs $y_2' \oplus y_2$ otherwise. Since the punctured input of a marked circuit (i.e., the punctured key) is hidden, the adversary is unlikely to modify the output of the marked circuit on the punctured input without significantly modifying its functionality. Then, the unremovability follows.

Now, let C_1 and C_2 be two circuits that are generated by embedding two different messages $\mathtt{m}_1$ and $\mathtt{m}_2$ respectively into the same secret key k. Obviously, the punctured input x^* of C_1 and C_2 are identical and $C_1(x^*) \neq C_2(x^*)$. Thus, an adversary obtaining the two circuits C_1 and C_2 can locate the punctured input x^* and modify the value on it. In particular, it can construct a circuit $\tilde{C}$ from C_1 and C_2 that

$$\tilde{C}(x) = \begin{cases} C_1(x) \oplus 1^m & \text{if } C_1(x) \neq C_2(x). \\ C_1(x) & \text{otherwise.} \end{cases}$$

where m is the output length of $\mathtt{PF}$. The circuit $\tilde{C}$ differs with C_1, C_2 and $\mathsf{PF}(k, \cdot)$ on merely one input and is not embedded with any message. Thus, the adversary succeeds in launching the collusion attacks. One can use a similar idea to launch collusion attacks to other current watermarking schemes supporting message-embedding, too.[5]

1.1 Our Results

In this paper, we explore the existence of watermarkable cryptographic primitives secure against the collusion attacks and:

- We present the notion of *collusion resistant watermarking scheme* to capture the collusion attacks. The collusion resistant watermarking scheme requires a stronger unremovability (collusion resistant unremovability) that allows the adversary to obtain circuits embedded with different messages for the same functionality.
- We give a construction of collusion resistant watermarkable PRF, which is the first watermarkable cryptographic primitive secure against the collusion attacks. To construct collusion resistant watermarkable PRF, we introduce a new message-embedding technique in the watermarking setting, and propose a new primitive, namely, private multi-programmable PRF, which may be of independent interest.

[5] We remark that this will not affect the claimed security for current watermarking schemes, the attacks only show that they are not applicable in scenarios that collusion attacks are available.

– Based on our collusion resistant watermarkable PRF, we construct watermarkable symmetric key encryption (SKE) scheme, watermarkable message authentication code (MAC) scheme, watermarkable PKE scheme and watermarkable signature scheme, all of which are secure against the collusion attacks. Besides, our watermarkable PKE scheme is the first of its kind achieving the decentralized key generation property and satisfying stateless and unforgeable simultaneously.

In summary, in Table 1, we compare the main features achieved by current watermarking schemes and our watermarking schemes.

Table 1: The Comparison.

		Decentralized Key Generation	Public Extraction	Unforge-ability	Message Embedding	Stateless	Collusion Resistant
[NW15]	PRF	✓	✓	✗	✓	✓	✗
	PKE	✗	✓	✗	✓	✓	✗
	SIG*	✗	✓	✗	✓	✓	✗
[CHV15]	PRF	✓	✓	✓†	✗	✓	-
[CHN+16]	PRF	✓	✓	✗	✓	✓	✗
	PKE	✗	✓	✗	✓	✓	✗
	SIG	✗	✓	✗	✓	✓	✗
[BLW17]	PRF	✗	✗	✓	✓	✓	✗
[KW17]	PRF	✓	✗	✓	✓	✓	✗
[BKS17]	PKE	✗	✗	✓	✗	✗	-
	PKE	✗	✓	✓	✗	✗	-
Ours	PRF	✓	✗	✓	✓	✓	✓
	SKE	✓	✗	✓	✓	✓	✓
	MAC	✓	✗	✓	✓	✓	✓
	PKE	✓	✗	✓	✓	✓	✓
	SIG	✗	✗	✓	✓	✓	✓

∗: We use "SIG" to denote signature schemes.
†: The watermarking scheme in [CHV15] can only achieve a relaxed form of unforgeability.

1.2 Our Techniques

Next, we give a brief overview of how we achieve these results.

On Constructing Collusion Resistant Watermarkable PRF. The collusion attack for the watermarkable PRF works in two steps: first, the attacker locates (parts of) the "punctured inputs" by comparing the outputs of circuits embedded with different messages; then it removes or modifies the marked messages by modifying the value on these located inputs. Since black-box extraction (i.e., the extraction algorithm regards

the input circuit as a black-box oracle) is needed in current construction of watermarking schemes against arbitrary removal strategies, circuits embedded with different messages will always evaluate differently on some inputs. Thus, it seems that the first step will always work. However, we can still hope to prevent the attacker from succeeding in the second step by embedding the message in a more robust manner.

In this work, we construct collusion resistant watermarkable PRF by introducing a new robust message embedding method in the watermarking settings. In more detail, our watermarking scheme is built on a variant of the private programmable PRF, which allows one to puncture and reprogram on multiple inputs and is called private multi-programmable PRF in this work. The marking secret key is also a secret key K of a normal PRF F and a random string z in the domain of the private multi-programmable PRF PF. To embed a message $\mathtt{m} \in [1, N]$ for some polynomial N into a key k of PF, the mark algorithm computes a $(N \times t)$-dimension matrix for a suitable polynomial t

$$(\{x_{i,j}, y_{i,j}\}_{i\in[1,N],j\in[1,t]}) = \mathsf{F}(K, \mathsf{PF}(k, z))$$

where each element in this matrix is an input/output pair of PF. Then it punctures and reprograms the secret key k according to the first $\mathtt{m}$ rows of the matrix

$$ck \leftarrow \mathit{Puncture}(k, \{x_{i,j}, y_{i,j}\}_{i\in[1,\mathtt{m}],j\in[1,t]})$$

and outputs a circuit C that evaluates PF with ck. Also, here $C(\cdot)$ and $\mathsf{PF}(k, \cdot)$ evaluate identically on all inputs except that $C(x_{i,j}) = y_{i,j}$ for $i \in [1, \mathtt{m}]$, $j \in [1, t]$. To extract the message given a circuit C, the extraction algorithm first recovers the matrix

$$(\{x_{i,j}, y_{i,j}\}_{i\in[1,N],j\in[1,t]}) = \mathsf{F}(K, C(z))$$

then it computes the number of inputs that is "punctured" in C in each row

$$\forall i \in [1, N], \alpha_i = \|\{j \mid C(x_{i,j}) = y_{i,j}\}\|$$

It outputs "unmarked" if α_1 is small and it outputs i if $\alpha_i - \alpha_{i+1}$ is large.

Next, we show why the above construction idea can lead to the collusion resistant unremovability. For simplicity, we consider an adversary that merely obtains 2 marked circuits, say, C_1 and C_2, embedded with different messages, say, $\mathtt{m}_1, \mathtt{m}_2 \in [1, N]$, for the same key k of PF (w.l.o.g, we assume that $\mathtt{m}_1 < \mathtt{m}_2$). Let $\tilde{C}$ be the circuit submitted by the adversary. Note that $\tilde{C}$ will differ with C_1 and C_2 on negligible fraction of inputs. Due to the pseudorandomness of F, z is hidden to the adversary, so the adversary is not likely to modify the output on z in $\tilde{C}$. Thus, we can safely assume that the same matrix is used in the mark algorithm (to generate C_1 and C_2) and in the extraction algorithm (to extract $\tilde{C}$). Now, denote the matrix as $\mathcal{M} = \{x_{i,j}, y_{i,j}\}_{i\in[1,N],j\in[1,t]}$ and let $\alpha_i = \|\{j \mid C(x_{i,j}) = y_{i,j}\}\|$ be the internal variable used when extracting $\tilde{C}$. We next analyze the value of α_i for $i \in [1, N]$.

Observe that although the adversary can find the set $\{x_{i,j}\}_{i\in[\mathtt{m}_1+1,\mathtt{m}_2],j\in[1,t]}$, i.e. all inputs from the $(\mathtt{m}_1 + 1)$th to the $\mathtt{m}_2$th row of $\mathcal{M}$, which are evaluated differently in C_1 and C_2, it is infeasible for the adversary to locate the row index of each input $x_{i,j}$, i.e. the ith row and the jth row are indistinguishable to the adversary for $i, j \in [\mathtt{m}_1 + 1, \mathtt{m}_2]$.

Therefore, the value of α_i and α_j for $i, j \in [\mathtt{m}_1 + 1, \mathtt{m}_2]$ are not likely to differ too much. Also, as all inputs in the ith row for $i \geq \mathtt{m}_2 + 1$ are punctured in neither C_1 nor C_2 and are totally hidden to the adversary, the adversary is not likely to find and puncture them in $\tilde{C}$. Thus, the value of α_i for $i \in [\mathtt{m}_2 + 1, N]$ will be close to 0. Besides, assuming that all inputs in the ith row for $i \leq \mathtt{m}_1$ are also hidden to the adversary, since they are punctured in both C_1 and C_2, the adversary is not likely to find and "unpuncture" them in $\tilde{C}$. Thus, the value of α_i for $i \in [1, \mathtt{m}_1]$ will be close to t. Therefore, the extraction algorithm on $\tilde{C}$ will output either $\mathtt{m}_1$ or $\mathtt{m}_2$, which implies that the adversary will fail. It remains to show the hiding of the first $\mathtt{m}_1$ rows. This is hoped to come from the constraint privacy of the underlying private multi-programmable PRF. However, constraint privacy that is formally defined and proved in current works does not suit our proof (we will discuss this in Sec. 4). Fortunately, we find that the *consistent privacy*, which is discussed informally in [BLW17], is applicable to our setting, and we can design a private multi-programmable PRF with consistent privacy by adapting the private programmable PRF in [BKM17].

This is the basic idea how we achieve security against collusion attacks. To construct a full-fledged collusion resistant watermarkable PRF, we still need to tackle with other problems, e.g. handling arbitrary polynomial collusion, dealing with marking oracle queries, proving unforgeability, etc. Note that our method for message-embedding is somewhat similar to the techniques used in [BSW06] for constructing traitor tracing schemes, which is a multi-receiver PKE scheme allowing tracing of secret key leakers. However, we need to deal with different issues when constructing the scheme and conducting the security proof. We will provide more details on how to define and construct our private multi-programmable PRF in Sec. 4 and will give our main construction of collusion resistant watermarkable PRF in Sec. 5.

On Constructing Collusion Resistant Watermarking Schemes for Advanced Cryptographic Functionalities. It is worth noting that our constructed watermarking scheme for PRF is also a collusion resistant watermarking scheme for the evaluation algorithm of a puncturable PRF. This is useful when constructing advanced watermarkable cryptographic primitives. More precisely, to construct advanced collusion resistant watermarkable cryptographic primitives from our collusion resistant watermarkable PRF, we rely on the observation that one can construct a SKE scheme (MAC scheme, PKE scheme or signature scheme) with decryption algorithm (resp. mac algorithm, decryption algorithm, or sign algorithm) that is "nothing more than a (puncturable) PRF evaluation". The observation was first presented in [NW15, CHN+16] to construct the watermarkable PKE scheme and the watermarkable signature scheme therein.

However, there is a subtle issue to complete the construction. More precisely, when constructing watermarkable PKE schemes, the public key may leak information about the secret key, which may help the adversary to remove or modify the mark of a marked secret key. In [NW15, CHN+16], this problem is solved by generating the public key from the marked secret key in the construction. But watermarking schemes constructed in this manner no longer have the decentralized key generation property.[6] In this pa-

[6] Similar issue also occurs when constructing watermarkable signature schemes, but we do not know how to solve it in the signature setting.

per, we observe that if the underlying watermarkable (puncturable) PRF satisfies an additional property, then the original public key, which is generated from an unmarked secret key, is in fact computationally indistinguishable from a public key generated from a marked secret key, i.e., it will not help the adversary to break the unremovability of the watermarking scheme for the advanced primitive. The property is easy to obtain and can be fulfilled by our constructed watermarkable (puncturable) PRF. So, we can construct watermarking scheme for normal PKE schemes whose key generation algorithm does not need to take a marking secret key as input, which preserves the decentralized key generation property. We will provide more details on how to construct collusion resistant watermarking schemes for advanced cryptographic primitives in Sec. 6.

1.3 Related Works

Additional Related Works on Watermarking Schemes. There are numerous works (see [CMB+07] and references therein) attempting to use ad hoc techniques to watermark a wide class of digital objects, e.g. images, audios, videos, etc., but these constructions lack rigorous security analysis and are (potentially) vulnerable to some attacks. In another line of research [NSS99, YF11, Nis13], watermarking schemes for the cryptographic object (e.g the key, the signature etc.) are constructed and rigorously analyzed under the assumption that the adversary will not change the format of the watermarked objects. The work [HMW07] refined the definition for watermarking schemes, and formally defined several properties, e.g. the unforgeability and the messge-embedding property, for the first time. Although it focuses on watermarking schemes for perceptual objects, the definition ideas can be adopted to the program watermarking cases.

Traitor Tracing Scheme. The notion of collusion resistant watermarking scheme is somewhat similar to the notion of traitor tracing scheme, which was first presented in [CFN94]. Traitor tracing schemes has been formally studied for a long time and there are numerous works in this area (see e.g. [BSW06, BN08, BZ14, NWZ16] and references therein for an overview of previous works). Especially, recently, in [NWZ16], traitor tracing scheme supporting arbitrary information embedding is constructed. However, as these two notions have a few inherent differences, solutions to the traitor tracing problem do not yield watermarking schemes directly. First, while the traitor tracing scheme concentrates on tracing secret key leakers in an encryption scheme, the watermarking scheme aims at marking general purpose programs. Another difference is that in a traitor tracing scheme, secret keys of all users are functionally equivalent and are issued by a center, while in a watermarking scheme, user can choose their keys all by themselves and keys with different functionalities can be watermarked.

Private Constrained PRF The concept of private constrained PRF was first introduced in [BLW17]. Then, in a series of works, private constrained PRF with different constraints are constructed from standard lattice assumptions [BKM17, CC17, BTVW17]. In [BLW17], the authors also presents a variant of the private constrained PRF, which is called private programmable PRF and allows one to assign the outputs on the punctured inputs. This could enable one to test whether an input is punctured in certain

cases, and can be used to construct watermarking schemes. Then, in [KW17], translucent constrained PRF, another variant of private constrained PRF that allows one to test whether an input is punctured directly, is defined and constructed from standard lattice assumptions.

We remark that all current private constrained PRFs from standard assumption can only achieve a weak single-key security, which only allows the adversary to obtain a single constrained key. So, they are not suitable for our construction of watermarking schemes with collusion resistant unremovability, where the adversary is expected to obtain different constrained keys.

2 Preliminaries

Notations. Let a be a string, then we use $a[i]$ to denote the ith character of a for an integer i not exceeding the length of a, and use $a[i : j]$ to denote the substring $(a[i], a[i+1], \ldots, a[j])$ of a for integers $i \leq j$ not exceeding the length of a. Let $\mathcal{S}$ be a finite set, then 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 n, m be two integers, we write $FUN_{n,m}$ to denote the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$. 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 that is not less than a and not greater than b. Following the syntax in [BLW17], for a circuit family C indexed by a few, say m, constants, we write $C[c_1, \ldots, c_m]$ to denote a circuit with constants $c_1, \ldots, c_m$. We also follow the syntax in [BLW17, KW17] to denote the circuit similarity.

Definition 2.1 ([BLW17]). *Fix a circuit class C on n-bit inputs. For two circuits C, $C' \in C$ and for a non-decreasing function $f : \mathbb{N} \to \mathbb{N}$, we write $C \sim_f C'$ to denote that the two circuits agree on all but an $1/f(n)$ fraction of inputs. More formally, we define*

$$C \sim_f C' \qquad \Longleftrightarrow \qquad \Pr_{x \xleftarrow{\$} \{0,1\}^n} [C(x) \neq C'(x)] \leq 1/f(n).$$

We also write $C \nsim_f C'$ to denote that C and C' differ on at least a $1/f(n)$ fraction of inputs. More formally, we define

$$C \nsim_f C' \qquad \Longleftrightarrow \qquad \Pr_{x \xleftarrow{\$} \{0,1\}^n} [C(x) \neq C'(x)] \geq 1/f(n).$$

To prove the security of the watermarking scheme for the decryption algorithm of the PKE scheme, we will use the Chernoff bound. There are various forms of the Chernoff bound, here we use the one from [Goe15].

Lemma 2.1 (Chernoff Bounds). *Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}(X) = \sum_{i=1}^{n} p_i$. Then*

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

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

The Hypergeometric Distribution. In this work, we will also use the hypergeometric distribution to help analyse the success probability of the adversary in the proof of Theorem 5.1. Here, we recall the notion and a few properties of the hypergeometric distribution. Let N, K, n be natural numbers that $K \leq N$ and $n \leq N$, then the hypergeometric distribution $\mathcal{H}(K, N, n)$ is the number of "good" elements in n elements sampled without replacement from a set of N elements with K good ones. Formally, the probability mass function of a random variable X following the hypergeometric distribution $\mathcal{H}(K, N, n)$ is defined as

$$\Pr[X = k] = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

for $max(0, n + K - N) \leq k \leq min(K, n)$, where we use $\binom{a}{b}$ to denote the binomial coefficient for natural numbers $b \leq a$. We will use the tail bound of the hypergeometric distribution in our proof, which is formally described as following.

Lemma 2.2 ([Chv79]). *Let X be a random variable following the hypergeometric distribution $\mathcal{H}(K, N, n)$, and $\delta >= 0$, then we have:*

$$\Pr[X \leq \frac{K}{N} \cdot n - \delta] \leq e^{-2\delta^2/n}$$

$$\Pr[X \geq \frac{K}{N} \cdot n + \delta] \leq e^{-2\delta^2/n}$$

2.1 Cryptographic Primitives

Next, we recall a few cryptographic primitives that are employed in this work.

The Puncturable Pseudorandom Function with Key Injectivity. The notion of puncturable pseudorandom function was first formalized by Sahai and Waters in [SW14]. They also show that a PRF function constructed via the GGM-framework [GGM84a] is a puncturable PRF. In this work, we will use a slightly stronger version of puncturable PRF, namely, puncturable PRF with key injectivity, which can be defined as follows.[7]

Definition 2.2. *A puncturable PRF family $F = (KeyGen, Eval, Puncture, PunctureEval)$ with key injectivity, key space $\mathcal{K}$, input space $\{0,1\}^n$ and output space $\{0,1\}^m$ consists of four algorithms:* [8]

[7] Here, we follow the concept of "key injectivity" defined in [KW17], which requires that for any different keys k_1 and k_2 and for any input x, $F_{k_1}(x) \neq F_{k_2}(x)$. There are also some different definition of of "key injectivity". For instance, in [CHN+16], the "key injectivity" (we call it "weak key injectivity" in this paper.) requires that for a randomly chosen k_1, the probability that there exists $k_2 \neq k_1$ and x that $F_{k_1}(x) = F_{k_2}(x)$ is negligible; besides, in [FOR17], a computational "key injectivity" that is similar to the notion of "collision resistance" is also defined.

[8] $\mathcal{K}, n, m$ will vary with the security parameter λ, but for simplicity of notation, we do not specify this explicitly in this work.

- ***KeyGen.*** *On input the security parameter λ, the key generation algorithm outputs the secret key $k \in \mathcal{K}$.*
- ***Eval.*** *On input a secret key $k \in \mathcal{K}$ and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs a string $y \in \{0,1\}^m$.*
- ***Puncture.*** *On input a secret keys $k \in \mathcal{K}$ and a polynomial-size set $\mathsf{S} \subseteq \{0,1\}^n$, the puncture algorithm outputs a punctured key ck.*
- ***PunctureEval.*** *On input a punctured key ck and an input $x \in \{0,1\}^n$, the punctured evaluation algorithm outputs an string $y \in \{0,1\}^m \cup \{\perp\}$.*

and satisfies the following conditions:

- ***Correctness.*** *For any $k \in \mathcal{K}$, any polynomial size set $\mathsf{S} \subseteq \{0,1\}^n$, and any $x \in \{0,1\}^n \backslash \mathsf{S}$, let $ck \leftarrow Puncture(k, \mathsf{S})$, then we have $PunctureEval(ck, x) = Eval(k, x)$.*
- ***Key Injectivity.*** *For any $k_1, k_2 \in \mathcal{K}$ that $k_1 \neq k_2$, and any $x \in \{0,1\}^n$, we have $Eval(k_1, x) \neq Eval(k_2, x)$.*
- ***Pseudorandomness.*** *For all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$,*

$$|\Pr[k \leftarrow KeyGen(1^\lambda) : \mathcal{A}^{O_k^{PR}(\cdot)}(1^\lambda) = 1] - \Pr[f \xleftarrow{\$} FUN_{n,m} : \mathcal{A}^{O_f^{R}(\cdot)}(1^\lambda) = 1]| = negl(\lambda)$$

where the oracle $O_k^{PR}(\cdot)$ takes as input a string $x \in \{0,1\}^n$ and returns $Eval(k, x)$, and the oracle $O_f^R(\cdot)$ takes as input a string $x \in \{0,1\}^n$ and returns $f(x)$.
- ***Constrained Pseudorandomness.*** *For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\Pr\left[\begin{array}{l} (\mathsf{S}, \sigma) \leftarrow \mathcal{A}_1(1^\lambda); \\ k \leftarrow KeyGen(1^\lambda); \\ ck \leftarrow Puncture(k, \mathsf{S}); \\ b \xleftarrow{\$} \{0,1\}; \\ \mathtt{Y}_0 = \{Eval(k,x)\}_{x \in \mathsf{S}}; \\ \mathtt{Y}_1 \xleftarrow{\$} (\{0,1\}^n)^{\|\mathsf{S}\|}; \end{array} : \mathcal{A}_2(\sigma, ck, \mathsf{S}, \mathtt{Y}_b) = b \right] \leq 1/2 + negl(\lambda)$$

where $\mathsf{S} \subseteq \{0,1\}^n$ is a polynomial-size set, and σ is the state of $\mathcal{A}_1$.

The translucent t-puncturable PRF constructed in [KW17] is also a t-puncturable PRF with above defined key injectivity, but may be an overkill for our purpose. It also has shortcomings such as computational correctness. So, in Appendix A, we also give a much simpler construction of puncturable PRF satisfying Definition 2.2 from the LWE assumption.

The Indistinguishability Obfuscator. The notion of indistinguishability obfuscator was first proposed by Barak et al. in [BGI+01], and the indistinguishability obfuscator for all polynomial-size circuits was first instantiated by Garg et al. in [GGH+13].

Definition 2.3 ([BGI+01, GGH+13]). *A uniform PPT machine iO is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if it satisfies the following conditions:*

- ***Correctness.*** *For all security parameters $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_\lambda$, and all inputs x, we have that*

$$\Pr[C' \leftarrow iO(C) : C'(x) = C(x)] = 1$$

– ***indistinguishability.*** *For any PPT adversary $\mathcal{A}$, for all security parameters $\lambda \in \mathbb{N}$, and all pairs of circuits $C_0, C_1 \in C_\lambda$ that $C_0(x) = C_1(x)$ for all inputs x, we have that*

$$|\Pr[\mathcal{A}(iO(C_0)) = 1] - \Pr[\mathcal{A}(iO(C_1)) = 1]| \leq negl(\lambda)$$

3 The Definition of Collusion Resistant Watermarkable PRF

In this section, we give the formal definition of the collusion resistant watermarkable PRF, which is adapted from the secretly-extractable message-embedding watermarkable PRF defined in current works [BLW17, KW17].

Definition 3.1 (Watermarkable Family of PRFs [KW17, adapted]). *Let $PRF = (PRF.KeyGen, PRF.Eval)$ be a PRF family with key space $\mathcal{K}$, input space $\{0,1\}^n$ and output space $\{0,1\}^m$. The watermarking scheme with message space $\mathcal{M}$ for PRF (more accurately, the evaluation algorithm of PRF) consists of three algorithms:*

– ***Setup.*** *On input the security parameter λ, the setup algorithm outputs the watermarking secret key msk.*
– ***Mark.*** *On input the watermarking secret key msk, a secret key $k \in \mathcal{K}$ of PRF, and a message $\mathtt{m} \in \mathcal{M}$, the mark algorithm outputs a marked circuit C.*
– ***Extract.*** *On input the master secret key msk and a circuits C, the extraction algorithm outputs a string $\mathtt{m} \in \mathcal{M} \cup \{\perp\}$.*

Definition 3.2 (Watermarking Correctness [KW17, adapted]). *Correctness of the watermarking scheme requires that for any $k \in \mathcal{K}$ and $\mathtt{m} \in \mathcal{M}$, let $msk \leftarrow Setup(1^\lambda)$, $C \leftarrow Mark(msk, k, \mathtt{m})$, we have:*

– ***Functionality Preserving.*** *$C(\cdot) \sim_f PRF.Eval(k, \cdot)$ where $1/f(n)$ is negligible in the security parameter.*
– ***Extraction Correctness.*** *$\Pr[Extract(msk, C) \neq \mathtt{m}] = negl(\lambda)$.*

Before defining the security of the collusion resistant watermarkable PRF, we first define oracles the adversaries can query during the security experiments. Here, the marking oracle is identical to the one defined in [KW17], while we redefine the challenge oracle to capture the scenario that the adversary could see multiple circuits embedded with different messages for the same secret key.

– **Marking Oracle $\mathcal{O}^M_{msk}(\cdot,\cdot)$.** On input a message $\mathtt{m} \in \mathcal{M}$ and a PRF key $k \in \mathcal{K}$, the oracle returns the circuit $C \leftarrow Mark(msk, k, \mathtt{m})$.
– **Challenge Oracle $\mathcal{O}^C_{msk}(\cdot)$.** On input a set $\mathtt{M}$ of messages, the oracle first sample a key $k \leftarrow PRF.KeyGen(1^\lambda)$. Then, for each $\mathtt{m}_i \in \mathtt{M}$, it computes $C_i \leftarrow Mark(msk, k, \mathtt{m}_i)$. Finally, it returns the set $\mathtt{C} = \{C_i\}_{i \in [1, \|\mathtt{M}\|]}$.

Definition 3.3 (Collusion Resistant Unremovability). *The watermarking scheme for a PRF is collusion resistant unremovable if for all PPT and unremoving-admissible adversaries $\mathcal{A}$, $\Pr[ExptUR_{\mathcal{A}}(\lambda) = 1] = negl(\lambda)$, where we define the experiment $ExptUR$ as follows:*

1. *The challenger samples* $msk \leftarrow Setup(1^\lambda)$.
2. *The adversary* $\mathcal{A}$ *is allowed to access the marking oracle, and can query it multiple times.*
3. *The adversary* $\mathcal{A}$ *makes a query* $\mathtt{M}$ *to the challenge oracle and gets a set* $\mathtt{C}$ *of circuits back.*
4. *The adversary* $\mathcal{A}$ *is further allowed to access the marking oracle, and can query it multiple times.*
5. *Finally the adversary submits a circuit* $\tilde{C}$*, and the experiment outputs 1 if and only if* $Extract(msk, \tilde{C}) \notin \mathtt{M}$.

Here, an adversary $\mathcal{A}$ *is unremoving-admissible if with all but negligible probability, its submitted circuit* $\tilde{C}$ *satisfies that there exists circuit* $C \in \mathtt{C}$ *that* $\tilde{C} \sim_f C$ *for negligible* $1/f(n)$.

Definition 3.4 (δ-Unforgeability [KW17, adapted]). *The watermarking scheme for a PRF is* δ*-unforgeable if for all PPT and* δ*-unforging-admissible adversaries* $\mathcal{A}$*, we have* $\Pr[ExptUF_{\mathcal{A}}(\lambda) = 1] = negl(\lambda)$*, where we define the experiment* $ExptUR$ *as follows:*

1. *The challenger samples* $msk \leftarrow Setup(1^\lambda)$.
2. *The adversary* $\mathcal{A}$ *is allowed to access the marking oracle, and can query it multiple times.*
3. *Finally the adversary submits a circuit* $\tilde{C}$*, and the experiment outputs 1 if and only if* $Extract(msk, \tilde{C}) \neq \perp$.

Here, an adversary $\mathcal{A}$ *is* δ*-unforging-admissible if with all but negligible probability, its submitted circuit* $\tilde{C}$ *satisfies that* $\tilde{C} \not\sim_f C_i$ *for all* $i \in [1, Q]$*, where* Q *is the number of queries* $\mathcal{A}$ *made to the marking oracle,* C_i *is the output of the marking oracle on the* i*th query, and* $1/f(n) > \delta$.

4 The Private Multi-Programmable PRF

In this section, we define and construct the main component for constructing collusion resistant watermarkable PRF, namely, the private multi-programmable PRF, which is a variant of the private programmable PRF defined in [BLW17]. As mentioned in Sec. 1.2, compared to the private programmable PRF, the private multi-programmable PRF allows one to puncture and reprogram on multiple inputs and has a consistent privacy. Besides, we also require it to have the "key-injectivity" property to enable "decentralized PRF key generation" when constructing collusion resistant watermarkable PRF.

4.1 The Definition

Definition 4.1 (Private Multi-Programmable PRF). *A private multi-programmable PRF with key space* $\mathcal{K}$*, input space* $\{0,1\}^n$ *and output space* $\{0,1\}^m$ *consists of four algorithms:*

- ***KeyGen.*** *On input the security parameter* λ*, the key generation algorithm outputs the secret key* k.

- ***Eval.*** *On input a secret key k and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs a string $y \in \{0,1\}^m$.*
- ***Constrain.*** *On input a secret keys k and a polynomial-size set $\mathsf{S} \subseteq \{0,1\}^n \times \{0,1\}^m$, where for any distinct $(x,y),(x',y') \in \mathsf{S}$ we have $x \neq x'$[9], the constrain algorithm outputs a constrained key ck.*
- ***ConstrainEval.*** *On input a constrained key ck and an input $x \in \{0,1\}^n$, the constrained evaluation algorithm outputs an string $y \in \{0,1\}^m$.*

Definition 4.2 (Correctness). *Correctness of the private multi-programmable PRF requires that for any $k \in \mathcal{K}$, any $x \in \{0,1\}^n$, and any valid set $\mathsf{S} \subseteq \{0,1\}^n \times \{0,1\}^m$, let $ck \leftarrow Constrain(k, \mathsf{S})$, we have:*

- *If there exists $y \in \{0,1\}^m$ that $(x,y) \in \mathsf{S}$, then $ConstrainEval(ck, x) = y$.*
- *Otherwise, $ConstrainEval(ck, x) = Eval(k, x)$.*

Definition 4.3 (Key Injectivity). *Key Injectivity of the private multi-programmable PRF requires that for any $k_1, k_2 \in \mathcal{K}$ that $k_1 \neq k_2$, and any $x \in \{0,1\}^n$, we have $Eval(k_1, x) \neq Eval(k_2, x)$.*

Definition 4.4 (Pseudorandomness). *The private multi-programmable PRF is pseudorandom if for all PPT adversary $\mathcal{A}$,*

$$|\Pr[k \leftarrow KeyGen(1^\lambda) : \mathcal{A}^{O_k^{PR}(\cdot)}(1^\lambda) = 1] - \Pr[f \xleftarrow{\$} FUN_{n,m} : \mathcal{A}^{O^R(\cdot)}(1^\lambda) = 1]| = negl(\lambda)$$

where the oracle $O_k^{PR}(\cdot)$ takes as input a string $x \in \{0,1\}^n$ and returns $Eval(k,x)$, and the oracle $O_f^R(\cdot)$ takes as input a string $x \in \{0,1\}^n$ and returns $f(x)$.

Definition 4.5 (Selectively Constrained Pseudorandomness). *We say a private multi-programmable PRF is selectively constrained pseudorandom if for all PPT and constraining-admissible adversaries $\mathcal{A}$, $\Pr[ExptCPRF_{\mathcal{A}}(\lambda) = 1] \leq 1/2 + negl(\lambda)$, where we define the experiment $ExptCPRF$ as follows:*

1. *The challenger samples $k \leftarrow KeyGen(1^\lambda)$ and $b \xleftarrow{\$} \{0,1\}$. It also samples $f \xleftarrow{\$} FUN_{n,m}$.*
2. *The adversary $\mathcal{A}$ is required to submit its challenge input $x^* \in \{0,1\}^n$ in the beginning, and gets a string $y^* \in \{0,1\}^m$ back. If $b = 0$, $y^* = Eval(k,x)$ and if $b = 1$, $y^* = f(x)$.*
3. *Then, the adversary is allowed to access the following two oracles:*
   - ***Constrain Oracle.*** *On input a valid set $\mathsf{S} \subseteq \{0,1\}^n \times \{0,1\}^m$, the oracle returns a constrained key $ck \leftarrow Constrain(k, \mathsf{S})$.*
   - ***Evaluation Oracle.*** *On input an input $x \in \{0,1\}^n$, the oracle returns an output $y = Eval(k,x)$.*
4. *Eventually, $\mathcal{A}$ outputs a bit b' and the experiment outputs 1 iff $b = b'$.*

Here, an adversary $\mathcal{A}$ is constraining-admissible if for each set S submitted to the constrain oracle, there exists $y \in \{0,1\}^m$ that $(x^*, y) \in \mathsf{S}$ and for each input x submitted to the evaluation oracle, we have $x^* \neq x$.

[9] Here, we denote sets statisfying this condition as "valid sets"

Definition 4.6 (Selectively Consistent Privacy). *The private multi-programmable PRF is selectively consistently private if for all PPT and privacy-admissible adversaries $\mathcal{A}$, $|\Pr[ExptPri_{0,\mathcal{A}}(\lambda) = 1] - \Pr[ExptPri_{1,\mathcal{A}}(\lambda) = 1]| \leq negl(\lambda)$, where we define the experiment $ExptPri_b$ as follows:*

1. *In the beginning of the experiment, the adversary $\mathcal{A}$ first submits a list of $2Q$ polynomial size sets $(\mathtt{X}_{1,0}, \mathtt{X}_{1,1}, \ldots, \mathtt{X}_{Q,0}, \mathtt{X}_{Q,1})$ to the challenger (Q is a polynomial determined by $\mathcal{A}$ and is not preciously fixed in the experiment), where each $\mathtt{X}_{i,j} \subseteq \{0,1\}^n$.*
2. *The challenger samples $k \leftarrow KeyGen(1^\lambda)$. It also samples $f \xleftarrow{\$} FUN_{n,m}$.*
3. *Then the challenger generates $\mathtt{S}_{i,j} = \{(x, f(x)) \mid x \in \mathtt{X}_{i,j}\}$ for $i \in [1, Q]$ and $j \in \{0, 1\}$, computes $ck_i \leftarrow Constrain(k, \mathtt{S}_{i,b})$, and returns $(ck_1, \ldots ck_Q)$ to $\mathcal{A}$.*
4. *Then, the adversary is allowed to access the following oracle:*
   - ***Evaluation Oracle.*** *On input an input $x \in \{0,1\}^n$, the oracle returns an output $y = Eval(k, x)$.*
5. *Eventually, $\mathcal{A}$ outputs a bit b' which is also the output of the experiment.*

Here, for any $x \in \{0,1\}^n$, $i \in [1, Q]$ and $j \in \{0,1\}$, we define $d_{x,i,j} = 1$ if $x \in \mathtt{X}_{i,j}$, and $d_{x,i,j} = 0$ otherwise. Then, an adversary $\mathcal{A}$ is privacy-admissible if (1) for any $x \in \{0,1\}^n$, and for any $i, j \in [1, Q]$, $d_{x,i,0} \oplus d_{x,i,1} \oplus d_{x,j,0} \oplus d_{x,j,1} = 0$; (2) for any x that has been submitted to the evaluation oracle, and any $i \in [1, Q]$, $d_{x,i,0} = d_{x,i,1}$.

Remark 4.1. Recall that in previous private constrained PRFs, the privacy-admissibility requires that (1) for any $x \in \{0,1\}^n$, and for any $i, j \in [1, Q]$, $d_{x,i,0} \vee d_{x,j,0} = d_{x,i,1} \vee d_{x,j,1}$; (2) for any x that has been submitted to the evaluation oracle, and any $i \in [1, Q]$, $d_{x,i,0} = d_{x,i,1}$. It immediately implies that when given multiple punctured keys, the punctured inputs cannot be hidden (the indistinguishability still holds, but each punctured inputs will appear in both side). More precisely, here we regard the case that $b = 0$ as the real world case and the case that $b = 1$ as the ideal case. So to hide a real world punctured input x (w.l.o.g. we assume $d_{x,1,0} = 1$), we need to establish the indistinguishability on condition that $d_{x,1,1} = \ldots = d_{x,Q,1} = 0$. This requirement contradicts previous privacy-admissibility as for any $i \in [1, Q]$, $1 = d_{x,1,0} \vee d_{x,i,0} \neq d_{x,1,1} \vee d_{x,i,1} = 0$. In contrast, the consistent privacy-admissibility does not suffer this problem. In particular, as long as x is punctured in all punctured keys in the real world, for any $i, j \in [1, Q]$, $0 = d_{x,i,0} \oplus d_{x,j,0} = d_{x,i,1} \oplus d_{x,j,1} = 0$, which satisfies the consistent privacy-admissibility. For more discussion on these two types of privacy definition, we refer the readers to [BLW17].

4.2 The Construction

In this section, we upgrade the private programmable PRF constructed in [BLW17] to the multi-punctured-inputs setting.

Let $PPRF = (PPRF.KeyGen, PPRF.Eval, PPRF.Puncture, PPRF.PunctureEval)$ be a puncturable PRF with key injectivity property, key space $\mathcal{K}$, input space $\{0,1\}^n$, and output space $\{0,1\}^m$. Let iO be an indistinguishability obfuscator for all polynomial-size circuits. Then, our private multi-programmable PRF $MPRF$ with key space $\mathcal{K}$, input space $\{0,1\}^n$ and output space $\{0,1\}^m$ works as follows:

– **KeyGen.** On input a security parameter λ, the key generation algorithm outputs $k \leftarrow PPRF.KeyGen(1^\lambda)$.
– **Eval.** On input a secret key $k \in \mathcal{K}$ and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs $y = PPRF.Eval(k, x)$.
– **Constrain.** On input a secret key $k \in \mathcal{K}$ and a polynomial size set S, the constrain algorithm outputs the circuit $C \leftarrow iO(C^1[k, \mathsf{S}])$, where C^1 is defined in Figure 1. [10]
– **ConstrainEval.** On input a constrained secret key ck and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs $y = ck(x)$.

Constant: a secret key k and a polynomial size set S.
Input: $x \in \{0,1\}^n$

1. If there eixsts $y \in \{0,1\}^m$ that $(x, y) \in \mathsf{S}$, outputs y.
2. Otherwise, outputs $PPRF.Eval(k, x)$.

Fig. 1 The circuit C^1.

Theorem 4.1. *If PPRF is a secure puncturable PRF with key injectivity, and iO is a secure indistinguishability obfuscator for all polynomial-size circuits, then MPRF is a secure private multi-programmable PRF as defined in 4.1.*

Proof. The correctness of $MPRF$ comes from correctness of iO directly, and the key injectivity and the pseudorandomness of $MPRF$ come from the key injectivity and the pseudonmess of $PPRF$ respecitvely. It remains to prove the selectively constrained pseudorandomness and the selective consistent privacy of $MPRF$.

Proof of selectively constrained pseudorandomness. To prove the selectively constrained pseudorandomness of $MPRF$, we define the following two games:

– **Game 0.** This is the real experiment $ExptCPRF$.
– **Game 1.** This is identical to Game 0 except that the challenger generates $ck = PPRF.Puncture(k, x^*)$ after $\mathcal{A}$ submits the challenge input x^*, and uses ck instead of k to answer the evaluation oracle queries. Besides, it returns $C \leftarrow iO(C^2[ck, \mathsf{S}])$ when $\mathcal{A}$ quries the constrain oracle with S, where C^2 is defined in Figure 2.

Indistinguishability of Game 0 and Game 1 comes from the correctness of $PPRF$ and the indistinguishability of iO. In particular, first, as $\mathcal{A}$ is not allowed to submit x^* in an evaluation oracle query, by the correctness of the $PPRF$, the evaluation oracle is answered identically in Game 0 and Game 1. Also, as for each S submitted to the constrain oracle, there exists y that $(x^*, y) \in \mathsf{S}$, ck is not required to compute on x^* in each

[10] Note that the circuit C_1, as well as all circuits appeared in the proof of Theorem 4.1, will be padded to the same size.

circuit $C^2[ck, \mathtt{S}]$. So, by the correctness of the $PPRF$, circuit $C^1[k, \mathtt{S}]$ and $C^2[ck, \mathtt{S}]$ are identically evaluated. Thus, by the indistinguishability of iO, the views of $\mathcal{A}$ in Game 0 and in Game 1 are computationally indistinguishable.

In Game 1, as the view of $\mathcal{A}$ (apart from the challenge y^*) can be simulated with ck (instead of k), by the constrained pseudorandomness of $PPRF$, the experiment outputs 1 with a probability of $1/2+negl(\lambda)$. That completes the proof of selectively constrained pseudorandomness of $MPRF$.

Constant: a constrained secret key ck, and a polynomial size set $\mathtt{S}$.
Input: $x \in \{0,1\}^n$

1. If there eixsts $y \in \{0,1\}^m$ that $(x, y) \in \mathtt{S}$, outputs y.
2. Otherwise, outputs $PPRF.PunctureEval(ck, x)$.

Fig. 2 The circuit C^2

Proof of selective consistent privacy. To prove the selective consistent privacy of $MPRF$, we first define $L_r = \|\{x \mid x \in \{0,1\}^n \wedge \exists i \in [1, Q] d_{x,i,0} \neq d_{x,i,1}\}\|$, where Q and $d_{x,i,j}$ is the variable in an experiment $ExptPri$ launched by an adversary $\mathcal{A}$ with internal randomness r. Let L be the maximum L_r for all r in the randomness space of $\mathcal{A}$. Since $\mathcal{A}$ could only submit $2Q$ polynomial size sets for a polynomial Q, L is also polynomial in λ. Next we define **Game 0** as the the real experiment $ExptPri_0$, and define **Game 1** as the the real experiment $ExptPri_1$. Note that to sample and evaluate the random function f, the challenger could employ the lazy sampling method, namely, sample a uniform $y \in \{0,1\}^m$ for each distinct x if needed. We also define the following hybrid games for $i \in [0, L]$.

Game H_i. In Game H_i, on input the $2Q$ challenge sets, the challenger first generates the lexicographical order list $\mathtt{D}$ containing all $x \in \{0,1\}^n$ that $\exists j \in [1, Q], d_{x,j,0} \neq d_{x,j,1}$. It also generates the set $\mathtt{D}_i$ containing the first i elements in $\mathtt{D}$ ($\mathtt{D}_0 = \emptyset$). Then it proceeds identically as in the epxeriment $ExptPri$ except generating the challenge constrained keys. In particular, to generate a challenge constrained secret key ck_j, the challenger computes $ck_j \leftarrow iO(C^3[k, \mathtt{S}_{j,0}, \mathtt{S}_{j,1}, \mathtt{D}_i])$, where C^3 is defined in Figure 3.

Obviously, the circuit $C^3[k, \mathtt{S}_{j,0}, \mathtt{S}_{j,1}, \mathtt{D}_0]$ is identically evaluated to the circuit $C^1[k, \mathtt{S}_{j,0}]$. Also, as for any $x \notin \mathtt{D}$ and any $j \in [1, Q]$, $d_{x,j,0} = d_{x,j,1}$, the circuit $C^3[k, \mathtt{S}_{j,0}, \mathtt{S}_{j,1}, \mathtt{D}_L]$ is identically evaluated to the circuit $C^1[k, \mathtt{S}_{j,1}]$. Thus, by the indistinguishability of iO, Game 0 is computationally indistinguishable from Game H_0, and Game 1 is computationally indistinguishable from Game H_L. It remains to argue the indistinguishability between Game H_i and Game H_{i+1} for $i \in [0, L]$, and we prove this by defining the following games.

• *Game $H_{i,1}$.* This is identical to Game H_i except that the challenger sets $\hat{x}$ as the $(i+1)$th element in $\mathtt{D}$, computes $ck \leftarrow PPRF.Puncture(k, \hat{x})$, $\hat{y} = PPRF.Eval(k, \hat{x})$ and sam-

Constant: a secret keys k, two polynomial size sets $\mathsf{S}_0, \mathsf{S}_1$, and a set D_i.
Input: $x \in \{0,1\}^n$

1. If $x \in \mathsf{D}_i$, $e = 1$; else, $e = 0$.
2. If there eixsts $y \in \{0,1\}^m$ that $(x, y) \in \mathsf{S}_e$, outputs y.
3. Otherwise, outputs $PPRF.Eval(k, x)$.

Fig. 3 The circuit C^3.

ples $\hat{y}'$ uniformly at random after sampling the secret key k and the random function f. Then, to generate a challenge constrained secret key ck_j, the challenger computes $ck_j \leftarrow iO(C^4[ck, \hat{y}, \hat{y}', \mathsf{S}'_{j,0}, \mathsf{S}'_{j,1}, \mathsf{D}_i, \hat{x}, d_{\hat{x},j,0}])$, where C^4 is defined in Figure 4, $\mathsf{S}'_{j,0}$ is generated identically to $\mathsf{S}_{j,0}$ except that it will not include $(\hat{x}, *)$, and $\mathsf{S}'_{j,1}$ is also generated identically to $\mathsf{S}_{j,1}$ except that it will not include $(\hat{x}, *)$. It also uses ck to answer the evluation oracle, Note that, here we set the value of $f(\hat{x})$ as $\hat{y}$ implicitly.

By the condition 2 of the privacy-admissiblility of $\mathcal{A}$, it will not submit $\hat{x}$ to the evaluation oracle. Also, by the correctness of $PPRF$, the circuit $C^4[ck, \hat{y}, \hat{y}', \mathsf{S}'_{j,0}, \mathsf{S}'_{j,1}, \mathsf{D}_i, \hat{x}, d_{\hat{x},j,0}]$ is identically evaluated to the circuit $C^3[k, \mathsf{S}_{j,0}, \mathsf{S}_{j,1}, \mathsf{D}_i]$. Thus, indistinguishability between Game H_i and Game $H_{i,1}$ comes from the correctness of $PPRF$ and the indistinguishability of iO directly.

Constant: a constrained secret key ck, two strings $\hat{y}, \hat{y}'$, two polynomial size sets $\mathsf{S}_0, \mathsf{S}_1$, a set D_i, a string $\hat{x}$ and a bit d.
Input: $x \in \{0,1\}^n$

1. If $x = \hat{x}$ and $d = 1$, outputs $\hat{y}'$.
2. If $x = \hat{x}$ and $d = 0$, outputs $\hat{y}$.
3. If $x \in \mathsf{D}_i$, $e = 1$; else, $e = 0$.
4. If there eixsts $y \in \{0,1\}^m$ that $(x, y) \in \mathsf{S}_e$, outputs y.
5. Otherwise, outputs $PPRF.PunctureEval(ck, x)$.

Fig. 4 The circuit C^4.

• *Game* $H_{i,2}$. This is identical to Game $H_{i,1}$ except that $\hat{y}$ is sampled uniformly at random from $\{0,1\}^m$.

Indistinguishability between Game $H_{i,1}$ and Game $H_{i,2}$ comes from the constrained pseudorandomness of $PPRF$ directly.

• *Game* $H_{i,3}$. This is identical to Game $H_{i,2}$ except that $\hat{y}' = PPRF.Eval(k, \hat{x})$.

Indistinguishability between Game $H_{i,2}$ and Game $H_{i,3}$ comes from the constrained pseudorandomness of $PPRF$ directly.

• *Game* $H_{i,4}$. This is identical to Game $H_{i,3}$ except that when generating a challenge constrained secret key ck_j, the challenger computes $ck_j \leftarrow iO(C^5[k, \hat{y}, \mathtt{S}_{j,0}, \mathtt{S}_{j,1}, \mathtt{D}_i, \hat{x}, d_{\hat{x},j,0}])$, where the value of $f(\hat{x})$ is set to be $\hat{y}$ and C^5 is defined in Figure 5. Besides, each evaluation oracle is answered with k.

By the condition 2 of the privacy-admissiblility of $\mathcal{A}$, it will not submit $\hat{x}$ to the evaluation oracle. Also, by the correctness of $PPRF$, the circuit $C^5[k, \hat{y}, \mathtt{S}_{j,0}, \mathtt{S}_{j,1}, \mathtt{D}_i, \hat{x}, d_{\hat{x},j,0}])$ is identically evaluated to the circuit $C^4[ck, \hat{y}, \hat{y}', \mathtt{S}'_{j,0}, \mathtt{S}'_{j,1}, \mathtt{D}_i, \hat{x}, d_{\hat{x},j,0}]$. Thus, indistinguishability between Game $H_{i,3}$ and Game $H_{i,4}$ comes from the correctness of $PPRF$ and the indistinguishability of iO directly.

Note that, by the condition 1 of the privacy-admissiblility of $\mathcal{A}$, for any $x \in \{0,1\}^n$ if there exists $j \in [1, Q]$ that $d_{x,j,0} \neq d_{x,j,1}$, then for any $j' \in [1, Q]$, $d_{x,j',0} \neq d_{x,j',1}$. Thus, $d_{\hat{x},j,0} \neq d_{\hat{x},j,1}$ for all $j \in [1, Q]$, so the circuit $C^5[k, \hat{y}, \mathtt{S}_{j,0}, \mathtt{S}_{j,1}, \mathtt{D}_i, \hat{x}, d_{\hat{x},j,0}])$ and the circuit $C^3[k, \mathtt{S}_0, \mathtt{S}_1, \mathtt{D}_{i+1}]$ are identically evaluated. Thus, Game $H_{i,4}$ is indistinguishable from Game H_{i+1} by the indistinguishability of iO. That completes the proof.

Constant: a secret key k, a string $\hat{y}$, two polynomial size sets $\mathtt{S}_0, \mathtt{S}_1$, a set $\mathtt{D}_i$, a string $\hat{x}$ and a bit d.
Input: $x \in \{0,1\}^n$

1. If $x = \hat{x}$ and $d = 1$, outputs $PPRF.Eval(k, \hat{x})$.
2. If $x = \hat{x}$ and $d = 0$, outputs $\hat{y}$.
3. If $x \in \mathtt{D}_i$, $e = 1$; else, $e = 0$.
4. If there eixsts $y \in \{0,1\}^m$ that $(x, y) \in \mathtt{S}_e$, outputs y.
5. Otherwise, outputs $PPRF.Eval(k, x)$.

Fig. 5 The circuit C^5.

□

5 Collusion Resistant Watermarkable PRF from Private Multi-Programmable PRF

In this section, we show how to obtain collusion resistant watermarkable PRF from any private multi-programmable PRF via constructing a watermarking scheme for its evaluation algorithm.

Let λ be the security parameter. Let N be a postive integer that is polynomial in λ, $t = (N+1)^2 \cdot \omega(\log \lambda)$. Let δ be a positive real value and $d = \lambda/\delta = poly(\lambda)$. Let $MPRF = (MPRF.KeyGen, MPRF.Eval, MPRF.Constrain, MPRF.ConstrainEval)$ be a secure private multi-programmable PRF with key space $\{0,1\}^\ell$, input space $\{0,1\}^n$, and

output space $\{0,1\}^m$, where n is polynomial in λ. Let $F : \mathcal{K} \times (\{0,1\}^{md}) \to \{0,1\}^{Ntn+\ell}$ be a secure PRF. Then, our watermarking scheme WM with message space $[1, N]$ for $MPRF$ works as follows:

- **Setup.** On input a security parameter λ, the setup algorithm first samples $K \xleftarrow{\$} \mathcal{K}$, and $(z_1, \ldots, z_d) \xleftarrow{\$} (\{0,1\}^n)^d$. Then it outputs $msk = (K, z_1, \ldots, z_d)$.
- **Mark.** On input a watermarking secret key $msk = (K, z_1, \ldots, z_d)$, a secret key $k \in \{0,1\}^\ell$ for $MPRF$ and a message $\mathtt{m} \in [1, N]$, the mark algorithm first computes $w = (MPRF.Eval(k, z_1), \ldots, MPRF.Eval(k, z_d))$, then it computes $(\{x_{i,j}\}_{i\in[1,N],j\in[1,t]}, k') = F(K, w)$. Next it generates the set $\mathtt{S} = \{(x_{i,j}, MPRF.Eval(k', x_{i,j}))\}_{i\in[1,\mathtt{m}],j\in[1,t]}$, and computes $ck \leftarrow MPRF.Constrain(k, \mathtt{S})$. Finally, it outputs the circuit $C(\cdot) = MPRF.ConstrainEval(ck, \cdot)$.
- **Extract.** On input a watermarking secret key $msk = (K, z_1, \ldots, z_d)$, and a circuit C, the extraction algorithm first computes $w = (C(z_1), \ldots, C(z_d))$, then it computes $(\{x_{i,j}\}_{i\in[1,N],j\in[1,t]}, k') = F(K, w)$. Next, for $i \in [1, N]$, it computes $\alpha_i = \|\{j \mid C(x_{i,j}) = MPRF.Eval(k', x_{i,j})\}\|$. It also sets $\alpha_0 = t$ and $\alpha_{N+1} = 0$. Then it finds the first i^* satisfying $\alpha_{i^*} - \alpha_{i^*+1} \geq \frac{t}{N+1}$, and outputs $\perp$ if $i^* = 0$ and outputs i^* otherwise. Note that, by the pigeonhole principle, such i^* always exists.

Remark 5.1. Since $MPRF$ is in fact a puncturable PRF, WM is also a watermarking scheme for the evaluation algorithm of a puncturable PRF. So, here we also construct a collusion resistant watermarkable puncturable PRF, whose definition is similar to the the definition of the collusion resistant watermarkable normal PRF defined in Sec. 3. Note that unlike the puncturable marked PRF defined in [NW15], we need to neither mark a punctured key nor puncture a marked key here.

Theorem 5.1. *If $MPRF$ is a secure private multi-programmable PRF as defined in Sec. 4.1, and F is a secure PRF, then WM is a secure watermarking scheme with collusion resistant unremovability and δ-unforgeability for $MPRF$.*

Proof. To prove that WM is a secure watermarking scheme for $MPRF$, we need to prove that it has the functionality preserving, the extraction correctness, the collusion resistant unremovability, and the δ-unforgeability.

Proof of Correctness. Functionality preserving of WM comes from the correctness of $MPRF$ and the fact that the set $\mathtt{S}$ used in the mark algorithm contains at most Nt elements, which is negligible compared with the size of the input space (2^n).

Next, we prove the extraction correctness of WM. Let $\hat{w}, \{\hat{x}_{i,j}\}_{i\in[1,N],j\in[1,t]}, \hat{k}'$ be internal variables used when generating the marked circuit C by embedding a message $\mathtt{m}$ into the secret key k, and $\check{w}, \{\check{x}_{i,j}\}_{i\in[1,N],j\in[1,t]}, \check{k}'$ be internal variables used when extracting C. By the pseudorandomness of F, the probability that there exists i, j, h that $\hat{x}_{i,j} = z_h$ is negligible. So, by the correctness of $MPRF$, $\hat{w} = \check{w}$ with all but negligible probability, which indicates that $\hat{k}' = \check{k}'$ and $\forall i \in [1, N]$, $j \in [1, t]$, $\hat{x}_{i,j} = \check{x}_{i,j}$ with all but negligible probability. Again, by the pseudorandomness of F, the probability that $\check{k}' = k$, and the probability that there exists i, j, i', j' that $(i, j) \neq (i', j')$ but $\check{x}_{i,j} = \check{x}_{i',j'}$ are also negligible. So, by the correctness and the key injectivity of $MPRF$, $\alpha_1 = \ldots = \alpha_{\mathtt{m}} = t$ and $\alpha_{\mathtt{m}+1} = \ldots = \alpha_{N+1} = 0$ with all but negligible probability. In summary, the probability that the output of the extraction algorithm is not $\mathtt{m}$ is negligible.

Proof of Collusion Resistant Unremovability. To prove the unremovability of WM, we define the following games between a challenger and a PPT unremoving-admissible adversary $\mathcal{A}$:

• *Game 0.* This is the real experiment $ExptUR$. More precisely, the challenger proceeds as follows.

1. The challenger samples $K \xleftarrow{\$} \mathcal{K}$ and $(z_1, \ldots, z_d) \xleftarrow{\$} (\{0,1\}^n)^d$.
2. Then, it answers marking oracle queries from $\mathcal{A}$, and each time on receiving a query $(k, \mathtt{m}) \in \{0,1\}^\ell \times [1, N]$, it first computes $w = (MPRF.Eval(k, z_1), \ldots, MPRF.Eval(k, z_d))$, computes $v = (\{x_{i,j}\}_{i\in[1,N],j\in[1,t]}, k') = F(K, w)$, generates the set $\mathtt{S} = \{(x_{i,j}, MPRF.Eval(k', x_{i,j}))\}_{i\in[1,\mathtt{m}],j\in[1,t]}$, computes $ck \leftarrow MPRF.Constrain(k, \mathtt{S})$ and returns the circuit $C(\cdot) = MPRF.ConstrainEval(ck, \cdot)$.
3. Once $\mathcal{A}$ queries the marking oracle with a set $\hat{\mathtt{M}} = \{\hat{\mathtt{m}}_i\}_{i\in[1,Q]}$ for some polynomial Q (w.l.o.g, we assume that the Q messages in $\hat{\mathtt{M}}$ are sorted, namely, $\hat{\mathtt{m}}_1 < \hat{\mathtt{m}}_2 < \ldots < \hat{\mathtt{m}}_Q$), the challenger samples $\hat{k} \xleftarrow{\$} \{0,1\}^\ell$, computes $\hat{w} = (MPRF.Eval(\hat{k}, z_1), \ldots, MPRF.Eval(\hat{k}, z_d))$, and computes $\hat{v} = (\{\hat{x}_{i,j}\}_{i\in[1,N],j\in[1,t]}, \hat{k}') = F(K, \hat{w})$. Then for $h \in [1, Q]$, it generates the set $\hat{\mathtt{S}}_h = \{(\hat{x}_{i,j}, MPRF.Eval(\hat{k}', \hat{x}_{i,j}))\}_{i\in[1,\hat{\mathtt{m}}_h],j\in[1,t]}$, computes $\hat{ck}_h \leftarrow MPRF.Constrain(\hat{k}, \hat{\mathtt{S}}_h)$ and returns the circuit $\hat{C}_h(\cdot) = MPRF.ConstrainEval(\hat{ck}_h, \cdot)$.
4. Then it answers marking oracle queried by $\mathcal{A}$, just as in Phase 2.
5. Finally, on input a circuit $\tilde{C}$ from $\mathcal{A}$, it computes $\check{w} = (\tilde{C}(z_1), \ldots, \tilde{C}(z_d))$ and $\check{v} = (\{\check{x}_{i,j}\}_{i\in[1,N],j\in[1,t]}, \check{k}') = F(K, \check{w})$, and for $i \in [1, N]$, it computes $\alpha_i = \|\{j \mid \tilde{C}(\check{x}_{i,j}) = MPRF.Eval(\check{k}', \check{x}_{i,j})\}\|$. It also sets $\alpha_0 = t$ and $\alpha_{N+1} = 0$ and finds the first i^* satisfying $\alpha_{i^*} - \alpha_{i^*+1} \geq \frac{t}{N+1}$. It outputs 1 iff $i^* \notin \hat{\mathtt{M}}$.

• *Game 1.* This is identical to Game 0 except that the challenger uses a random function $f(\cdot) \xleftarrow{\$} FUN_{md,Ntn+\ell}$ instead of the pseudorandom function $F(K, \cdot)$ during the experiment. Note that the challenger could compute $f(\cdot)$ via the lazy sampling. More precisely, it maintains a table $\{(w_i, v_i)\}_{i\in[1,T]} \subset \{0,1\}^{md} \times \{0,1\}^{Ntn+\ell}$, where T is the current table size. The challenger will search the table each time computing $f(\cdot)$ on an input w, and set the result as v_i if there exists (w_i, v_i) in the table that $w_i = w$. Otherwise, it samples a fresh $v \xleftarrow{\$} \{0,1\}^{Ntn+\ell}$, stores (w, v) in the table and sets the result to be v.

Indistinguishability of Game 0 and Game 1 comes from the pseudorandomness of F directly.

• *Game 2.* This is identical to Game 1 except that the way the challenger "computes" f. More precisely, it maintains a table $\{(k_i, v_i)\}_{i\in[1,T]} \subseteq \{0,1\}^\ell \times \{0,1\}^{Ntn+\ell}$, where T is the current table size. When answering a mark oracle query $(k, \mathtt{m})$, the challenger first searches k in the table; it sets the result of $f(\cdot)$ to be v_i if there exists (k_i, v_i) in the table that $k_i = k$; otherwise, it samples a fresh $v \xleftarrow{\$} \{0,1\}^{Ntn+\ell}$, stores (k, v) in the table and sets the result to be v. Also, when answering the challenge oracle, after sampling the challenge key $\hat{k} \xleftarrow{\$} \{0,1\}^\ell$, the challenger searches $\hat{k}$ in the table; it sets the result of $f(\cdot)$ to be v_i if there exists (k_i, v_i) in the table that $k_i = k$; otherwise, it samples a fresh $v \xleftarrow{\$} \{0,1\}^{Ntn+\ell}$, stores $(\hat{k}, v)$ in the table and sets the result to be v. Besides, at the beginning of Phase 5, assuming the current table is of size T, for $i \in [1, T]$, the

challenger computes $w_i = (MPRF.Eval(k_i, z_1), \ldots, MPRF.Eval(k_i, z_d))$. It aborts and outputs "2" if there exists $i, j \in [1, T]$ that $k_i \neq k_j$ and $w_i = w_j$ (here, We call this event as "Bad_1"). Finally, when computing the function $f(\check{w})$ in Phase 5, the challenger sets $f(\check{w})$ to be v_i if there exists $i \in [1, T]$ that $w_i = \check{w}$; otherwise, it samples $v \xleftarrow{\$} \{0,1\}^{Ntn+\ell}$ and sets the result to be v.

Notice that Game 1 and Game 2 are identical if the event Bad_1 does not occur in Game 2. By the key injectivity of the $MPRF$, $\Pr[Bad_1] = 0$ in Game 2, thus the views of $\mathcal{A}$ in Game 1 and Game 2 are identically distributed.

• *Game 3.* This is identical to Game 2 except that after checking whether Bad_1 occurs, the challenger further checks if Bad_2 occurs, where Bad_2 occurs iff there exists $i \in [1, Q]$, $j \in [1, d]$ that $\hat{C}_i(z_j) \neq MPRF.Eval(\hat{k}, z_j)$. The experiment aborts and outputs 3 if Bad_2 occurs and proceeds identically to Game 2 otherwise.

Game 2 and Game 3 are identical as long as Bad_2 does not occur. Note that in Game 2 and in Game 3, each z_j is sampled uniformly and is independent from the generation of these $\hat{C}_i$s. Also, each $\hat{C}_i(\cdot)$ differs from $MPRF.Eval(\hat{k}, \cdot)$ on only a negligible fraction of inputs. Thus, for every $i \in [1, Q]$, $j \in [1, d]$, $\Pr[\hat{C}_i(z_j) \neq MPRF.Eval(\hat{k}, z_j)] = negl(\lambda)$. By the union bound, $\Pr[Bad_2] = negl(\lambda)$ in Game 3.

• *Game 4.* This is identical to Game 3 except that the challenger further checks if Bad_3 occurs after checking the occurrence of Bad_2, where Bad_3 occurs iff there exists $j \in [1, d]$ that $\tilde{C}(z_j) \neq MPRF.Eval(\hat{k}, z_j)$. The experiment aborts and outputs 4 if Bad_3 occurs. Besides, it also sets $f(\check{w}) = \hat{v} = (\{\hat{x}_{i,j}\}_{i\in[1,N], j\in[1,t]}, \hat{k}')$ directly in Phase 5.

Game 3 and Game 4 are identical as long as Bad_3 does not occur. Note that in Game 3 and in Game 4, each z_j is sampled uniformly and is independent from the view of $\mathcal{A}$. Also, by the unremoving-admissibility of $\mathcal{A}$, with all but negligible probability, there exists i that $\tilde{C}(\cdot)$ differs from $\hat{C}_i(\cdot)$ on only a negligible fraction of inputs. Thus, with all but negligible probability, $\exists i, \forall j, \tilde{C}(z_j) = \hat{C}_i(z_j)$. Also, since Bad_2 does not occur when checking Bad_3, for every $i \in [1, Q]$, $j \in [1, d]$, $\hat{C}_i(z_j) = MPRF.Eval(\hat{k}, z_j)$. Therefore, $\Pr[Bad_3] = negl(\lambda)$ in Game 4.

• *Game 5.* This is identical to Game 4 except that when answering the challenge oracle, the challenger samples a fresh $\hat{v} \xleftarrow{\$} \{0,1\}^{Ntn+\ell}$ directly without searching $\hat{k}$ in the table, and it will not put the tuple $(\hat{k}, \hat{v})$ into the table, either.

Game 4 and Game 5 are identical unless $\mathcal{A}$ is able to submit $\hat{k}$ to the mark oracle, which occurs with only a negligible probability by the selectively constrained pseudorandomness of $MPRF$.

• *Game 6.* This is identical to Game 5 except that the challenger sets $\alpha_i = \|\{j \mid \tilde{C}(\hat{x}_{i,j}) = \hat{C}_Q(\hat{x}_{i,j})\}\|$ for $i \in [1, \hat{\mathfrak{m}}_Q]$ in Phase 5.

By the correctness of $MPRF$, Game 5 and Game 6 are identical.

• *Game 7.* This is identical to Game 6 except that the challenger sets $\alpha_i = 0$ for $i \in [\hat{\mathfrak{m}}_Q + 1, N]$ in Phase 5.

It is sufficient to argue that in Game 6, we also have $\alpha_i = 0$ for $i \in [\hat{\mathfrak{m}}_Q + 1, N]$ with all but negligible probability. First, by the unremoving-admissibility of $\mathcal{A}$ and the fact that for all $i \in [1, Q]$, $\hat{C}_i$ and $MPRF.Eval(\hat{k}, \cdot)$ differs on only polynomial inputs, with all but negligible probability, $\tilde{C}(\cdot)$ differs from $MPRF.Eval(\hat{k}, \cdot)$ on only a negligible fraction of inputs. Also, note that the value of $\hat{x}_{i,j}$ for $i > \hat{\mathfrak{m}}_Q$ is sampled uniformly and is independent of the view of $\mathcal{A}$. Thus, the probability that there exists $i \in [\hat{\mathfrak{m}}_Q + 1,$

$N], j \in [1,t]$ that $\tilde{C}(\hat{x}_{i,j}) \neq MPRF.Eval(\hat{k}, \hat{x}_{i,j})$ is negligible. Moreover, as $(\hat{k}, \hat{k}')$ are uniform over $(\{0,1\}^\ell)^2$, $\Pr[\hat{k} = \hat{k}'] = negl(\lambda)$. So by the key injectivity of $MPRF$, the probability that there exists $i \in [\hat{\mathtt{m}}_Q + 1, N]$, $j \in [1,t]$ that $\tilde{C}(\hat{x}_{i,j}) = MPRF.Eval(\hat{k}', \hat{x}_{i,j})$ is negligible, which implies that in Game 6 we also have $\alpha_i = 0$ for $i \in [\hat{\mathtt{m}}_Q + 1, N]$ with all but negligible probability.

• *Game 8.* This is identical to Game 7 except that in Phase 3, the challenger use a random function $g \xleftarrow{\$} FUN_{n,m}$ instead of the pseudorandom function $MPRF.Eval(\hat{k}', \cdot)$, i.e. $\hat{\mathsf{S}}_h = \{(\hat{x}_{i,j}, g(\hat{x}_{i,j}))\}_{i\in[1,\hat{\mathtt{m}}_h], j\in[1,t]}$ for $h \in [1,Q]$.

As the secret key $\hat{k}'$ has not been appeared explicitly in the view of $\mathcal{A}$, indistinguishability between Game 7 and Game 8 comes from the pseudorandomness of $MPRF$ directly.

• *Game 9.* This is identical to Game 8 except that the challenger sets $\hat{\mathsf{S}}_h = \{(\hat{x}_{i,j}, g(\hat{x}_{i,j}))\}_{i\in[\hat{\mathtt{m}}_1+1,\hat{\mathtt{m}}_h], j\in[1,t]}$ for $h \in [1,Q]$. Note that here $\hat{\mathsf{S}}_h$ contains $\hat{\mathtt{m}}_1 \cdot t$ less elements compared to that in previous games.

Indistinguishability between Game 8 and Game 9 comes from the privacy of $MPRF$. More precisely, let P_8 and P_9 be the probabilities that Game 8 and Game 9 output 1 respectively, then if $|\mathsf{P}_8 - \mathsf{P}_9|$ is not negligible, there exists an adversary $\mathcal{B}$ that can break the privacy of $MPRF$, which works as follows:

1. In the beginning, the adversary $\mathcal{B}$ samples $\{x_{i,j}\}_{i\in[1,N], j\in[1,t]}$ and $\{z_i\}_{i\in[1,d]}$ uniformly at random from $\{0,1\}^n$ and aborts if there are repeated values among them. Obviously, with all but negligible probability, $\mathcal{B}$ will not abort.
2. Then $\mathcal{B}$ starts running $\mathcal{A}$ and answers mark oracle queries made by $\mathcal{A}$.
3. When the adversary makes a query $\mathtt{M} = (\mathtt{m}_1, \ldots, \mathtt{m}_Q)$ to the challenge oracle, $\mathcal{B}$ sets $\mathtt{X}_{h,0} = \{x_{i,j}\}_{i\in[1,\mathtt{m}_h], j\in[1,t]}$, and $\mathtt{X}_{h,1} = \{x_{i,j}\}_{i\in[\mathtt{m}_1+1,\mathtt{m}_h], j\in[1,t]}$, for $h \in [1,Q]$. Then it submits $(\mathtt{X}_{1,0}, \mathtt{X}_{1,1}, \ldots, \mathtt{X}_{Q,0}, \mathtt{X}_{Q,1})$ to its challenger. Next, it uses the returned constrained keys to generate the marked circuits and returns them back to $\mathcal{A}$.
4. Then $\mathcal{B}$ continues to answers mark oracle queries made by $\mathcal{A}$.
5. After the adversary $\mathcal{A}$ submits its challenge $\tilde{C}$, $\mathcal{B}$ first checks if Bad_2 and Bad_3 occurs. To help check these events, $\mathcal{B}$ needs to query each z_i for $i \in [1,d]$ to the evaluation oracle. Then, it computes α_i for $i \in [1,N]$. Note that here each α_i can be computed without referring to the random function g, which is implicitly set to be the random function used by the challenger of $\mathcal{B}$.
6. Finally, the adversary $\mathcal{B}$ finds the first i^* that $\alpha_{i^*} - \alpha_{i^*+1} \geq \frac{t}{N+1}$ and outputs 1 iff $i^* \notin \mathtt{M}$.

Here $\mathcal{B}$ is privacy-admissible. To see this, we first define $\mathtt{T}_1 = \{x_{i,j}\}_{i\in[1,\mathtt{m}_1], j\in[1,t]}$, $\mathtt{T}_2 = \{x_{i,j}\}_{i\in[\mathtt{m}_1+1,\mathtt{m}_Q], j\in[1,t]}$, and $\mathtt{T}_3 = \{0,1\}^n - (\mathtt{T}_1 \cup \mathtt{T}_2)$. Obviously, for any $i, j \in [1,3]$, $\mathtt{T}_i \cap \mathtt{T}_j = \emptyset$. Then we check the two conditions for privacy-admissible. First, for any $x \in \mathtt{T}_1$ and for any $i, j \in [1,Q]$, $d_{x,i,0} = d_{x,j,0} = 1$ and $d_{x,i,1} = d_{x,j,1} = 0$ (recall that $d_{x,a,b} = 1$ iff $x \in \mathtt{X}_{a,b}$ for $a \in [1,Q]$ and $b \in \{0,1\}$); for any $x \in \mathtt{T}_2$ and for any i, $j \in [1,Q]$, $d_{x,i,0} = d_{x,i,1}$ and $d_{x,j,0} = d_{x,j,1}$; for any $x \in \mathtt{T}_3$ and for any $i, j \in [1,Q]$, $d_{x,i,0} = d_{x,j,0} = d_{x,i,1} = d_{x,j,1} = 0$. Thus, the first condition is satisfied. Then, for each z_h for $h \in [1,d]$, $z_h \in \mathtt{T}_3$, thus $d_{z_h,i,0} = d_{z_h,i,1} = 0$ for $i \in [1,Q]$, i.e. the second condition is satisfied.

Also, it is easy to check that if the bit b sampled by the challenger of $\mathcal{B}$ is 0, the environment simulated by $\mathcal{B}$ is statically indistinguishable from the environment $\mathcal{A}$ faces in

Game 8; and otherwise, the environment simulated by $\mathcal{B}$ is statically indistinguishable from the environment $\mathcal{A}$ faces in Game 9. So, if $|\mathsf{P}_8 - \mathsf{P}_9|$ is not negligible, then the privacy-admissible adversary $\mathcal{B}$ can break the selective consistent privacy of $MPRF$.

• *Game 10.* This is identical to Game 9 except that the challenger further checks if Bad_4 occurs after checking the occurrence of Bad_3, where Bad_4 occurs iff there exists $i, i' \in [1, N]$, $j, j' \in [1, t]$ that $(i, j) \neq (i', j')$ but $\hat{x}_{i,j} = \hat{x}_{i',j'}$.

It is obvious that $\Pr[Bad_4] \leq (Nt)^2/2^n$ which is negligible. Thus, Game 9 and Game 10 are identical with all but negligible probability.

It remains to show that the probability that Game 10 outputs 1 is negligible. First, we argue that in Game 10, $\alpha_1 = \ldots = \alpha_{\hat{\mathtt{m}}_1} = t$ with all but negligible probability. To see this, note that by the unremoving-admissibility of $\mathcal{A}$, with all but negligible probability, there exists $h \in [1, Q]$ that $\tilde{C}(\cdot)$ differs from $\hat{C}_h(\cdot)$ on only a negligible fraction of inputs. Also, in Game 10, $\{\hat{x}_{i,j}\}_{i\in[1,\hat{\mathtt{m}}_1],j\in[1,t]}$ is sampled uniformly and is independent of the view of $\mathcal{A}$. Thus, with all but negligible probability, we have $\tilde{C}(\hat{x}_{i,j}) = \hat{C}_h(\hat{x}_{i,j}) = \hat{C}_Q(\hat{x}_{i,j})$ for all $i \in [1, \hat{\mathtt{m}}_1]$, $j \in [1, t]$. Therefore, with all but negligible probability, in Game 10, $\alpha_1 = \ldots = \alpha_{\hat{\mathtt{m}}_1} = t$.

Then we argue that in Game 10, for any $i \in [\hat{\mathtt{m}}_1, \hat{\mathtt{m}}_Q]$ that $i \notin \hat{\mathtt{M}}$, we have $\alpha_i - \alpha_{i+1} < \frac{t}{N+1}$ with all but negligible probability. First, fix any $i \in [\hat{\mathtt{m}}_1, \hat{\mathtt{m}}_Q] - \hat{\mathtt{M}}$, then it must lies between two queried messages, i.e. there exists $a \in [1, Q]$ that $\hat{\mathtt{m}}_a < i < \hat{\mathtt{m}}_{a+1}$. Let $\mathtt{X}_a = \{\hat{x}_{i,j}\}_{i\in[\hat{\mathtt{m}}_a+1,\hat{\mathtt{m}}_{a+1}]}$, and $s = \|\mathtt{X}_a\|$. Also, let $r = \|\{x \mid x \in \mathtt{X}_a \wedge \tilde{C}(x) = \hat{C}_Q(x)\}\|$. As the exact partion of $\mathtt{X}_a$ is independent of the view of $\mathcal{A}$, we need not divide $\mathtt{X}_a$ untill the extraction phase. Thus, the value of α_i and α_{i+1} are distributed according to the distribution $\mathcal{H}(r, s, t)$ respectively, where the probability is taken over the randomness used when dividing $\mathtt{X}_a$. Therefore, we have $\Pr[\alpha_i \geq (\frac{r}{s} + \frac{1}{2(N+1)})t] \leq e^{-\frac{t}{2(N+1)^2}}$ and $\Pr[\alpha_{i+1} \leq (\frac{r}{s} - \frac{1}{2(N+1)})t] \leq e^{-\frac{t}{2(N+1)^2}}$, both of which are negligible. By the union bound, we have with all but negligible probability, $\alpha_i - \alpha_{i+1} < \frac{t}{N+1}$. Finally, by the union bound, the probability that there exists $i \in [\hat{\mathtt{m}}_1, \hat{\mathtt{m}}_Q] - \hat{\mathtt{M}}$ that $\alpha_i - \alpha_{i+1} \geq \frac{t}{N+1}$ is negligible.

Now, with all but negligible probability, we have $\alpha_0 = \ldots = \alpha_{\hat{\mathtt{m}}_1} = t$, $\alpha_{\hat{\mathtt{m}}_Q+1} = \ldots = \alpha_{N+1} = 0$, and $\alpha_i - \alpha_{i+1} < \frac{t}{N+1}$ for $i \in [\hat{\mathtt{m}}_1, \hat{\mathtt{m}}_Q] - \hat{\mathtt{M}}$. So, with all but negligible probability, the position i that $\alpha_i - \alpha_{i+1} \geq \frac{t}{N+1}$ must be in $\hat{\mathtt{M}}$, which implies that Game 10 will output 1 with only a negligible probability.

That completes the proof of collusion resistant unremovability.

Proof of Unforgeability. To prove the δ-unforgeability of WM, we define the following games between a challenger and a PPT δ-unforging-admissible adversary $\mathcal{A}$.

• *Game 0.* This is the real experiment $ExptUF$. More precisely, the challenger proceeds as follows.

1. The challenger samples $K \xleftarrow{\$} \mathcal{K}$ and $(z_1, \ldots, z_d) \xleftarrow{\$} (\{0,1\}^n)^d$, it also maintains a table $\mathtt{T}$, which is initialized as an empty set.
2. Then, it answers marking oracle queries from $\mathcal{A}$. More precisely, on the hth query, on receiving $(k_h, \mathtt{m}_h) \in \{0,1\}^\ell \times [1, N]$, it first computes $w_h = (MPRF.Eval(k_h, z_1), \ldots, MPRF.Eval(k_h, z_d))$ and $v_h = (\{x_{h,i,j}\}_{i\in[1,N],j\in[1,t]}, k'_h) = F(K, w_h)$, generates the set $\mathsf{S}_h = \{(x_{h,i,j}, MPRF.Eval(k'_h, x_{h,i,j}))\}_{i\in[1,\mathtt{m}_h],j\in[1,t]}$ and the constrained key $ck_h \leftarrow MPRF.Constrain(k_h, \mathsf{S}_h)$, returns the circuit $C_h(\cdot) = MPRF.ConstrainEval(ck_h, \cdot)$,

and puts (k_h, v_h, C_h) into the table $\mathtt{T}$. Here, we define the number of marking oracle queries made by $\mathcal{A}$ as Q.

3. Finally, on input a circuit $\tilde{C}$ from $\mathcal{A}$, it computes $\check{w} = (\tilde{C}(z_1), \ldots, \tilde{C}(z_d))$ and $\check{v} = (\{\check{x}_{i,j}\}_{i\in[1,N],j\in[1,t]}, \check{k}') = F(K, \check{w})$, and for $i \in [1, N]$, it computes $\alpha_i = \|\{j \mid \tilde{C}(\check{x}_{i,j}) = MPRF.Eval(\check{k}', \check{x}_{i,j})\}\|$. It also sets $\alpha_0 = t$ and $\alpha_{N+1} = 0$ and finds the first i^* satisfying $\alpha_{i^*} - \alpha_{i^*+1} \geq \frac{t}{N+1}$. It outputs 1 iff $i^* \neq 0$.

• *Game 1.* This is identical to Game 0 except that the challenger uses a random function $f(\cdot) \xleftarrow{\$} FUN_{md,Ntn+\ell}$ instead of the pseudorandom function $F(K, \cdot)$ during the experiment. Indistinguishability of Game 0 and Game 1 comes from the pseudorandomness of F directly.

• *Game 2.* This is identical to Game 1 except that the way the challenger "computes" f. More precisely, when answering a mark oracle query $(k, \mathtt{m})$, the challenger first searches k in the table $\mathtt{T}$; it sets the result of $f(\cdot)$ to be v_i if there exists $(k_i, v_i, C_i) \in \mathtt{T}$ in the table that $k_i = k$; otherwise, it samples a fresh $v \xleftarrow{\$} \{0,1\}^{Ntn+\ell}$ and sets the result of $f(\cdot)$ to be v. Besides, at the beginning of Phase 3, for $i \in [1, Q]$, the challenger computes $w_i = (MPRF.Eval(k_i, z_1), \ldots, MPRF.Eval(k_i, z_d))$. It aborts and outputs "2" if there exists $i, j \in [1, Q]$ that $k_i \neq k_j$ and $w_i = w_j$ (here, We call this event as "Bad_1"). Finally, when computing the function $f(\check{w})$ in Phase 5, the challenger sets $f(\check{w})$ to be v_i if there exists $i \in [1, Q]$ that $w_i = \check{w}$; otherwise, it samples $v \xleftarrow{\$} \{0,1\}^{Ntn+\ell}$ and sets the result to be v.

Notice that Game 1 and Game 2 are identical if the event Bad_1 does not occur in Game 2. By the key injectivity of the $MPRF$, $\Pr[Bad_1] = 0$ in Game 2, thus the views of $\mathcal{A}$ in Game 1 and Game 2 are identically distributed.

• *Game 3.* This is identical to Game 2 except that after checking whether Bad_1 occurs, the challenger further checks if Bad_2 occurs, where Bad_2 occurs iff there exists $i \in [1, Q]$ that $w_i = \check{w}$. The experiment aborts and outputs 3 if Bad_2 occurs and proceeds identically to Game 2 otherwise.

Game 2 and Game 3 are identical as long as Bad_2 does not occur. Note that in Game 2 and in Game 3, each z_j is sampled uniformly and is independent of the view of $\mathcal{A}$. Also, by the δ-unforging-admissibility of $\mathcal{A}$ and the fact that for all $i \in [1, Q]$, $C_i(\cdot)$ and $MPRF.Eval(k_i, \cdot)$ differ on only polynomial inputs, with all but negligible probability, for all $i \in [1, Q]$, $\tilde{C}(\cdot)$ differs from $MPRF.Eval(k_i, \cdot)$ on at least $(\delta - negl(\lambda)) \cdot 2^n$ inputs. Thus, for any $i \in [1, Q]$, $\Pr[w_i = \check{w}] = \Pr[\forall j \in [1, d] : \tilde{C}(z_j) = MPRF.Eval(k_i, z_j)] \leq (1 - (\delta - negl(\lambda)))^d + negl(\lambda) \leq e^{-\lambda/2} + negl(\lambda)$, which is negligible. Therefore, by the union bound, the probability that there exists w_i that $\check{w} = w_i$ is also negligible.

• *Game 4.* This is identical to Game 3 except that in the extraction algorithm, the challenger use a random function $g(\cdot) \xleftarrow{\$} FUN_{n,m}$ instead of $MPRF.Eval(\check{k}', \cdot)$.

Indistinguishability between Game 3 and Game 4 comes from the pseudorandomness of $MPRF$ directly since $\check{k}'$ is sampled uniformly and is independent of the view of $\mathcal{A}$. Besides, in Game 4, as $\alpha_1 = \|\{j \mid \tilde{C}(\check{x}_{1,j}) = g(\check{x}_{1,j})\}\|$, the probability that $\alpha_1 \neq 0$ is negligible. Therefore, with only a negligible probability, the experiment will output 1 in Game 4. That completes the proof of δ-unforgeability.

□

6 Collusion Resistant Watermarking Schemes for Other Cryptographic Functionalities

In this section, we demonstrate the usefulness of our collusion resistant watermarkable PRF via building more advanced collusion resistant watermarkable cryptographic primitives from it. More precisely, the constructed advanced watermarkable cryptographic primitives include the collusion resistant watermarkable SKE scheme, the collusion resistant watermarkable MAC scheme, the collusion resistant watermarkable PKE scheme, and the collusion resistant watermarkable signature scheme. The constructions are all based on the observation that there exists a construction of the PKE scheme (signature scheme, SKE scheme, or MAC scheme) in [SW14] (resp. [SW14], [GGM84b], or [GGM84b]) whose decryption algorithm (resp. sign algorithm, decryption algorithm, or mac algorithm) is nothing more than a (puncturable) PRF evaluation, which was initially presented in [NW15, CHN+16] and was used to construct the watermarkable PKE scheme and the watermarkable signature scheme therein. Here, as an example, we only give a detailed description for how to construct watermarkable PKE schemes, and the other three primitives, namely, the watermarkable signature scheme, the watermarkable SKE scheme and the watermarkable MAC scheme can be constructed and proved in a similar way.

Overview. Roughly speaking, the watermarking scheme for the decryption algorithm of a PKE scheme constructed from the watermarkable puncturable PRF works as follows. First, note that the secret key of the PKE scheme is just the secret key of the underlying puncturable PRF. Then, to embed a message into the secret key, the mark algorithm runs the mark algorithm for the underlying puncturable PRF to generate a circuit C, and outputs a circuit C' that $C'(x_1, x_2) = C(x_1) \oplus x_2$. To extract the message, the extraction algorithm first generates a circuit C', that $C'(x) = C(x, y) \oplus y$ for a properly chosen y[11], then it inputs the circuit C' into the extraction algorithm for the underlying (puncturable) PRF and outputs the result. Security, including the correctness, the collusion resistant unremovability and the unforgeability, of the watermarking scheme for the decryption algorithm of the PKE scheme can be reduced to the security of the watermarking scheme for the underlying (puncturable) PRF.

However, there is a subtle issue in the proof of the collusion resistant unremovability. More precisely, to better describe the ability of the real-world attackers, in the definition of the collusion resistant unremovability, the adversary should be additionally given the public key of the challenge secret key. However, in the PKE scheme constructed in [SW14], the public key is just an obfuscated program containing the secret key, which may reveal additional information about the secret key and helps the adversary to remove or modify the marked messages. In [NW15, CHN+16], this problem is solved by setting the public key to be an obfuscated program containing the *marked* secret key, but this will additionally involve the marking secret key in the key generation algorithm of the PKE scheme, which prevents the users from generating public key/secret key pairs by themselves. Here, we give another approach to solve this problem, which preserves the "decentralized key generation" property. To complete

[11] We will describe how to choose y below.

this approach, we additionally require that the underlying puncturable PRF has a new property "dispersibility", which is formally defined in Appendix B. Roughly, it requires that the set $\mathcal{D}$ of inputs that are evaluated differently under the marked secret key and the original secret key are not likely to intersect with a pre-defined small set. To see how dispersibility helps, recall that in the public key of a PKE scheme constructed in [SW14], which is an obfuscated program, the underlying puncturable PRF will only evaluate on strings in the range of a PRG $\mathtt{G}$, which is negligible compared to the input space of the puncturable PRF. By the dispersibility, $\mathcal{D}$ and the range of $\mathtt{G}$ are disjoint with high probability. So, with high probability, the program contained in the public key works identically after the secret key of the puncturable PRF has been replaced with a marked one in the program, and by the indistinguishability of the indistinguishability obfuscator, the two public keys are computationally indistinguishable, which implies that the public key that contains the original secret key will not help the adversary to break the collusion resistant unremovability.

The Construction. Next, we give a detailed construction of the collusion resistant watermarkable PKE scheme, whose security definition is formally given in Appendix C.1.

Let λ be the security parameter and δ be a positive real value that $\lambda/\delta = poly(\lambda)$. Let $PPRF = (PPRF.KeyGen, PPRF.Eval, PPRF.Puncture, PPRF.PunctureEval)$ be a puncturable PRF with key space $\{0,1\}^\ell$, input space $\{0,1\}^n$, and output space $\{0,1\}^m$. Let $PKE = (PKE.KeyGen, PKE.Enc, PKE.Dec)$ be a PKE scheme that is constructed from $PPRF$ under the framework in [SW14]. In particular, the secret key space of PKE is $\{0,1\}^\ell$, and the decryption algorithm of PKE is defined as $PKE.Dec(k, c_1, c_2) = PPRF.Eval(k, c_1) \oplus c_2$. Let $\mathtt{F}$ be a quasi-polynomial secure PRF, i.e. $\mathtt{F}$ remains pseudorandomness for any probabilistic quasi-polynomial time adversary (this property is easy to obtain as we have no additional requirement on $\mathtt{F}$), with key space $\{0,1\}^\kappa$, input space $\{0,1\}^n$ and output space $\{0,1\}^m$. Let $WM' = (WM'.Setup, WM'.Mark, WM'.Extract)$ be a watermarking scheme with message space $\mathcal{M}$ for the evaluation algorithm of $PPRF$. Also, let $G : \{0,1\}^{n/2} \to \{0,1\}^n$ and iO be the PRG and the indistinguishability obfuscator used in the construction of PKE respectively. Then the watermarking scheme WM for the decryption algorithm of PKE works as follows.

- **Setup.** On input a security parameter λ, the setup algorithm samples $K \xleftarrow{\$} \{0,1\}^\kappa$, runs $msk' \leftarrow WM'.Setup(1^\lambda)$, and outputs $msk = (msk', K)$.
- **Mark.** On input a watermarking secret key $msk = (msk', K)$, a secret key $k \in \{0,1\}^\ell$ for PKE and a message $\mathtt{m} \in \mathcal{M}$, the mark algorithm first computes $C' \leftarrow WM'.Mark(msk', k, \mathtt{m})$ and outputs the circuit C that

$$C(x_1, x_2) = C'(x_1) \oplus x_2.$$

- **Extract.** On input a watermarking secret key $msk = (msk', K)$, and a circuit C, the extraction algorithm generates C' that

$$C'(x) = C(x, \mathtt{F}_K(x)) \oplus \mathtt{F}_K(x)$$

, and returns $WM'.Extract(msk', C')$.

Theorem 6.1. *If WM' is a secure watermarking scheme with collusion resistant unremovability, δ-unforgeability and dispersibility for the puncturable PRF PPRF and* F *is a quasi-polynomial secure PRF, then WM is a secure watermarking scheme with collusion resistant unremovability and $4\sqrt{\delta}$-unforgeability for the decryption algorithm of the PKE scheme PKE.*

We omit the proof here and put it into Appendix C.2.

References

[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*. Springer-Verlag Berlin Heidelberg, 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. Cryptology ePrint Archive, Report 2017/557, 2017. http://eprint.iacr.org/2017/557.

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

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 501–510. ACM, 2008.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. *EUROCRYPT*, pages 719–737, 2012.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, volume 4004, pages 573–592. Springer, 2006.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from lwe. Cryptology ePrint Archive, Report 2017/795, 2017. http://eprint.iacr.org/2017/795.

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

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO*, pages 480–499. Springer, 2014.

[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.

[Chv79] Vasek Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285–287, 1979.

[CHV15] Aloni Cohen, Justin Holmgren, and Vinod Vaikuntanathan. Publicly verifiable software watermarking. *IACR Cryptology ePrint Archive*, 2015:373, 2015.

[CMB+07] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan Kaufmann, 2007.

[FOR17] Pooya Farshim, Claudio Orlandi, and Razvan Rosie. Security of symmetric primitives under incorrect usage of keys. *IACR Transactions on Symmetric Cryptology*, 2017(1):449–473, 2017.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49. IEEE, 2013.

[GGM84a] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct randolli functions. In *FOCS*, pages 464–479. IEEE, 1984.

[GGM84b] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *CRYPTO*, pages 276–288. Springer, 1984.

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

[HMW07] Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. *TCC*, pages 362–382, 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*, 2017.

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

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.

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

[NW15] Ryo Nishimaki and Daniel Wichs. Watermarking cryptographic programs against arbitrary removal strategies. *IACR Cryptology ePrint Archive*, 2015:344, 2015.

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

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

[YAL+17] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. Lattice-based techniques for accountable anonymity: Composition of abstract sterns protocols and weak prf with efficient protocols from lwr. Cryptology ePrint Archive, Report 2017/781, 2017. http://eprint.iacr.org/2017/781.

[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.

A Puncturable Pseudorandom Function with Key Injectivity

In this section, we give a construction of the puncturable pseudorandom function with key injectivity, which is defined in Definition 2.2, from the LWE assumption. Our construction is similar to the puncturable PRF with *weak* key injectivity from the LWE assumption constructed in [CHN+16], but we refine the parameters to achieve the key injectivity.[12] For completeness, we describe the detailed construction here.

More precisely, our construction works in two steps. First, we prove that the GGM-framework preserves the key injectivity.

Definition A.1 (Pseudorandom Generator with left-right-injectivity.). *Let* $PRG : \{0,1\}^k \to \{0,1\}^{2l}$ *be a pseudorandom generator. Let* $PRG_0(\cdot) = PRG(\cdot)[1 : l]$ *and* $PRG_1(\cdot) = PRG(\cdot)[l+1, 2l]$. *Then PRG is left-right-injective if both* PRG_0 *and* PRG_1 *are injective, namely, for any* $k_1, k_2 \in \{0,1\}^k$ *that* $k_1 \neq k_2$, *we have* $PRG_0(k_1) \neq PRG_0(k_2)$ *and* $PRG_1(k_1) \neq PRG_1(k_2)$.

Lemma A.1. *Let* $PRG^{(1)}, \ldots, PRG^{(t)}$ *be t pseudorandom generators that for* $i \in [1,t]$, $PRG^{(i)} : \{0,1\}^{l_{i-1}} \to \{0,1\}^{2l_i}$ *is left-right-injective and for* $i \in [0,t]$, l_i *is polynomial in the security parameter* λ. *Then the GGM tree-based construction of PRF that uses* $PRG^{(i)}$ *at the ith level, namely,* $PRF_K(x) = PRG^{(t)}_{x[t]}(PRG^{(t-1)}_{x[t-1]}(\ldots PRG^{(1)}_{x_1}(K)))$, *is a secure puncturable PRF with key injectivity with key space* $\{0,1\}^{l_0}$, *input space* $\{0,1\}^t$ *and output space* $\{0,1\}^{l_t}$.

Proof. As has been proved in previous works [GGM84a, BW13, KPTZ13, BGI14, SW14], the constructed PRF function F is a puncturable PRF. It remains to show that the puncturable PRF F is key injective. Now, for any $K_1, K_2 \in \{0,1\}^{l_0}$ that $K_1 \neq K_2$ and for any $x \in \{0,1\}^t$, we set $K_{1,0} = K_1$, $K_{2,0} = K_2$ and $K_{i,j} = PRG^j_{x[j]}(K_{i,j-1})$ for $i \in \{1,2\}$ and $j \in [1,t]$. Note that if $K_{1,j-1} \neq K_{2,j-1}$, by the left-right-injectivity of the underlying PRGs, we have $K_{1,j} \neq K_{2,j}$. Also, as $K_1 \neq K_2$, we can conclude that $K_{1,t} \neq K_{2,t}$. Finally, by the definition of F, we have $F_{K_1}(x) = K_{1,t}$ and $F_{K_2}(x) = K_{2,t}$, which implies that $F_{K_1}(x) \neq F_{K_2}(x)$. That completes the proof. □

Next, we construct a series of pseudorandom generators with "left-right-injectivity" from the LWE assumption, which satisfy the requirement in Lemma A.1.

[12] Recall that the difference between our "key injectivity" and their "weka key injectivity" is somewhat similar to the difference between the "collision resistance" and the "target-collision resistance".

Lemma A.2. *Let λ be the security parameter, let $t = O(\lambda)$, let $q_i = 2^{(t+1-i)\lambda}$ for $i \in [0, t]$, let $n_0 = poly(\lambda)$ and $n_i = n_{i-1} \cdot ((t+2-i)\lambda + 1)/((t+1-i)\lambda - 1)$ for $i \in [1, t]$, and let $L_i = n_i \cdot (t+1-i)\lambda$. Also, let $\boldsymbol{A}_{i,0}, \boldsymbol{A}_{i,1} \xleftarrow{\$} \mathbb{Z}_{q_{i-1}}^{n_i \times n_{i-1}}$ for $i \in [1, t]$. Then $PRG = (PRG^{(1)}, \ldots, PRG^{(t)})$ is a series of PRG with left-right-injectivity that satisfies the requirement in Lemma A.1, where for $i \in [1, t]$, $PRG^{(i)} : \mathbb{Z}_{q_{i-1}}^{n_{i-1}} \to \mathbb{Z}_{q_i}^{2n_i}$ (or alternatively, from $\{0,1\}^{L_{i-1}}$ to $\{0,1\}^{2L_i}$) is defined as follows:*

$$PRG_0^{(i)}(\boldsymbol{x}) = \lfloor q_i/q_{i-1} \cdot \boldsymbol{A}_{i,0} \cdot \boldsymbol{x} \rceil \mod q_i$$
$$PRG_1^{(i)}(\boldsymbol{x}) = \lfloor q_i/q_{i-1} \cdot \boldsymbol{A}_{i,1} \cdot \boldsymbol{x} \rceil \mod q_i$$

Proof. Pseudorandomness of these PRGs follows directly from the learning with rounding assumption, which is first introduced in [BPR12] and is as hard as the LWE assumption if the parameter is properly set. Also, as has been proved in [YAL+17], the map from $\boldsymbol{x}$ to $\lfloor q_i/q_{i-1} \cdot \boldsymbol{A}_{i,b} \cdot \boldsymbol{x} \rceil \mod q_i$ is injective with all but negligible probability over the choice of $\boldsymbol{A}_{i,b}$, for $i \in [1, t]$ and $b \in \{0, 1\}$. It remains to show that L_i is polynomial in λ for all $i \in [0, t]$. First, $L_0 = n_0 \cdot (t+1)\lambda = poly(\lambda) \cdot O(\lambda) \cdot \lambda$, which is polynomial in λ. Then for $i \in [1, t]$, let $j = t + 1 - i$, and we have

$$\begin{aligned} L_i/L_{i-1} &= (n_i \cdot (t+1-i) \cdot \lambda)/(n_{i-1} \cdot (t+2-i) \cdot \lambda) \\ &= (((j+1)\lambda + 1) \cdot j)/((j\lambda - 1) \cdot (j+1)) \\ &= 1 + (2j+1)/((j+1)(j\lambda - 1)) \\ &\leq 1 + 2/(j\lambda - 1) \end{aligned}$$

This implies that for $i \in [1, t]$,

$$\begin{aligned} L_i &\leq L_0 \cdot \prod_{j=1}^{t} (1 + 2/(j\lambda - 1)) \\ &\leq L_0 \cdot (1 + 2/(\lambda - 1))^t \\ &\leq L_0 \cdot e^{2t/(\lambda-1)} \\ &= L_0 \cdot e^c \end{aligned}$$

for some constant c, i.e. L_i is also polynomial in the security parameter λ. That completes the proof. □

Note that the PRGs (thus, the PRF) are not always left-right-injective ones (resp. a key injective one), and it is only guaranteed that when randomly sampling the PRG functions (resp. the PRF family), they fail to be left-right-injective ones (resp. a key injective one) with only a negligible probability. In this paper, for the simplicity of description, we omit the negligible fail probability, and assume that a family of puncturable PRF with key injectivity is always used.

B On the Dispersibility of Watermarkable PRF

In this section, we give a formal definition of the "dispersibility" property and show that if the underlying normal PRF $\mathtt{F}$ satisfies an additional property, the watermarking

scheme constructed in Sec. 5 has dispersibility. Formally, the "dispersibility" is defined as follows.

Definition B.1 (Dispersibility.). *Let $WM = (WM.Setup, WM.Mark, WM.Extract)$ be a watermarking scheme with message space $\mathcal{M}$ for the evaluation algorithm of a (puncturable) PRF with key space $\{0,1\}^\ell$, input space $\{0,1\}^n$, and output space $\{0,1\}^m$. Let λ be the security parameter. Then WM has dispersibility if for any $\mathtt{S}$ that $\frac{\|S\|}{2^n} = negl(\lambda)$ and any $k \in \mathcal{K}$, we have*

$$\Pr[\exists \mathtt{m} \in \mathcal{M}, \{x \in \{0,1\}^n \mid C'(x) \neq \mathtt{F}_k(x)\} \cap \mathtt{S} \neq \emptyset] \leq negl(\lambda)$$

where $msk \leftarrow WM.Setup(1^\lambda)$, $C' \leftarrow WM.Mark(msk, k, \mathtt{m})$, and the probability is taken over the sample of the marking key msk.

Lemma B.1. *Let WM be the watermarking scheme for a PRF function constructed in Sec. 5. Let $\mathtt{F}$ be the underlying normal PRF (not the watermarked PRF). If $F : \mathcal{K} \times \{0,1\}^{md} \to \{0,1\}^{Ntn+\ell}$ satisfies that for any input $w \in \{0,1\}^{md}$, and any $v \in \{0,1\}^{Ntn+\ell}$, $\Pr[K \xleftarrow{\$} \mathcal{K} : \mathtt{F}(K, w) = u] = 1/2^{Ntn+\ell}$, then WM has dispersibility.*

Proof. First, recall that in WM, fixing the secret key k and the marking secret key $msk = (K, z_1, \ldots, z_d)$, then for any $\mathtt{m} \in \mathcal{M}$, $\{x \in \{0,1\}^n \mid C'(x) \neq \mathtt{F}_k(x)\} \subseteq \mathcal{X}$, where $C' \leftarrow WM.Mark(msk, k, \mathtt{m})$, $w = (MPRF.Eval(k, z_1), \ldots, MPRF.Eval(k, z_d))$, $(\{x_i\}_{i\in[1,Nt]}, k') = F(K, w)$ and $\mathcal{X} = \{x_i\}_{i\in[1,Nt]}$. So it is sufficient to show that for any $\mathtt{S}$ and any k, we have

$$\Pr[\mathcal{X} \cap \mathtt{S} \neq \emptyset] \leq negl(\lambda) \tag{1}$$

where the probability is taken over the sample of msk. Next, we fix z_1 to z_d, then it is sufficient to prove that for any $\mathtt{S}$, any k, and any $(z_1, \ldots, z_d)$, Equation (1) holds, where the probability is taken over the sample of K. Note that now, by the additional property for $\mathtt{F}$, $\mathcal{X}$ is uniform in $(\{0,1\}^n)^{Nt}$. So, for each $i \in [1, Nt]$, $\Pr[x_i \in \mathtt{S}] \leq negl(\lambda)$. Then, by the union bound, $\Pr[\mathcal{X} \cap \mathtt{S} \neq \emptyset] \leq negl(\lambda)$. That completes the proof. □

We remark that the additional property for the normal PRF $\mathtt{F}$ required in Lemma B.1 is easy to obtain, and there are many candidate constructions, e.g. the well-known NaorReingold PRF [NR04].

C Omitted Details for The Construction of Collusion Resistant Watermarkable PKE Schemes

C.1 The Definition

The collusion resistant watermarkable PKE scheme can be defined similarly to the definition of the collusion resistant watermarkable PRF, with the main difference that in the challenge oracle, the adversary is further given the public key of the challenge secret key.

Definition C.1 (Watermarkable Family of PKE). *Let $PKE = (PKE.KeyGen, PKE.Enc, PKE.Dec)$ be a PKE scheme with secret key space $\mathcal{SK}$. The watermarking scheme with message space $\mathcal{M}$ for PKE (more accurately, the decryption algorithm of PKE) consists of three algorithms:*

- ***Setup.*** *On input the security parameter λ, the setup algorithm outputs the watermarking secret key msk.*
- ***Mark.*** *On input the watermarking secret key msk, a secret key $sk \in \mathcal{SK}$ of PKE, and a message $\mathtt{m} \in \mathcal{M}$, the mark algorithm outputs a marked circuit C.*
- ***Extract.*** *On input the master secret key msk and a circuits C, the extraction algorithm outputs a string $\mathtt{m} \in \mathcal{M} \cup \{\perp\}$.*

Definition C.2 (Watermarking Correctness). *Correctness of the watermarking scheme requires that for any $sk \in \mathcal{SK}$ and $\mathtt{m} \in \mathcal{M}$, let $msk \leftarrow Setup(1^\lambda)$, $C \leftarrow Mark(msk, sk, \mathtt{m})$, we have:*

- ***Functionality Preserving.*** *$C(\cdot) \sim_f PKE.Dec(sk, \cdot)$ where $1/f(n)$ is negligible in the security parameter.*
- ***Extraction Correctness.*** *$\Pr[Extract(msk, C) \neq \mathtt{m}] = negl(\lambda)$.*

Before defining the security of the collusion resistant watermarkable PKE, we first define oracles the adversaries can query during the security experiments. Recall that in the challenge oracle, the adversary is further given the challenge public key.

- **Marking Oracle $O^M_{msk}(\cdot,\cdot)$.** On input a message $\mathtt{m} \in \mathcal{M}$ and a secret key $sk \in \mathcal{SK}$, the oracle returns the circuit $C \leftarrow Mark(msk, sk, \mathtt{m})$.
- **Challenge Oracle $O^C_{msk}(\cdot)$.** On input a set $\mathtt{M}$ of messages, the oracle first generates a key pair $(sk, pk) \leftarrow PKE.KeyGen(1^\lambda)$. Then, for each $\mathtt{m}_i \in \mathtt{M}$, it computes $C_i \leftarrow Mark(msk, sk, \mathtt{m}_i)$. Finally, it returns the set $\mathtt{C} = \{C_i\}_{i \in [1, \|\mathtt{M}\|]}$ and the public key pk.

Definition C.3 (Collusion Resistant Unremovability). *The watermarking scheme for a PKE scheme is collusion resistant unremovable if for all PPT and unremoving-admissible adversaries $\mathcal{A}$, $\Pr[ExptUR_{\mathcal{A}}(\lambda) = 1] = negl(\lambda)$, where we define the experiment $ExptUR$ as follows:*

1. *The challenger samples $msk \leftarrow Setup(1^\lambda)$.*
2. *The adversary $\mathcal{A}$ is allowed to access the marking oracle, and can query it multiple times.*
3. *The adversary $\mathcal{A}$ makes a query $\mathtt{M}$ to the challenge oracle and gets a set $\mathtt{C}$ of circuits and a public key pk back.*
4. *The adversary $\mathcal{A}$ is further allowed to access the marking oracle, and can query it multiple times.*
5. *Finally the adversary submits a circuit $\tilde{C}$, and the experiment outputs 1 iff $Extract(msk, \tilde{C}) \notin \mathtt{M}$.*

Here, an adversary $\mathcal{A}$ is unremoving-admissible if with all but negligible probability, its submitted circuit $\tilde{C}$ satisfies that there exists circuit $C \in \mathtt{C}$ that $\tilde{C} \sim_f C$ for negligible $1/f(n)$.

Definition C.4 (δ-Unforgeability). *The watermarking scheme for a PKE scheme is δ-unforgeable if for all PPT and δ-unforging-admissible adversaries $\mathcal{A}$, $\Pr[ExptUF_{\mathcal{A}}(\lambda) = 1] = negl(\lambda)$, where we define the experiment $ExptUR$ as follows:*

1. *The challenger samples $msk \leftarrow Setup(1^\lambda)$.*
2. *The adversary $\mathcal{A}$ is allowed to access the marking oracle, and can query it multiple times.*
3. *Finally the adversary submits a circuit $\tilde{C}$, and the experiment outputs 1 iff $Extract(msk, \tilde{C}) \neq \perp$.*

Here, an adversary $\mathcal{A}$ is δ-unforging-admissible if with all but negligible probability, its submitted circuit $\tilde{C}$ satisfies that $\tilde{C} \not\sim_f C_i$ for all $i \in [1, Q]$, where Q is the number of queires $\mathcal{A}$ made to the marking oracle, C_i is the output of the marking oracle on the ith query, and $1/f(n) > \delta$.

C.2 The Proof for Theorem 6.1

Proof. The functionality-preserving and the extraction correctness of WM follows directly from the functionality-preserving and the extraction correctness of WM' respectively.

$4\sqrt{\delta}$-Unforgeability. The $4\sqrt{\delta}$-unforgeability of WM can be reduced to the δ-unforgeability of WM'. But before describing the reduction, we first prove an auxiliary lemma.

Lemma C.1. *For any PPT $4\sqrt{\delta}$-unforging-admissible adversary $\mathcal{A}$, we have*

$$\Pr\left[\begin{array}{ll} msk' \leftarrow WM'.KeyGen(1^\lambda); & \\ K \xleftarrow{\$} \{0,1\}^\kappa; & \\ \tilde{C} \leftarrow \mathcal{A}^{O_{msk'}(\cdot,\cdot)}(1^\lambda); & : \exists C' \in \mathcal{C}', C' \sim_{1/\delta} \tilde{C}' \\ \tilde{C}' \text{ is defined as :} & \\ \tilde{C}'(x) = \tilde{C}(x, F_K(x)) \oplus F_K(x) & \end{array}\right] \leq negl(\lambda)$$

Here, the oracle $O_{msk'}$ initializes two empty sets $\mathcal{C}'$ and $\mathcal{C}$ in the beginning; and on input a query $(k, \mathtt{m})$ from $\mathcal{A}$, it computes $C' \leftarrow WM'.Mark(msk', k, m)$, puts C' into $\mathcal{C}'$, generates a circuit C that $C(x_1, x_2) = C'(x_1) \oplus x_2$, puts C into $\mathcal{C}$ and outputs C. We also call two circuits $C \in \mathcal{C}$ and $C' \in \mathcal{C}'$ "paired circuits" if they are generated in the same oracle query.

Proof. To prove Lemma C.1, first, we consider a mental game that $F_K(\cdot)$ is replaced with a random function $f \xleftarrow{\$} FUN_{n,m}$, namely, considering the (exponential-size) circuit $\check{C}' = \tilde{C}(x, f(x)) \oplus f(x)$ instead of $\tilde{C}'$. Let $C' \in \mathcal{C}'$, $C \in \mathcal{C}$ be two paired circuits. Note that for any $x \in \{0,1\}^n$, $\check{C}'(x) = C'(x)$ iff $\tilde{C}(x, f(x)) = C(x, f(x))$. Then, let

$$\varepsilon = \frac{\|\{(x,y) \in \{0,1\}^n \times \{0,1\}^m \mid \tilde{C}(x,y) \neq C(x,y)\}\|}{2^{n+m}}.$$

Also, for $x \in \{0,1\}^n$, let

$$\varepsilon_x = \frac{\|\{y \in \{0,1\}^m \mid \tilde{C}(x,y) \neq C(x,y)\}\|}{2^m}.$$

Obviously, we have $\sum_{x\in\{0,1\}^n} \varepsilon_x = 2^n \cdot \varepsilon$. Next, we define the set $\mathcal{S} = \{x \mid \varepsilon_x \geq \varepsilon/2\}$. Thus, we have

$$\begin{aligned} 2^n \cdot \varepsilon &= \sum_{x\in\{0,1\}^n} \varepsilon_x \\ &\leq (2^n - \|\mathcal{S}\|) \cdot \frac{\varepsilon}{2} + \|\mathcal{S}\| \\ &\leq 2^n \cdot \frac{\varepsilon}{2} + \|\mathcal{S}\| \end{aligned}$$

which implies that $\|\mathcal{S}\| \geq 2^n \cdot \frac{\varepsilon}{2}$. Now, let $\mathcal{D}$ be the random variable indicating the number of inputs that are evaluated differently by $\check{C}'$ and C', where the probability is taken over the choice of f. Also, for $x \in \{0,1\}^n$, let $\mathcal{D}_x$ be the random variable indicating whether $\check{C}'(x) \neq C'(x)$ (1 for not equal), where the probability is taken over the choice of f, and let $\mathcal{E}_x$ be the random variable that being 1 with probability $\varepsilon/2$. Then, we have

$$\begin{aligned} \Pr[\mathcal{D} \leq 2^{n-3}\varepsilon^2] &= \Pr[\sum_{x\in\{0,1\}^n} \mathcal{D}_x \leq 2^{n-3}\varepsilon^2] \\ &\leq \Pr[\sum_{x\in\mathcal{S}} \mathcal{D}_x \leq 2^{n-3}\varepsilon^2] \\ &\leq \Pr[\sum_{x\in\mathcal{S}} \mathcal{E}_x \leq 2^{n-3}\varepsilon^2] \\ &\leq \Pr[\sum_{x\in[1,2^n\cdot\frac{\varepsilon}{2}]} \mathcal{E}_x \leq 2^{n-3}\varepsilon^2] \\ &\leq e^{-2^{n-5}\varepsilon^2} \end{aligned}$$

where the last inequality follows the chernoff bound as $\mathbb{E}(\sum_{x\in[1,2^n\cdot\frac{\varepsilon}{2}]} \mathcal{E}_x) = 2^{n-2} \cdot \varepsilon^2$. As $\mathcal{A}$ is unforging-admissible, $\varepsilon \geq 4\sqrt{\delta}$ with all but negligible probability, and if $\varepsilon \geq 4\sqrt{\delta}$, we have $2^{n-3}\varepsilon^2 \geq 2^{n+1}\delta$, which implies that

$$\begin{aligned} \Pr[\mathcal{D} \leq 2^{n+1}\delta] &\leq \Pr[\mathcal{D} \leq 2^{n-3}\varepsilon^2] \\ &\leq e^{-2^{n-5}\varepsilon^2} \\ &\leq e^{-2^{n-1}\delta} \end{aligned}$$

which is negligible. Therefore, with all but negligible probability, $\mathcal{D} \geq 2^{n+1}\delta$. The above analysis works for any $C' \in \mathcal{C}'$. So, by the union bound, we have with all but negligible probability, $\check{C}' \nprec_{1/(2\delta)} C'$ for all $C' \in \mathcal{C}'$.

Next, we turn back to the circuit $\tilde{C}'$ and show that with all but negligible probability, $C' \nprec_{1/\delta} \tilde{C}'$ for all $C' \in \mathcal{C}'$. Assume that in contrast with a non-negligible probability,

there exists $C' \in \mathcal{C}'$ that $C' \sim_{1/\delta} \tilde{C}'$, then we can construct an adversary $\mathcal{B}'$ that breaks the pseudorandomness of $\mathtt{F}$.

In more detail, the adversary $\mathcal{B}'$ generates $msk' \leftarrow WM'.Setup(1^\lambda)$ and simulates the oracle $\mathcal{O}_{msk'}(\cdot,\cdot)$ for $\mathcal{A}$ with msk' (The sets $\mathcal{C}$ and $\mathcal{C}'$ are also defined as in the body of Lemma C.1). Then, after $\mathcal{A}$ outputs a circuit $\tilde{C}$, $\mathcal{B}'$ runs the following procedure for each $C' \in \mathcal{C}'$.

1. $\mathcal{B}'$ samples $d = \lambda/\delta$ random inputs $\{x_i\}_{i\in[1,d]} \xleftarrow{\$} (\{0,1\}^n)^d$, queries each x_i to its oracle and obtains the set $\{y_i\}_{i\in[1,d]}$.
2. It computes $\beta_{C'} = \|\{i \in [1,d] \mid \tilde{C}(x_i, y_i) \oplus y_i \neq C'(x_i)\}\|$.
3. It outputs 1 and aborts if $\beta_{C'} < \frac{3}{2}\lambda$.

Finally, if $\mathcal{B}'$ completes the repetition without aborting, it outputs 0.

Note that if $\mathcal{B}'$ is answered with a random function f, it can perfectly simulate the view of $\mathcal{A}$ in the mental game, and $\beta_{C'} = \|\{i \in [1,d] \mid \check{C}'(x_i) \neq C'(x_i)\}\|$ for each $C' \in \mathcal{C}'$. Since with all but negligible probability, $\check{C}' \nsim_{1/(2\delta)} C'$ for all $C' \in \mathcal{C}'$, $\Pr[\beta_{C'} \leq \frac{3}{2}\lambda] \leq e^{-\lambda/16}$ for each $C' \in \mathcal{C}'$ by the chernoff bound. Again, by the union bound, $\mathcal{B}'$ outputs 1 with only a negligible probability in this case. In contrast, if $\mathcal{B}'$ is answered by the pseudorandom function with a key K, it can perfectly simulate the view of $\mathcal{A}$ in the game defined in the body of Lemma C.1, and $\beta_{C'} = \|\{i \in [1,d] \mid \tilde{C}'(x_i) \neq C'(x_i)\}\|$ for each $C' \in \mathcal{C}'$. Since we assume that with a non-negligible probability, there exists $C'^* \in \mathcal{C}'$ that $C'^* \sim_{1/\delta} \tilde{C}'$, we have $\Pr[\beta_{C'^*} \geq \frac{3}{2}\lambda] \leq e^{-\lambda/10}$ by the chernoff bound. So, $\mathcal{B}'$ will outputs 1 with a non-negligible probability in this case. Therefore, by the pseudorandomness of $\mathtt{F}$, with all but negligible probability, $C' \nsim_{1/\delta} \tilde{C}'$ for all $C' \in \mathcal{C}'$. That completes the proof of Lemma C.1. □

Now, we are ready to show the reduction. More precisely, assuming $\mathcal{A}$ is a PPT $4\sqrt{\delta}$-unforging-admissible adversary that can break the $4\sqrt{\delta}$-unforgeability of WM, then we construct a PPT δ-unforging-admissible adversary $\mathcal{B}$ that breaks the δ-unforgeability of WM' as follows.

The adversary $\mathcal{B}$ evokes the adversary $\mathcal{A}$, and answers $\mathcal{A}$'s marking oracle queries with his own marking oracle. In particular, on input a query $(k, \mathtt{m})$ from $\mathcal{A}$, $\mathcal{B}$ queries $(k, \mathtt{m})$ to its own marking oracle, and on receiving the response C', it returns the circuit C that $C(x_1, x_2) = C'(x_1) \oplus x_2$ to $\mathcal{A}$. In the end, after $\mathcal{A}$ outputs a circuit $\tilde{C}$, $\mathcal{B}$ samples $K \xleftarrow{\$} \{0,1\}^\kappa$ and returns

$$\tilde{C}'(x) = \tilde{C}(x, F_K(x)) \oplus F_K(x)$$

back to its challenger.

Now, let msk' be the marking secret key of the challenger of $\mathcal{B}$ and $msk = (msk', K)$. Note that $WM.Extract(msk, \tilde{C}) = WM'.Extract(msk', \tilde{C}')$. Also, here $\mathcal{B}$ can perfectly simulate the view of $\mathcal{A}$ in the unforgeability experiment, so $\mathcal{A}$ can submit a circuit that $WM.Extract(msk, \tilde{C}) \neq \perp$ with non-negligible probability. So, $\mathcal{B}$ can also submit a circuit that $WM'.Extract(msk', \tilde{C}') \neq \perp$ with non-negligible probability. Besides, by Lemma C.1, with all but negligible probability, $\mathcal{B}$ is δ-unforging admissible. Therefore, $\mathcal{B}$ is a PPT δ-unforging-admissible adversary that breaks the unforgeability of WM'.

That completes the proof of $4\sqrt{\delta}$-unforgeability.

Collusion Resistant Unremovability. Next, we prove the unremovability of WM. First, we define the following games:

• *Game 0.* This is the real experiment $ExptUR$ for the watermarkable PKE scheme.

• *Game 1.* This is identical to Game 0, except the way the challenger answers the challenge oracle. In particular, on input a set $\mathtt{M}$ to the challenge oracle, the challenger first samples the challenge secret key $\hat{k}$ and computes the marked keys $\hat{C}_i$ for $i \in [1, \|\mathtt{M}\|]$ as in Game 0, but then it computes the challenge public key $\hat{pk}$ from $\hat{C}_1$ instead of from $\hat{k}$, i.e. it computes $\hat{pk} \leftarrow iO(\mathsf{P}[\hat{C}_1])$, where P is defined in Figure 6.

Indistinguishability between Game 0 and Game 1 comes from the dispersibility of WM' and the indistinguishability of iO. First, by the dispersibility of WM', with all but negligible probability, the set $\{x \in \{0,1\}^n \mid \hat{C}_1(x) \neq F_{\hat{k}}(\cdot)\}$ and the range of G, which is at most $2^{n/2}$ and is negligible compared to 2^n, are disjoint. Thus, with all but negligible probability, $\hat{C}_1(\cdot)$ and $F_{\hat{k}}(\cdot)$ works identically in the obfuscated program in the public key of the PKE scheme. So, by the indistinguishability of iO, the adversary can distinguish Game 0 and Game 1 with only a negligible probability.

Constant: a marked decryption circuit C.
Input: the randomness $r \in \{0,1\}^{n/2}$ and the message $u \in \{0,1\}^m$

1. Let $t = G(r)$.
2. Output $c = (t, C(t, 0^m) \oplus u)$

Fig. 6 The circuit P.

Next, we will show that for any PPT unremoving-admissible adversary, the probability that it wins in Game 1 is negligible. We also first prove an auxiliary lemma.

Lemma C.2. *For any PPT unremoving-admissible adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\Pr\left[\begin{array}{l} msk' \leftarrow WM'.KeyGen(1^\lambda); \\ K \xleftarrow{\$} \{0,1\}^\kappa; \\ (\{\mathtt{m}_i\}_{i\in[1,Q]}, \sigma) \leftarrow \mathcal{A}_1^{\mathcal{O}_{msk'}(\cdot,\cdot)}(1^\lambda); \\ \hat{k} \xleftarrow{\$} \{0,1\}^\ell; \\ \textit{for } i \in [1,Q]: \\ \quad \hat{C'}_i \leftarrow WM'.Mark(msk', \hat{k}, \mathtt{m}_i); \\ \quad \hat{C}_i \textit{ is defined as :} \\ \quad \hat{C}_i(x_1, x_2) = \hat{C'}_i(x_1) \oplus x_2; \\ \hat{pk} \leftarrow iO(\mathsf{P}[\hat{C}_1]); \\ \tilde{C} \leftarrow \mathcal{A}_2^{\mathcal{O}_{msk'}(\cdot,\cdot)}(\sigma, \{\mathtt{m}_i\}_{i\in[1,Q]}, \{\hat{C}_i\}_{i\in[1,Q]}, pk); \\ \tilde{C}' \textit{ is defined as :} \\ \tilde{C}'(x) = \tilde{C}(x, F_K(x)) \oplus F_K(x) \end{array} \;:\; \exists i \in [1,Q], \hat{C'}_i \sim_{1/negl(\lambda)} \tilde{C}' \right] \geq 1 - negl(\lambda)$$

where Q is a polynomial and σ is the state of $\mathcal{A}_1$. Here, on input a query $(k, \mathtt{m})$ from $\mathcal{A}$, the oracle $\mathcal{O}_{msk'}$ computes $C' \leftarrow WM'.Mark(msk', k, m)$, generates a circuit C that $C(x_1, x_2) = C'(x_1) \oplus x_2$, and outputs C.

Proof. To prove Lemma C.2, we also first consider a mental game that $F_K(\cdot)$ is replaced with a random function $f \xleftarrow{\$} FUN_{n,m}$, namely, considering the (exponential-size) circuit $\check{C}' = \tilde{C}(x, f(x)) \oplus f(x)$ instead of $\tilde{C}'$. Note that for any $i \in [1,Q]$, $x \in \{0,1\}^n$, $\check{C}'(x) = \hat{C'}_i(x)$ iff $\tilde{C}(x, f(x)) = \hat{C}_i(x, f(x))$. Then, for any $i \in [1,Q]$ let

$$\varepsilon = \frac{\|\{(x,y) \in \{0,1\}^n \times \{0,1\}^m \mid \tilde{C}(x,y) \neq \hat{C}_i(x,y)\}\|}{2^{n+m}}$$

Also, for $x \in \{0,1\}^n$, let

$$\varepsilon_x = \frac{\|\{y \in \{0,1\}^m \mid \tilde{C}(x,y) \neq \hat{C}_i(x,y)\}\|}{2^m}.$$

Again, we have $\sum_{x\in\{0,1\}^n} \varepsilon_x = 2^n \cdot \varepsilon$. Next, for any function p in λ, we define the set $\mathcal{S}_p = \{x \mid \varepsilon_x \geq p\varepsilon\}$. Thus, we have

$$2^n \cdot \varepsilon = \sum_{x\in\{0,1\}^n} \varepsilon_x \geq \|\mathcal{S}_p\| \cdot p\varepsilon$$

which implies that $\|\mathcal{S}_p\| \leq 2^n/p$. Now, let $\mathcal{D}$ be the random variable indicating the number of inputs that are evaluated differently by $\check{C}'$ and $\hat{C'}_i$, where the probability is taken over the choice of f. Also, for $x \in \{0,1\}^n$, let $\mathcal{D}_x$ be the random variable indicating whether $\check{C}'(x) \neq \hat{C'}_i(x)$ (1 for not equal), where the probability is taken over the choice of f. Next, we conisder two cases:

– **Case I:** $\varepsilon \geq \frac{1}{2^n}$**.** In this case, we set $p = \frac{1}{\sqrt{\varepsilon}}$, $\mathcal{S} = \mathcal{S}_{1/\sqrt{\varepsilon}}$ and $\mathcal{E}_x$ be the random variable that being 1 with probability $\sqrt{\varepsilon}$. Then we have $\|\mathcal{S}\| \leq 2^n \cdot \sqrt{\varepsilon}$ and

$$\begin{aligned}
\Pr[\mathcal{D} \geq 3 \cdot 2^n \cdot \sqrt{\varepsilon}] &= \Pr[\sum_{x\in\{0,1\}^n} \mathcal{D}_x \geq 3 \cdot 2^n \cdot \sqrt{\varepsilon}] \\
&\leq \Pr[\|\mathcal{S}\| + \sum_{x\notin\mathcal{S}} \mathcal{D}_x \geq 3 \cdot 2^n \cdot \sqrt{\varepsilon}] \\
&\leq \Pr[2^n \cdot \sqrt{\varepsilon} + \sum_{x\notin\mathcal{S}} \mathcal{E}_x \geq 3 \cdot 2^n \cdot \sqrt{\varepsilon}] \\
&= \Pr[\sum_{x\notin\mathcal{S}} \mathcal{E}_x \geq 2 \cdot 2^n \cdot \sqrt{\varepsilon}] \\
&\leq \Pr[\sum_{x\in\{0,1\}^n} \mathcal{E}_x \geq 2 \cdot 2^n \cdot \sqrt{\varepsilon}] \\
&\leq e^{-2^n \cdot \sqrt{\varepsilon}/3}
\end{aligned}$$

  which is negligible (since $2^n \cdot \sqrt{\varepsilon}/3 \geq 2^{n/2}/3$), where the last inequality follows the chernoff bound as $\mathbb{E}(\sum_{x\in\{0,1\}^n} E_x) = 2^n \cdot \sqrt{\varepsilon}$.
– **Case II:** $\varepsilon < \frac{1}{2^n}$**.** In this case, we set $p = \frac{1}{2^{n/2}\cdot\varepsilon}$, $\mathcal{S} = \mathcal{S}_{1/(2^{n/2}\cdot\varepsilon)}$ and $\mathcal{E}_x$ be the random variable that being 1 with probability $1/2^{n/2}$. Then we have $\|\mathcal{S}\| \leq 2^{3n/2} \cdot \varepsilon$ and

$$\begin{aligned}
\Pr[\mathcal{D} \geq 3 \cdot 2^{n/2}] &= \Pr[\sum_{x\in\{0,1\}^n} \mathcal{D}_x \geq 3 \cdot 2^{n/2}] \\
&\leq \Pr[\|\mathcal{S}\| + \sum_{x\notin\mathcal{S}} \mathcal{D}_x \geq 3 \cdot 2^{n/2}] \\
&\leq \Pr[2^{3n/2} \cdot \varepsilon + \sum_{x\notin\mathcal{S}} \mathcal{E}_x \geq 3 \cdot 2^{n/2}] \\
&\leq \Pr[2^{n/2} + \sum_{x\notin\mathcal{S}} \mathcal{E}_x \geq 3 \cdot 2^{n/2}] \\
&= \Pr[\sum_{x\notin\mathcal{S}} \mathcal{E}_x \geq 2 \cdot 2^{n/2}] \\
&\leq \Pr[\sum_{x\in\{0,1\}^n} \mathcal{E}_x \geq 2 \cdot 2^{n/2}] \\
&\leq e^{-2^{n/2}/3}
\end{aligned}$$

  which is negligible, where the inequality at the fourth line follows the fact $\varepsilon \leq 1/2^n$ and the last inequality follows the chernoff bound as $\mathbb{E}(\sum_{x\in\{0,1\}^n} E_x) = 2^{n/2}$.

In general, let $\mu = max(\sqrt{\varepsilon}, 2^{-n/2})$, then we have $\Pr[\mathcal{D} \geq 3 \cdot 2^n \cdot \mu] \leq negl(\lambda)$. By the union bound, we have with all but negligible probability, for all $i \in [1, Q]$, $\check{C}' \sim_{1/(3\mu)} \hat{C}'_i$.

Next, we turn back to the circuit $\tilde{C}'$ and show that with all but negligible probability, there exists $i \in [1, Q]$ that $\tilde{C}' \sim_{1/negl(\lambda)} \hat{C}'_i$. Assume that in contrast with a non-negligible probability, $\tilde{C}' \nsim_{poly(\lambda)} \hat{C}'_i$ for all $i \in [1, Q]$, then we can construct an adversary $\mathcal{B}'$ that breaks the pseudorandomness of $\mathtt{F}$. The adversary $\mathcal{B}'$ works similarly to the adversary

$\mathcal{B}'$ constructed in the proof of the unforgeability of watermarkable PKE schemes. However, to detect whether there are $1/p$ fraction of inputs that are evaluated differently by $\tilde{C}'$ and $\hat{C}'_i$ for a polynomial p that is not apriori bounded, the adversary $\mathcal{B}'$ for $\mathtt{F}$ need to run in super-polynomial time. So we need $\mathtt{F}$ to have quasi-polynomial security.

In particular, $\mathcal{B}'$ first run the experiment defined in the body of Lemma C.2 with $\mathcal{A}$ and receives a circuit $\tilde{C}$ (but it does not sample the secret key K of $\mathtt{F}$). Then, for $j \in [1, Q]$, assuming $\tilde{C}$ and $\hat{C}_j$ evaluate differently on ε_j fraction of inputs and define $\mu_j = max(\sqrt{\varepsilon_j}, 2^{-n/2})$, $\mathcal{B}'$ runs the following procedure:

1. $\mathcal{B}'$ attempts to estimate ε_j (and thus μ_j) via running $\tilde{C}$ and $\hat{C}'_j$ on a large quasi-polynomial number of inputs, it can give an estimate of ε_j if ε_j is not very small and if the estimation fails it can safely assume that ε_j is a small enough negligible function.
2. It moves to the next repetition if ε_j is not negligible; otherwise, it continues the following procedure.
3. It samples $d_j = 1/\xi_j$ random inputs $\{x_i\}_{i\in[1,d]} \xleftarrow{\$} (\{0,1\}^n)^d$, where ξ_j is a negligible function that is inverse-quasi-polynomial and is super-polynomial larger than $3 \cdot \mu_j$, namely, $3 \cdot \mu_j \cdot \lambda^{\omega(1)} \leq \xi_j \leq 1/\lambda^{O(\log^c \lambda)}$ for a proper positive constant c.
4. It queries each x_i to its oracle and obtains the set $\{y_i\}_{i\in[1,d]}$.
5. It outputs 0 and aborts if for for all $i \in [1, d_j]$, $\tilde{C}(x_i, y_i) = \hat{C}_1(x_i, y_i)$.

Finally, if $\mathcal{B}'$ completes the repetition without aborting, it outputs 1.

Note that if $\mathcal{B}'$ is answered with a random function f, it can perfectly simulate the view of $\mathcal{A}$ in the mental game, and for each $j \in [1, Q]$ and $i \in [1, d]$, $\tilde{C}(x_i, y_i) = \check{C}'(x_i) \oplus y_i$ and $\hat{C}_j(x_i, y_i) = \hat{C}'_j(x_i) \oplus y_i$, i.e. $\tilde{C}(x_i, y_i) \neq \hat{C}_j(x_i, y_i)$ iff $\check{C}'(x_i) \neq \hat{C}'_j(x_i)$. Since the adversary $\mathcal{A}$ is unremoving-admissible, with all but negligible probability, there exists $j \in [1, Q]$ that $\hat{C}_j \sim_{1/negl(\lambda)} \tilde{C}$, i.e. with all but negligible probability, there exists repetition goes into the third step in the above repeated procedure. As with all but negligible probability, for all $j \in [1, Q]$, $\check{C}' \sim_{1/(3\mu_j)} \hat{C}'_j$, by the union bound, in a repetition that goes into the third step, the probability that there exists $i \in [1, d_j]$ that $\check{C}'(x_i) \neq \hat{C}'_j(x_i)$ does not exceed $3\mu_j \cdot d_j = 3\mu_j/\xi_j \leq 1/\lambda^{\omega(1)}$, which is negligible. So, $\mathcal{B}'$ will output 0 with all but negligible probability in this case. In contrast, if $\mathcal{B}'$ is answered by the pseudorandom function with a key K, it can perfectly simulate the view of $\mathcal{A}$ in the game defined in the body of Lemma C.2, and for each $j \in [1, Q]$, $i \in [1, d_j]$, $\tilde{C}(x_i, y_i) = \tilde{C}'(x_i) \oplus y_i$ and $\hat{C}_j(x_i, y_i) = \hat{C}'_j(x_i) \oplus y_i$, i.e. $\tilde{C}(x_i, y_i) \neq \hat{C}_j(x_i, y_i)$ iff $\tilde{C}'(x_i) \neq \hat{C}'_j(x_i)$. As it is assumed that for all $j \in [1, Q]$, $\tilde{C}' \not\sim_{1/p} \hat{C}'_j$ for some p, for each j, (even the repetition goes into the third step,) the probability that there does not exists $i \in [1, d]$ that $\tilde{C}'(x_i) \neq \hat{C}'_j(x_i)$ is $(1 - 1/p)^{1/\xi_j} \leq e^{-1/(p\xi_j)} \leq e^{-\lambda^{\omega(1)}}$, which is negligible. By the union bound, $\mathcal{B}'$ will output 0 with only a negligible probability in this case. Therefore, by the (quasi-polynomial)-pseudorandomness of $\mathtt{F}$, with all but negligible probability, there exists $j \in [1, Q]$ that $\tilde{C}' \sim_{negl(\lambda)} \hat{C}'_j$. That completes the proof of Lemma C.2. □

Now we are ready to show that no PPT unremoving-admissible adversary can win in Game 1 with a non-negligible probability. More precisely, assuming $\mathcal{A}$ is a PPT

unremoving-admissible adversary that can win in Game 1 with a non-negligible probability, then we construct a PPT unremoving-admissible adversary $\mathcal{B}$ that breaks the collusion resistant unremovability of WM' as follows.

The adversary $\mathcal{B}$ evokes the adversary $\mathcal{A}$, and answers $\mathcal{A}$'s oracle queries with his own oracle. In particular, on input a query $(k, \mathtt{m})$ to the marking oracle from $\mathcal{A}$, $\mathcal{B}$ queries $(k, \mathtt{m})$ to its own marking oracle, and on receiving the response C', it returns $C(x_1, x_2) = C'(x_1) \oplus x_2$ to $\mathcal{A}$. Also, on input a query $\mathtt{M}$ to the challenge oracle from $\mathcal{A}$, $\mathcal{B}$ also queries $\mathtt{M}$ to its own challenge oracle, and on receiving the response set $\{\hat{C}'_i\}_{i\in[1,\|\mathtt{M}\|]}$, it returns a set $\{\hat{C}_i\}_{i\in[1,\|\mathtt{M}\|]}$ as well as a public key $\hat{pk} \leftarrow iO(\mathsf{P}[\hat{C}_1])$, where $\hat{C}_i(x_1, x_2) = \hat{C'}_i(x_1) \oplus x_2$ for $i \in [1, \|\mathtt{M}\|]$. In the end, after $\mathcal{A}$ outputs a circuit $\tilde{C}$, $\mathcal{B}$ samples $K \xleftarrow{\$} \{0,1\}^\kappa$ and returns the circuit

$$\tilde{C'}(x) = \tilde{C}(x, F_K(x)) \oplus F_K(x)$$

back to its challenger.

Now, let msk' be the marking secret key of the challenger of $\mathcal{B}$ and $msk = (msk', K)$. Note that $WM.Extract(msk, \tilde{C}) = WM'.Extract(msk', \tilde{C'})$. Also, here $\mathcal{B}$ can perfectly simulate the view of $\mathcal{A}$ in the unremovability experiment, so $\mathcal{A}$ can submit a circuit that $WM.Extract(msk, \tilde{C}) \notin \mathtt{M}$ with non-negligible probability. So, $\mathcal{B}$ can also submit a circuit that $WM'.Extract(msk', \tilde{C'}) \notin \mathtt{M}$ with non-negligible probability. Besides, by Lemma C.2, with all but negligible probability, $\mathcal{B}$ is unremoving admissible. Therefore, $\mathcal{B}$ is a PPT unremoving-admissible adversary that breaks the collusion resistant unremovability of WM'.

That completes the proof of collusion resistant unremovability. □