Collusion Resistant Watermarking Schemes for Cryptographic Functionalities

Rupeng Yang[1,2], Man Ho Au[2⋆], Junzuo Lai[3⋆], Qiuliang Xu[4⋆], and Zuoxia Yu[2]

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

Abstract. A cryptographic watermarking scheme embeds a message into a program while preserving its functionality. Recently, a number of watermarking schemes have been proposed, which are proven secure in the sense that given *one* marked program, any attempt to remove the embedded message will substantially change its functionality.

In this paper, we formally initiate the study of collusion attacks for watermarking schemes, where the attacker's goal is to remove the embedded messages given *multiple* copies of the same program, each with a different embedded message. This is motivated by practical scenarios, where a program may be marked multiple times with different messages.

The results of this work are twofold. First, we examine existing cryptographic watermarking schemes and observe that all of them are vulnerable to collusion attacks. Second, we construct collusion resistant watermarking schemes for various cryptographic functionalities (e.g., pseudorandom function evaluation, decryption, etc.). To achieve our second result, we present a new primitive called puncturable functional encryption scheme, which may be of independent interest.

Keywords: Watermarking, Watermarkable PRF, Collusion Resistance, Public Extraction

⋆ Corresponding author.

1 Introduction

A watermarking scheme allows one to embed some information into a program[1] without significantly changing its functionality. It has many natural applications, including ownership protection, information leaker tracing, etc.

The formal definition of watermarking schemes for programs is first presented by Barak et al. in [BGI+01]. Subsequently, new properties of watermarking schemes are presented in [HMW07, NW15, CHV15]. They are briefly summarized below.

- **Unremovability:** This is the essential security property for watermarking schemes, which requires that it should be hard to *remove* or *modify* the embedded information in a marked program without destroying it.
- **Public Extraction:** Anyone should be able to extract the embedded information in a marked program. In other words, the extraction key will be made public.
- **Public Marking:** Anyone should be able to embed information into a program. In other words, the marking key will be made public.
- **Unforgeability:** Only the authorized entity who holds the marking key should be able to embed information into a program. Obviously, it requires keeping the marking key secret and is not compatible with the "public marking" property.
- **Message-Embedding:** This property allows one to embed a given string (instead of merely a mark symbol) into the watermarked program.

Despite being a natural concept and perceived to have a wide range of applications, watermarking schemes provably secure against arbitrary removal strategies were not presented until 2015. In [CHN+16] (which is a merged version of [NW15] and [CHV15]), Cohen et al. construct a publicly extractable watermarking scheme for the evaluation algorithm of pseudorandom functions (PRFs) from indistinguishability obfuscators. Based on the watermarkable PRF families, they also construct watermarkable public key encryption (PKE) schemes and watermarkable signature schemes. However, Cohen et al.'s schemes do not achieve standard unforgeability. Subsequently, Yang et al. [YAL+18] improve the watermarkable PRF in [CHN+16] to achieve both standard unforgeability and public extraction simultaneously.

In another line of research, initiated by Boneh et al. in [BLW17], watermarkable PRFs are constructed from variants of constraint-hiding constrained PRFs (e.g., privately programmable PRF and translucent puncturable PRF). Boneh et al.'s scheme is constructed from privately programmable PRF, which is instantiated from indistinguishability obfuscator in [BLW17]. Subsequently, based on a translucent puncturable PRF, Kim and Wu [KW17] present the first construction of watermarkable PRF from standard lattice assumptions. Then,

[1] In this paper, we focus on watermarking schemes for programs and only consider those with provable security against arbitrary removal strategies. We refer readers to Sec. 1.2 for an extended introduction to the area.

Peikert and Shiehian [PS18] construct privately programmable PRF from LWE, which provides another way to instantiate watermarkable PRF from standard assumptions. Recently, in [QWZ18] and [KW19], watermarkable PRFs with public marking are constructed from constraint-hiding constrained PRF and puncturable extractable PRF respectively, both of which can be instantiated from standard lattice assumptions.

Besides, a very simple yet elegant construction of watermarking scheme for any PKE scheme is presented by Baldimtsi et al. [BKS17]. However, their scheme does not support multi-message-embedding inherently. That is, each program can only be marked with at most one message during the life-time of the scheme.

Collusion Resistance of Watermarking schemes. In practical applications, it is usually required that unremovability of watermarking schemes should hold under "collusion attacks", where the attacker can access several copies of the same program embedded with different information. As a concrete example, consider the following scenario. A software development company developed a program and would like to outsource its testing to several organizations. To prevent these organizations from leaking the program, the company will employ a watermarking scheme to embed the name of the target organization into the copy being sent. Here, the watermarking scheme should enable the company to trace program leakers even when a few target organizations collude.

However, for all previous watermarking schemes [CHN$^+$16, BLW17, KW17, BKS17,PS18,YAL$^+$18,QWZ18,KW19], the unremovability is only proved against an adversary who attempts to remove or modify the embedded message given a *single* marked program[2], and it is unknown whether they are secure against collusion attacks. Thus, the following question arises naturally:

Can we build collusion resistant watermarking scheme?

1.1 Our Results

In this paper, we explore the existence of watermarking schemes secure against collusion attacks. First, we observe that unfortunately, all existing watermarking schemes are *vulnerable* to collusion attacks (we elaborate this in Sec. 2). Then, we consider how to develop watermarkable cryptographic primitives secure against the collusion attacks. Specifically, our contributions are as follows.

- We present the notion of *collusion resistant watermarking scheme* to capture collusion attacks. It requires a stronger unremovability (namely, collusion resistant unremovability) that allows the adversary to obtain watermarked circuits embedded with different messages for the same functionality.

[2] In a concurrent work [GKM$^+$19], collusion resistant watermarking schemes for public-key cryptographic primitives are presented. However, their constructions are under a relaxed notion of functionality-preserving. In this work, we achieve collusion resistance while preserving the original "statistical functionality-preserving" proposed in [CHN$^+$16].

- We give a construction of *collusion resistant watermarkable PRF*, which is the first watermarkable cryptographic primitive provably secure against the collusion attacks. To achieve this, we introduce a new message-embedding technique in the watermarking setting and propose a new primitive, namely, *puncturable functional encryption scheme*, which we believe will find additional applications in constructing advanced cryptographic primitives.
- Based on our construction of collusion resistant watermarkable PRF, we also construct watermarkable PKE schemes and watermarkable signature schemes, both of which have collusion resistant unremovability.

We compare the main features achieved by current watermarking schemes and our watermarking schemes in Table 1. We remark that in addition to collusion resistance, our schemes can achieve many desirable features, including public extraction, unforgeability, and message-embedding.

Table 1: The Comparison.

		Unforgeability	Public Extraction	Public Marking	Message Embedding	Collusion Resistance
[CHN+16]	PRF	✗	✓	✗	✓	✗
	PKE	✗	✓	✗	✓	✗
	Signature	✗	✓	✗	✓	✗
[YAL+18]	PRF	✓	✓	✗	✓	✗
[BLW17]	PRF	✓	✗	✗	✓	✗
[KW17]	PRF	✓	✗	✗	✓	✗
[QWZ18]	PRF	✗	✗	✓	✓	✗
[KW19]	PRF	✓	✗	✗	✓	✗
	PRF	✓†	✗	✓	✓	✗
[BKS17]	PKE	✓	✗	✗	✗	-
	PKE	✓	✓	✗	✗	-
This work	PRF	✓	✓	✗	✓	✓
	PKE	✓	✓	✗	✓	✓
	Signature	✓	✓	✗	✓	✓

†: Weaker versions of unforgeability are achieved for this scheme.

The presented collusion resistant watermarking schemes are built on several cryptographic primitives, which can be constructed from indistinguishability obfuscator and standard lattice assumptions.

Theorem 1.1 (Informal). *Assuming the worst-case hardness of appropriately parameterized GapSVP and SIVP problems and the existence of indistinguishability obfuscator, there exist collusion resistant watermarkable PRF/PKE/signature schemes.*

Remark 1.1. It is worth noting that our constructions of collusion resistant watermarking schemes rely on the existence of indistinguishability obfuscator. However, this seems essential, at least for collusion resistant watermarkable PRF. To see this, recall that as proved in [BGI+01], watermarking scheme perfectly preserving the functionality of the watermarked program does not exist. Thus, a marked key of PRF must evaluate differently with the original key on some inputs, i.e., the marked key can be viewed as a constrained key of the original key. Besides, the marked key should hide its constrained inputs, since otherwise, the attacker is likely to remove the embedded messages via resetting outputs on constrained inputs. Therefore, we can approximately view a collusion resistant watermarkable PRF as a collusion resistant constraint-hiding constrained PRF, which, as shown in [CC17], can imply indistinguishability obfuscator. Nonetheless, we are not able to formalize this intuition. It is an interesting open problem to give a formal construction of indistinguishability obfuscator from collusion resistant watermarkable PRF.

1.2 Related Works

Additional Related Works on Watermarking Schemes. In this paper, we concentrate on watermarking schemes provably secure against arbitrary removal strategies. There are also numerous works (see [CMB+07] and references therein) attempting to use ad hoc techniques to watermark a wide class of digital objects, such as images, audios, videos, etc. However, these constructions lack rigorous security analysis and are (potentially) vulnerable to some attacks.

In another line of research [NSS99, YF11, Nis13], watermarking schemes for cryptographic objects (e.g., the key, the signature, etc.) are constructed and rigorously analyzed. However, their security definition considers a restricted adversary that will not change the format of the watermarked objects.

Puncturable Symmetric Key Functional Encryption. One byproduct of this work is a new primitive called puncturable functional encryption. A similar primitive, which is called puncturable symmetric key functional encryption, is also studied in previous works [BV15, KNT18]. In particular, it is used to construct the indistinguishability obfuscator in these works.

We stress that these two types of primitives are incomparable. First, while succinctness is the key property for a puncturable symmetric key functional encryption scheme, it is not required in our puncturable functional encryption scheme. Thus, our scheme cannot be used in constructions of indistinguishability obfuscators. On the other hand, our puncturable functional encryption scheme will puncture a secret key on a ciphertext, but in a puncturable symmetric key functional encryption scheme, secret keys are punctured on a message or on a tag. Thus, their schemes are also inapplicable to our setting.

Traitor Tracing Scheme. The notion of collusion resistant watermarking scheme is somewhat similar to the notion of traitor tracing scheme, which aims at tracing secret key leakers among a set of users holding functionally equivalent secret keys in a broadcast encryption setting. Since first presented in [CFN94], traitor tracing has been formally studied for a long time (see e.g., [BSW06, BN08, BZ14, NWZ16, GKW18, CVW+18] and references therein for an overview of previous works).

Generally, in a traitor tracing scheme, there is a common public key pk and each user holds a different secret key. Data encrypted under the common public key can be decrypted by all users in the system. Moreover, there exists a tracing algorithm, which outputs a set of users on input a "pirate decoder" that can decrypt ciphertexts under pk. It is guaranteed that the tracing algorithm can identify at least one of the users in the coalition that produces the pirate decoder.

Comparing watermarking and traitor tracing. Both (collusion resistant) watermarking and traitor tracing will issue copies of a program (or a key), which are embedded with some information, to users and aim at recovering the embedded information from a functionally-similar program/key generated by them. However, solutions to the traitor tracing problem do not yield watermarking schemes directly, since these two notions also have several inherent differences.

First, in a traitor tracing scheme, secret keys of users are issued by a center, while in a watermarking scheme, user can choose their watermarked programs themselves. Another difference is that in a traitor tracing scheme, secret keys of all users are functionally equivalent, while in a watermarking scheme, programs with different functionalities can be watermarked in the same watermarking scheme. Besides, traitor tracing schemes focus on tracing secret key leakers in an encryption scheme, while watermarking schemes aim at marking general purpose programs (although we only know how to watermark some specific cryptographic functionalities currently).

A closer look at how to construct traitor tracing schemes. In [BSW06], Boneh et al. present a classic paradigm to construct traitor tracing schemes, which is also used or adapted in many subsequent works [BZ14, NWZ16, GKW18]. The construction proceeds in two steps.

First, a private linear broadcast encryption (PLBE) scheme is constructed. Recall that a PLBE scheme has a sequence $(sk_1, \ldots, sk_N)$ of N secret keys for a public key and each ciphertext is labeled with an integer in $[0, N]$. A secret key sk_i is only able to decrypt a ciphertext with label j when $j < i$. Thus, a ciphertext with label 0 can be decrypted by all secret keys, while a ciphertext with label N can not be decrypted by any secret key. Also, it is required that it is computationally infeasible to distinguish a ciphertext with label j and that with label $j - 1$ if sk_j is not given.

A PLBE scheme implies a traitor tracing scheme [BSW06, GKW18]. More concretely, the traitor tracing scheme supports a user set of size N and the ith user in that set is given secret key sk_i. Broadcast messages will be encrypted with label 0. When tracing colluders from a pirate decoder, the tracing algorithm feeds the decoder with ciphertexts labeled with 0 to N sequentially and

outputs i if there exists a "large gap" in decryption success probability between ciphertexts labeled with $i-1$ and those labeled with i. Note that the decoder can decrypt with a high success probability on ciphertext labeled with 0 (due to the usefulness of the decoder) and can decrypt with a negligible success probability on ciphertext labeled with N (due to the security of PLBE), thus, there must exists a large gap in decryption success probability between $i-1$ and i for some $i \in [N]$. Also, as no one could distinguish ciphertexts labeled with $i-1$ and that labeled with i without sk_i, the large gap must occur between $i-1$ and i such that the colluders possess sk_i. Therefore, the tracing algorithm can recover at least one of the colluders.

1.3 Roadmap

The rest of the paper is organized as follows. We give an overview of our construction in Sec. 2. Then in Sec. 3, we review notations used in this work. We present the formal definition of collusion resistant watermarkable PRF in Sec. 4. Then in Sec. 5, we define and construct puncturable functional encryption, which is employed to construct collusion resistant watermarkable PRF. We show our main construction of collusion resistant watermarkable PRF in Sec. 6 and present constructions of collusion resistant watermarking schemes for public key primitives in Sec. 7. Finally, in Sec. 8, we conclude our work with a few possible future works.

2 Technical Overview

In this section, we provide an overview of our construction of collusion resistant watermarkable PRF. Our starting point is the watermarking scheme WM_0 presented in [CHN+16] (or more accurately, its variant in [YAL+18]). We first explain why WM_0 (and all previous watermarking schemes) are not collusion resistant and describe the main challenges in achieving collusion resistance. Then we give a high-level idea on how to address these challenges.

A brief overview of WM_0**.** Roughly speaking, WM_0 works as follows. The extraction key/marking key pair of WM_0 is a public key/secret key pair (pk, sk) of a PKE scheme. To embed a message msg into a PRF key k, the marking algorithm outputs an obfuscation of the following circuit, which evaluates the function $\mathsf{PRF}(k, \cdot)$ correctly at all points, except for some "punctured points".

$$\mathtt{C}(x) = \begin{cases} f(\mu) \oplus msg & \text{if } \mu = \mathtt{Dec}(sk, x) \in \mathcal{V} \\ \mathsf{PRF}(k, x) & \text{otherwise.} \end{cases}$$

Here, $\mathtt{Dec}$ is the decryption algorithm of the underlying PKE scheme, $\mathcal{V}$ is a set defined by the PRF key k and f is a suitable function.

When extracting the embedded message from a watermarked circuit, the extraction algorithm first samples a string $\mu \in \mathcal{V}$ and encrypts it with the public

key pk. Next, it evaluates the circuit on the ciphertext and obtains an output z. Finally, it computes $msg = z \oplus f(\mu)$. The above extraction procedure will be repeated multiple times and the extraction algorithm will output the majority result or an "UNMARKED" symbol if no majority is found.

Security of the scheme relies on the fact that punctured points (i.e. those decrypted to a string in $\mathcal{V}$) are hidden[3]. As a result, the adversary, who is only allowed to alter the marked circuit slightly, is not able to change the output values on a large enough fraction of punctured points, and thus the extraction algorithm can still extract the correct message.

Why WM_0 ***is not collusion resistant?*** However, if watermarked circuits embedded with different messages for the same PRF key k are given, one can easily locate all punctured points via comparing the outputs of the circuits. In addition, it is easy to modify or remove the embedded messages via resetting outputs on all punctured points.

In more detail, given two circuits $\mathtt{C}_1$ and $\mathtt{C}_2$ that are generated by embedding different messages, say msg_1 and msg_2, into the same PRF key k, an attacker can output a circuit $\mathtt{C}^*$ embedded with a new message msg^* as follow:

$$\mathtt{C}^*(x) = \begin{cases} \mathtt{C}_1(x) \oplus msg_1 \oplus msg^* & \text{if } \mathtt{C}_1(x) \neq \mathtt{C}_2(x). \\ \mathtt{C}_1(x) & \text{otherwise.} \end{cases}$$

It is not hard to see that $\mathtt{C}^*$ will compute the PRF with key k correctly on almost all inputs except that it will output $f(\mu) \oplus msg^*$ on an input whose decryption μ is in $\mathcal{V}$. Therefore, the attacker can compromise the unremovability of WM_0 via a collusion attack[4].

Since nearly all[5] previous watermarking schemes are constructed following the blueprint proposed in [CHN+16], we can use a similar strategy to show that they are not collusion resistant. We stress that all collusion attacks are based on the fragility of the way messages are embedded and do not take advantage of the concrete instantiations of the schemes.

The challenge in achieving collusion resistance. To better explain why WM_0 is not able to achieve collusion resistance, we describe WM_0 in a modular manner.

In a high level, on input a PRF key k and a message msg, the marking algorithm of WM_0 works as follows:

[3] One could find some punctured points via generating them from public information, but cannot distinguish a random punctured point from a random point in the input space.

[4] We remark that this will not affect the claimed security of WM_0. The attacks only show that WM_0 is not applicable in scenarios where collusion attacks are a legit threat.

[5] The watermarking scheme proposed in [BKS17] is constructed in a different approach, however, it cannot embed different messages into the same program.

1. Generates two sequences $\mathcal{X} = (x_1, \ldots, x_l)$ and $\mathcal{Y} = (y_1, \ldots, y_l)$, where x_i and y_i are in the input space and the output space of the watermarked PRF respectively. More concretely, in WM_0, for each pair (x_i, y_i), $x_i = \mathtt{Enc}(pk, \mu)$ and $y_i = f(\mu)$ for some $\mu \in \mathcal{V}$.
2. Encodes the message msg into a sequence $\mathcal{Z} = (z_1, \ldots, z_l) = \mathtt{encode}(\mathcal{X}, \mathcal{Y}, msg)$, where z_i is also in the output space of the watermarked PRF. In more detail, messages are encoded into $\mathcal{Z}$ via a simple "exclusive or" operation in WM_0, i.e., $z_i = y_i \oplus msg$ for $i \in [1, l]$.
3. Outputs a circuit that computes the PRF with k correctly on inputs outside $\mathcal{X}$ and outputs z_i on input x_i (here, x_i is called a punctured point).

Correspondingly, we can abstract the extraction algorithm of WM_0, which takes as input a watermarked circuit $\mathtt{C}$, as follows:

1. Samples a set of pairs $\{x_i, y_i\}$ in $\mathcal{X} \times \mathcal{Y}$.
2. Evaluates $z_i = \mathtt{C}(x_i)$ for each x_i.
3. Recovers the message $msg = \mathtt{decode}(\{x_i, y_i, z_i\})$. Here, the decoding algorithm outputs the majority of $y_i \oplus z_i$.

The key observation underlying our collusion attack is that the simple "xor" encoding scheme is fragile in the collusion setting. First, for two different messages msg and msg', let $(z_1, \ldots, z_l) = \mathtt{encode}(\mathcal{X}, \mathcal{Y}, msg)$ and $(z'_1, \ldots, z'_l) = \mathtt{encode}(\mathcal{X}, \mathcal{Y}, msg')$, then we have $z_i \neq z'_i$ for $i \in [1, l]$. This makes it easy to locate all punctured points in $\mathcal{X}$ by comparing outputs of circuits embedded with different messages. In addition, it is easy to overwrite the encoded message in a codeword $\mathcal{Z} = (z_1, \ldots, z_l)$. For example, one can reset $z_i = z_i \oplus \Delta$ for $i \in [1, l]$ to xor the encoded message with Δ.

In [KW17, QWZ18, KW19], different message encoding schemes are applied. However, all of them inherit the aforementioned weakness to some extent, and thus are not robust against collusion attacks.

To solve this problem, we need to develop a robust message encoding scheme, where `decode` can recover the original embedded messages even if a collusion attacker can locate some punctured points[6] and will reset outputs on its located punctured points. Next, we explore how to develop a robust message encoding scheme and integrate it with the other part of WM_0.

Addressing the challenge: a robust message encoding scheme. We design our encoding scheme via using ideas from the realm of traitor tracing. In particular, our scheme is inspired by the well-known framework presented in [BSW06] (we recall this framework in Sec. 1.2).

The message space of our encoding scheme is $[1, N]$[7]. The input of the encoding algorithm is two sequences $\mathcal{X} = (x_1, \ldots, x_l), \mathcal{Y} = (y_1, \ldots, y_l)$ and a message

[6] This seems unavoidable since circuits embedded with different messages should be run differently on some points to enable message extraction.

[7] Here, we assume that N is polynomial in the security parameter and will show how to remove this restriction later.

$msg \in [1, N]$. Here, we divide the whole sequence $\mathcal{X}$ into N parts, namely, $\mathcal{X}_1$, $\ldots, \mathcal{X}_N$, each of which is labeled with an index in $[1, N]$ (we elaborate how to define $\mathcal{X}_i$ later). To encode a message msg, the encoding algorithm sets $z_i = y_i$ if $x_i \in \mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \mathcal{X}_{msg}$ and sets z_i to be the correct PRF output otherwise. The output of the encoding algorithm is the sequence $(z_1, \ldots, z_l)$.

We also modify the decoding algorithm. It takes as input a set of tuples (x_i, y_i, z_i), where (x_i, y_i) is sampled from $\mathcal{X} \times \mathcal{Y}$ and z_i is the output of the tested circuit on input x_i, and works as follows:

1. Set $p_0 = 1$ and $p_{N+1} = 0$.
2. For $ind \in [1, N]$, estimate the fraction p_{ind} of "correctly reprogrammed" points in set $\mathcal{X}_{ind}$, where a point x_i is "correctly reprogrammed" if $y_i = z_i$. This can be accomplished via testing polynomially-many points in $\mathcal{X}_{ind}$.
3. If there exists $ind \in [0, N]$ such that $|p_{ind} - p_{ind+1}|$ is noticeable (i.e., a "large gap" at ind is found), output the message $msg = ind$. Here, $msg = 0$ denotes the code is not decodable (i.e., the circuit is unmarked).

Next, we argue why our new message encoding scheme is robust under collusion attacks. Observe that, given a few (say 2) circuits $\mathtt{C}_1$ and $\mathtt{C}_2$ embedded with messages msg_1 and msg_2 respectively (w.l.o.g. assuming $msg_1 < msg_2$), the attacker can locate punctured points in $\mathcal{X}_{ind}$ for $ind \in (msg_1, msg_2]$ by comparing outputs of $\mathtt{C}_1$ and $\mathtt{C}_2$. However, we note that

- If the attacker cannot distinguish punctured points in $\mathcal{X}_{ind_1}$ and $\mathcal{X}_{ind_2}$ for $ind_1, ind_2 \in (msg_1, msg_2]$, it cannot make $|p_{ind_1} - p_{ind_2}|$ noticeable via resetting outputs on located punctured points.
- If the attacker cannot distinguish a punctured point $x_i \in \mathcal{X}_{ind}$ from a random point for $ind \notin (msg_1, msg_2]$, it will not be able to reset the output on such x_i. Thus, we have $p_{ind} = 1$ for $ind \in [1, msg_1]$ and $p_{ind} = 0$ for $ind \in (msg_2, N]$.

Consequently, if the aforementioned indistinguishability properties are guaranteed, the large gap(s) must occur at either msg_1 or msg_2 (or at both points), i.e., the decoding algorithm could output the embedded message(s).

One problem of the above solution is that the message space is restricted to be a polynomial-size set. This is because the decoding algorithm needs to scan all indices linearly to find a large gap. Addressing this problem, we employ the refined binary search presented in [BCP14, NWZ16] to search the "large gap". The search algorithm can find all (polynomially-many) large gap points from an *exponentially large* interval in a pre-defined polynomial time, as long as $|p_{ind_1} - p_{ind_2}|$ is negligible for all (adaptively chosen) interval $[ind_1, ind_2) \subseteq [0, N + 1]$ not containing a large gap point. In this way, we can set the message space to be $[1, N]$ for an exponentially large N.

Towards integrating our new encoding scheme with WM_0**.** Next, we integrate our encoding scheme with the remaining part of WM_0. First, we will specify how to label punctured points with indices. Then, we will show how to

achieve indistinguishability properties required by our robust message encoding scheme. More precisely, we will argue that for a collusion attacker, who can locate some punctured points via comparing outputs of watermarked circuits embedded with different messages, both the unlocated punctured points and labels of the located punctured points are hidden.

Labeling punctured points with indices. Recall that in WM_0, the domain of the PRF is the ciphertext space of a PKE scheme and punctured points are encryptions of plaintexts in a set $\mathcal{V}$. To label a punctured point with an index ind, we append ind to the underlying plaintext, i.e., we define $\mathcal{X}_{ind} = \{\mathtt{Enc}(pk, \mu \| ind)\}_{\mu \in \mathcal{V}}$, where pk is the public key of the underlying PKE scheme and serves as WM_0's extraction key.

Hiding punctured points and labels. Next, we explore how to hide unlocated punctured points and labels of located punctured points. For simplicity, we consider an adversary who gets two watermarked circuits $\mathtt{C}_1$ and $\mathtt{C}_2$ for the same PRF key k, which are embedded with messages msg_1 and msg_2 respectively, where $msg_1 < msg_2$. Recall that our message encoding scheme is able to recover the embedded messages if the following two properties are guaranteed:

- Pseudorandomness of punctured points in $\mathcal{X}_{ind}$ for an adaptively chosen $ind \notin (msg_1, msg_2]$.
- Indistinguishability between punctured points in $\mathcal{X}_{ind_1}$ and $\mathcal{X}_{ind_2}$ for adaptively chosen $ind_1, ind_2 \in (msg_1, msg_2]$.

Unfortunately, the PKE scheme employed in WM_0, which is a puncturable encryption scheme, does not provide the desired properties. To see this, consider an input x from $\mathcal{X}_{ind}$, where $ind \in (msg_1, msg_2]$. Since $\mathtt{C}_1(x) \neq \mathtt{C}_2(x)$, a secret key that can decrypt x must be included in both $\mathtt{C}_1$ and $\mathtt{C}_2$ (otherwise, the circuit cannot recognize it and deal with it properly). However, the puncturable encryption scheme cannot guarantee indistinguishability on ciphertexts that are decryptable.

To bridge the gap, we present a new cryptographic primitive that we call *puncturable functional encryption* and replace puncturable encryption used in WM_0 with it. Roughly speaking, a puncturable functional encryption scheme enhances a functional encryption scheme with the puncturing capability and enjoys both security of functional encryption schemes and that of puncturable encryption schemes. Especially, similar to a functional encryption, it has the "*adaptive indistinguishability*" property, which could ensure indistinguishability of ciphertexts as long as no secret key distinguishing them is provided. Also, similar to a puncturable encryption, it has the "*ciphertext pseudorandomness*" property, which could ensure pseudorandomness of a ciphertext given a secret key punctured on it.

Now, for two punctured points x_1 and x_2 in $\mathcal{X}_{ind_1}$ and $\mathcal{X}_{ind_2}$ respectively, where $ind_1, ind_2 \in (msg_1, msg_2]$, since none of them will be reprogrammed in $\mathtt{C}_1$ while both of them will be reprogrammed in $\mathtt{C}_2$, secret keys hardwired in $\mathtt{C}_1$ and $\mathtt{C}_2$ do not need to distinguish them. Thus, their indistinguishability comes

from the adaptive indistinguishability of the puncturable functional encryption scheme directly. Meanwhile, for a punctured points x in $\mathcal{X}_{ind}$ for $ind \notin (msg_1, msg_2]$, since $\mathtt{C}_1(x) = \mathtt{C}_2(x)$, we can regard $\mathtt{C}_1$ and $\mathtt{C}_2$ as the same circuit when considering pseudorandomness of x. Thus, the pseudorandomness of x can be reduced to the ciphertext pseudorandomness of the puncturable functional encryption scheme, just as what has been argued in the original security proof (in the non-collusion setting) for WM_0.

Constructing puncturable functional encryption. To construct a puncturable functional encryption scheme, we employ a functional encryption scheme, a puncturable encryption scheme, and a statistical sound non-interactive zero-knowledge (NIZK) proof. We integrate them via a "two-layer encryption" approach.

More precisely, a plaintext is first encrypted into an inner ciphertext using the functional encryption scheme. Then the NIZK proof is employed to prove that the inner ciphertext is properly encrypted. Finally, both the inner ciphertext and the proof is encrypted into an outer ciphertext under the puncturable encryption scheme. When decrypting a ciphertext, the decryption algorithm first decrypts the outer ciphertext. It aborts if the proof is invalid and outputs the decryption of the inner ciphertext otherwise. Main security properties of the constructed puncturable functional encryption (namely, adaptively indistinguishability and ciphertext pseudorandomness) reduce to corresponding security properties of underlying functional encryption and puncturable encryption respectively.

3 Notations

Let a be a string, we use $\|a\|$ to denote the length of a. Let $\mathcal{S}$ be a finite set, we use $\|\mathcal{S}\|$ to denote the size of $\mathcal{S}$, and use $s \xleftarrow{\$} \mathcal{S}$ to denote sampling an element s uniformly from set $\mathcal{S}$. For a string a and a set $\mathcal{S}$ of strings, we use $a\|\mathcal{S}$ to denote the set $\{x : \exists s \in \mathcal{S}, x = a\|s\}$. For n elements $e_1, \ldots, e_n$, we use $\{e_1, \ldots, e_n\}$ to denote a set containing these elements and use $(e_1, \ldots, e_n)$ to denote an ordered list of these elements. We write $negl(\cdot)$ to denote a negligible function, and write $poly(\cdot)$ to denote a polynomial. For integers $a \leq b$, we write $[a, b]$ to denote all integers from a to b. For two circuits $\mathtt{C}_1$ and $\mathtt{C}_2$, we write $\mathtt{C}_1 \equiv \mathtt{C}_2$ to denote that for any input x, $\mathtt{C}_1(x) = \mathtt{C}_2(x)$. 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 $c_1, \ldots, c_m$.

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

Lemma 3.1 (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*

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

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

Besides, we also employ some cryptographic primitives and their definitions are recalled in Appendix A.

4 Definition of Collusion Resistant Watermarkable PRF

In this section, we give the formal definition of the collusion resistant watermarkable PRF, which is adapted from definitions of watermarkable PRF in previous works [CHN+16,BLW17,KW17]. The main difference between our definition and previous ones is that the unremovability holds against an adversary that can obtain polynomially-many (instead of one) watermarked circuits for the same PRF key from the challenge oracle. Besides, the extraction algorithm takes an additional parameter q, which can be roughly viewed as the number of colluders, as input. The correctness and the unforgeability hold for arbitrary positive integer q; for the unremovability, a large enough q is needed. In particular, if q is larger than the number of colluders, the extraction algorithm can extract a non-empty subset of the coalition, while using a smaller q may lead to an error symbol.

Remark 4.1. Our definition of collusion resistant unremovability implicitly requires that the adversary is only allowed to obtain a bounded number (i.e., q) of watermarked circuits from the challenge oracle. Thus, it falls into the category of "bounded collusion resistance". Nonetheless, in our definition, the bound q does not need to be fixed in the setup phase and may be varied in different extraction procedures. In fact, if the extractor has a way to know the number of colluders in advance, the scheme remains secure against an arbitrary number of colluders. Besides, since the extraction algorithm is able to detect if a smaller q is used, in practice, the extractor can re-execute the extraction algorithm with a larger q after receiving an error message from the extraction algorithm.

Definition 4.1 (Watermarkable PRFs [CHN+16, BLW17, KW17, adapted]). *Let* $\mathsf{PRF} = (\mathsf{PRF}.\mathtt{KeyGen}, \mathsf{PRF}.\mathtt{Eval})$ *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 mark key* MK *and the extraction key* EK*.*
- ***Mark.*** *On input the mark key* MK*, a secret key* $k \in \mathcal{K}$ *of* PRF*, and a message* $msg \in \mathcal{M}$*, the marking algorithm outputs a marked circuit* $\mathtt{C}$*.*
- ***Extract.*** *On input the extraction key* EK*, a circuit* $\mathtt{C}$*, and a parameter* q*, the extraction algorithm outputs either a set* $\mathcal{L} \subseteq \mathcal{M}$ *or a symbol* UNMARKED *or an error symbol* $\perp$*.*

Definition 4.2 (Watermarking Correctness). *Correctness of the watermarking scheme requires that for any* $k \in \mathcal{K}$*,* $msg \in \mathcal{M}$*, and any polynomial* $q \geq 1$*, let* $(MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$*,* $\mathtt{C} \leftarrow \mathtt{Mark}(MK, k, msg)$*, we have:*

- ***Functionality Preserving.*** $\mathtt{C}(\cdot)$ *and* $\mathsf{PRF}.\mathtt{Eval}(k,\cdot)$ *compute identically on all but a negligible fraction of inputs.*
- ***Extraction Correctness.*** $\Pr[\mathtt{Extract}(EK,\mathtt{C},q) \neq \{msg\}] \leq negl(\lambda)$.

Before presenting the security definition of collusion resistant watermarkable PRF, we first introduce oracles the adversaries can query during the security experiments. Here, the marking oracle is identical to the one defined in previous works, while we redefine the challenge oracle to capture the scenario that the adversary can obtain multiple circuits embedded with different messages for the same secret key.

- **Marking Oracle** $\mathcal{O}^{M}_{MK}(\cdot,\cdot)$**.** On input a message $msg \in \mathcal{M}$ and a PRF key $k \in \mathcal{K}$, the oracle returns a circuit $\mathtt{C} \leftarrow \mathtt{Mark}(MK,k,msg)$.
- **Challenge Oracle** $\mathcal{O}^{C}_{MK}(\cdot)$**.** On input a polynomial-size set M of messages from $\mathcal{M}$, the oracle first samples a key $k^* \leftarrow \mathsf{PRF}.\mathtt{KeyGen}(1^\lambda)$. Then, for each $msg^*_i \in \mathsf{M}$, it computes $\mathtt{C}^*_i \leftarrow \mathtt{Mark}(MK,k^*,msg^*_i)$. Finally, it returns the set $\{\mathtt{C}^*_1,\ldots,\mathtt{C}^*_Q\}$, where $Q = \|\mathsf{M}\|$.

Definition 4.3 (Collusion Resistant Unremovability). *The watermarking scheme for a PRF is collusion resistant unremovable if for any polynomial q, for all polynomial-time (PPT) and unremoving-admissible adversaries $\mathcal{A}$, we have* $\Pr[\mathtt{ExptUR}_{\mathcal{A},q}(\lambda)=1] \leq negl(\lambda)$*, where we define the experiment* ExptUR *and unremoving-admissibility as follows:*

1. *The challenger samples* $(MK,EK) \leftarrow \mathtt{Setup}(1^\lambda)$ *and returns* EK *to* $\mathcal{A}$.
2. *Then,* $\mathcal{A}$ *is allowed to make multiple queries to the marking oracle.*
3. *Next,* $\mathcal{A}$ *submits a set* M^* *of* Q *messages in* $\mathcal{M}$ *to the challenge oracle and gets a set* C^* *of circuits back.*
4. *Then,* $\mathcal{A}$ *is further allowed to make multiple queries to the marking oracle.*
5. *Finally* $\mathcal{A}$ *submits a circuit* $\tilde{\mathtt{C}}$*. The experiment outputs 0 if*
   (a) $q < Q$ *and either* $\mathtt{Extract}(EK,\tilde{\mathtt{C}},q)$ *is a non-empty subset of* M^* *or it equals to the error symbol* $\perp$.
   (b) $q \geq Q$ *and* $\mathtt{Extract}(EK,\tilde{\mathtt{C}},q)$ *is a non-empty subset of* M^*.
   Otherwise, the experiment outputs 1.

Here, an adversary $\mathcal{A}$ *is unremoving-admissible if there exists circuit* $\mathtt{C}^*_i \in \mathsf{C}^*$ *that* $\mathtt{C}^*_i$ *and* $\tilde{\mathtt{C}}$ *compute identically on all but a negligible fraction of inputs.*

Definition 4.4 (δ-Unforgeability). *The watermarking scheme for a PRF is δ-unforgeable if for any polynomial $q \geq 1$ and for all PPT and δ-unforging-admissible adversaries $\mathcal{A}$, we have* $\Pr[\mathtt{ExptUF}_{\mathcal{A},q}(\lambda)=1] \leq negl(\lambda)$*, where we define the experiment* ExptUF *and unforging-admissiability as follows:*

1. *The challenger samples* $(MK,EK) \leftarrow \mathtt{Setup}(1^\lambda)$ *and returns* EK *to* $\mathcal{A}$.
2. *Then,* $\mathcal{A}$ *is allowed to make multiple queries to the marking oracle.*
3. *Finally,* $\mathcal{A}$ *submits a circuit* $\tilde{\mathtt{C}}$*. The experiment outputs 0 if* $\mathtt{Extract}(EK,\tilde{\mathtt{C}},q) = \mathsf{UNMARKED}$*; otherwise, the experiment output 1.*

Here, let Q' *be the number of queries* $\mathcal{A}$ *made to the marking oracle, then an adversary* $\mathcal{A}$ *is δ-unforging-admissible if for all* $i \in [1,Q']$*, its submitted circuit* $\tilde{\mathtt{C}}$ *and the circuit* $\mathtt{C}_i$ *compute differently on at least a δ fraction of inputs, where* $\mathtt{C}_i$ *is the output of the marking oracle on the ith query.*

5 Puncturable Functional Encryption

In this section, we define puncturable functional encryption and give a concrete construction. A puncturable functional encryption scheme can achieve functionalities and security of both puncturable encryption schemes and functional encryption schemes. Besides, as we will use the puncturable functional encryption scheme together with an indistinguishability obfuscator, we also require it to have an "iO-compatible correctness", which demands a decryption consistency for different secret keys. More precisely, when using two secret keys sk_1, sk_2 for functions f_1, f_2 respectively, for any string ct in the ciphertext space, either both secret keys will fail in decrypting it or there exists a plaintext μ in the plaintext space that decrypting ct under sk_1 and sk_2 will lead to $f_1(\mu)$ and $f_2(\mu)$ respectively.

5.1 Definition of Puncturable Functional Encryption

Definition 5.1 (Puncturable Functional Encryption). *A puncturable functional encryption scheme for a family of function $\mathcal{F}$[8] with plaintext space $\{0,1\}^m$ and ciphertext space $\{0,1\}^n$ consists of five algorithms:*

- ***Setup.*** *On input the security parameter λ, the setup algorithm outputs the master public key/master secret key pair (mpk, msk).*
- ***KeyGen.*** *On input the master secret key msk and a function $f \in \mathcal{F}$, the key generation algorithm outputs a secret key sk for f.*
- ***Enc.*** *On input the master public key mpk and a message $msg \in \{0,1\}^m$, the encryption algorithm outputs the ciphertext ct.*
- ***Dec.*** *On input a secret key sk and a ciphertext $ct \in \{0,1\}^n$, the decryption algorithm outputs a string msg or a decryption failure symbol $\perp$.*
- ***Puncture.*** *On input a secret key sk and two ciphertexts ct_1, ct_2, the puncture algorithm outputs a punctured secret key sk'.*

Next, we describe properties of puncturable functional encryption schemes.

Definition 5.2 (Properties of Puncturable Functional Encryption). *A puncturable functional encryption scheme* $\mathsf{PFE} = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Puncture})$ *with plaintext space $\{0,1\}^m$, ciphertext space $\{0,1\}^n$ and supported function family $\mathcal{F}$ is required to have the following properties.*

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

[8] In this work, we concentrate on schemes supporting function family $\mathcal{F}$ of polynomial-size circuit with output space $\{0,1\}^m$.

- ***Punctured Correctness.*** *For any $f \in \mathcal{F}$, any strings $ct_0, ct_1 \in \{0,1\}^n$ and any* unbounded *adversary $\mathcal{A}$, we have*

$$\Pr\left[\begin{array}{l}(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda);\\ sk \leftarrow \mathtt{KeyGen}(msk, f);\\ sk' \leftarrow \mathtt{Puncture}(sk, \{ct_0, ct_1\});\\ ct \leftarrow \mathcal{A}(mpk, msk, sk, sk');\end{array} : \begin{array}{r} ct \notin \{ct_0, ct_1\} \wedge \\ \mathtt{Dec}(sk, ct) \neq \mathtt{Dec}(sk', ct)\end{array}\right] \leq negl(\lambda)$$

- ***iO-Compatible Correctness.*** *For each master public key/master secret key pair (mpk, msk), the ciphertext space can be divided into two disjoint parts, namely, the valid ciphertext set $\mathcal{V}_{(mpk,msk)}$ and the invalid ciphertext set $\mathcal{I}_{(mpk,msk)}$, which satisfy $\mathcal{V}_{(mpk,msk)} \cup \mathcal{I}_{(mpk,msk)} = \{0,1\}^n$ and $\mathcal{V}_{(mpk,msk)} \cap \mathcal{I}_{(mpk,msk)} = \emptyset$. The iO-compatible correctness requires that:*

  1. *For any $f \in \mathcal{F}$ and any* unbounded *adversary $\mathcal{A}$, we have:*

$$\Pr\left[\begin{array}{l}(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda);\\ sk \leftarrow \mathtt{KeyGen}(msk, f);\\ ct \leftarrow \mathcal{A}(mpk, msk, sk);\end{array} : \begin{array}{c} ct \in \mathcal{I}_{(mpk,msk)} \wedge \\ \mathtt{Dec}(sk, ct) \neq \perp\end{array}\right] \leq negl(\lambda)$$

  2. *For any $f_1, f_2 \in \mathcal{F}$ and any* unbounded *adversary $\mathcal{A}$, we have:*

$$\Pr\left[\begin{array}{l}(mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda);\\ sk_1 \leftarrow \mathtt{KeyGen}(msk, f_1);\\ sk_2 \leftarrow \mathtt{KeyGen}(msk, f_2);\\ ct \leftarrow \mathcal{A}(mpk, msk, sk_1, sk_2);\end{array} : \begin{array}{l} ct \in \mathcal{V}_{(mpk,msk)} \wedge \\ (\forall msg \in \{0,1\}^m, \\ \mathtt{Dec}(sk_1, ct) \neq f_1(msg) \vee \\ \mathtt{Dec}(sk_2, ct) \neq f_2(msg))\end{array}\right] \leq negl(\lambda)$$

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

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

  where $\mathcal{O}_{msk}$ takes as input a function $f \in \mathcal{F}$ and outputs a secret key $sk \leftarrow \mathtt{KeyGen}(msk, f)$; for all f submitted to the oracle $\mathcal{O}_{msk}$, $f(msg_0) = f(msg_1)$; and the $\mathcal{O}_{msk}$ can only be queried two times. Note that in our security proof for collusion resistant watermarkable PRF, we only require a two-key security, thus we just define this type of adaptive indistinguishability here.

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

$$\Pr\left[\begin{array}{l}(st, msg, f) \leftarrow \mathcal{A}_1(1^\lambda);\\ (mpk, msk) \leftarrow \mathtt{Setup}(1^\lambda);\\ sk \leftarrow \mathtt{KeyGen}(msk, f);\\ ct_0 \leftarrow \mathtt{Enc}(mpk, msg);\\ ct_1 \xleftarrow{\$} \{0,1\}^n;\\ sk' \leftarrow \mathtt{Puncture}(sk, \{ct_0, ct_1\});\\ b \leftarrow \{0,1\};\\ b' \leftarrow \mathcal{A}_2(st, mpk, sk', ct_b, ct_{1-b});\end{array} : b = b'\right] \leq 1/2 + negl(\lambda)$$

5.2 Construction of Puncturable Functional Encryption

In this section, we present our construction of puncturable functional encryption.

Let λ be the security parameter. Let n, m, l, n' be positive integers that are polynomial in λ. Our construction is based on the following three building blocks:

- A functional encryption scheme $\mathsf{FE} = (\mathsf{FE}.\mathtt{Setup}, \mathsf{FE}.\mathtt{KeyGen}, \mathsf{FE}.\mathtt{Enc}, \mathsf{FE}.\mathtt{Dec})$ with plaintext space $\{0,1\}^m$, ciphertext space $\{0,1\}^n$ and encryption randomness space $\mathcal{R}$. Also, we require that it supports a family $\mathcal{F}$ of polynomial-size circuit with output space $\{0,1\}^m$.
- A statistically sound NIZK proof system $\mathsf{NIZK} = (\mathsf{NIZK}.\mathtt{KeyGen}, \mathsf{NIZK}.\mathtt{Prove}, \mathsf{NIZK}.\mathtt{Verify})$ for $\mathcal{L}$, where

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

  and require that the proof size is n' when proving a statement in $\mathcal{L}$.
- A puncturable encryption scheme $\mathsf{PE} = (\mathsf{PE}.\mathtt{KeyGen}, \mathsf{PE}.\mathtt{Puncture}, \mathsf{PE}.\mathtt{Enc}, \mathsf{PE}.\mathtt{Dec})$ with plaintext space $\{0,1\}^{n+n'}$ and ciphertext space $\{0,1\}^l$.

We construct $\mathsf{PFE} = (\mathsf{PFE}.\mathtt{Setup}, \mathsf{PFE}.\mathtt{KeyGen}, \mathsf{PFE}.\mathtt{Puncture}, \mathsf{PFE}.\mathtt{Enc}, \mathsf{PFE}.\mathtt{Dec})$ for $\mathcal{F}$, which has a plaintext space $\{0,1\}^m$ and a ciphertext space $\{0,1\}^l$, as follows:

- **Setup.** On input a security parameter λ, the setup algorithm generates $(mpk, msk) \leftarrow \mathsf{FE}.\mathtt{Setup}(1^\lambda)$, $crs \leftarrow \mathsf{NIZK}.\mathtt{KeyGen}(1^\lambda)$, and $(pk, sk) \leftarrow \mathsf{PE}.\mathtt{KeyGen}(1^\lambda)$. Then it outputs the master public key $MPK = (mpk, crs, pk)$ and the master secret key $MSK = (msk, sk, mpk, crs)$ of PFE.
- **KeyGen.** On input a master secret key $MSK = (msk, sk, mpk, crs)$ of PFE and a function $f \in \mathcal{F}$, the key generation algorithm generates $fsk \leftarrow \mathsf{FE}.\mathtt{KeyGen}(msk, f)$ and outputs a secret key $SK = (fsk, sk, mpk, crs)$ of PFE.

- **Enc.** On input a master public key $MPK = (mpk, crs, pk)$ of PFE and a message $msg \in \{0,1\}^m$, the encryption algorithm first samples $r \in \mathcal{R}$. Then, it computes $ct = \mathsf{FE}.\mathtt{Enc}(mpk, msg; r)$, and $\pi \leftarrow \mathsf{NIZK}.\mathtt{Prove}(crs, (mpk, ct), (msg, r))$. Finally, it outputs $CT \leftarrow \mathsf{PE}.\mathtt{Enc}(pk, ct\|\pi)$.
- **Dec.** On input a secret key $SK = (fsk, sk, mpk, crs)$ of PFE and a ciphertext $CT \in \{0,1\}^l$, the decryption algorithm first decrypts CT with the secret key of PE and gets $ct\|\pi \leftarrow \mathsf{PE}.\mathtt{Dec}(sk, CT)$. It aborts and outputs $\perp$ if $ct\|\pi = \perp$ or $\mathsf{NIZK}.\mathtt{Verify}(crs, (mpk, ct), \pi) = 0$. Otherwise, it outputs $\mathsf{FE}.\mathtt{Dec}(fsk, ct)$.
- **Puncture.** On input a secret key $SK = (fsk, sk, mpk, crs)$ of PFE and two ciphertexts $CT_1, CT_2 \in \{0,1\}^l$, the puncture algorithm generates $sk' \leftarrow \mathsf{PE}.\mathtt{Puncture}(sk, \{CT_1, CT_2\})$ and outputs $SK' = (fsk, sk', mpk, crs)$.

Theorem 5.1. *If* FE *is a secure functional encryption for* $\mathcal{F}$ *with perfect correctness and (two-key) adaptive security,* NIZK *is a NIZK proof system with adaptively statistical soundness and adaptive zero-knowledge for language* $\mathcal{L}$*, and* PE *is a secure puncturable encryption scheme, then* PFE *is a secure puncturable functional encryption as defined in Sec. 5.1.*

We give proof of Theorem 5.1 in Appendix B.

6 Construction of Collusion Resistant Watermarkable PRF

In this section, we show how to obtain collusion resistant watermarkable PRF families. In particular, we construct a collusion resistant watermarking scheme for any puncturable PRF with weak key-injectivity and constrained one-wayness.

Let λ be the security parameter. Let δ be a positive real value and $d = \lambda/\delta = poly(\lambda)$. Let n, m, l, κ be positive integers that are polynomial in λ and $n = l + poly(\lambda)$. Let

$$\mathsf{PRF} = (\mathsf{PRF}.\mathtt{KeyGen}, \mathsf{PRF}.\mathtt{Eval}, \mathsf{PRF}.\mathtt{Constrain}, \mathsf{PRF}.\mathtt{ConstrainEval})$$

be a family of puncturable PRF with key space $\mathcal{K}$, input space $\{0,1\}^n$, and output space $\{0,1\}^m$. Our watermarking scheme for PRF is built on the following building blocks.

- A puncturable functional encryption scheme $\mathsf{PFE} = (\mathsf{PFE}.\mathtt{Setup}, \mathsf{PFE}.\mathtt{KeyGen}, \mathsf{PFE}.\mathtt{Puncture}, \mathsf{PFE}.\mathtt{Enc}, \mathsf{PFE}.\mathtt{Dec})$ with plaintext space $\{0,1\}^{(d+1)\cdot l+\kappa}$, ciphertext space $\{0,1\}^n$ and encryption randomness space $\mathcal{R}$. Also, we require that it supports a family $\mathcal{F}$ of polynomial-size circuits with output space $\{0,1\}^{(d+1)\cdot l+\kappa}$.
- A family of prefix puncturable PRF $\mathsf{F} = (\mathsf{F}.\mathtt{KeyGen}, \mathsf{F}.\mathtt{Eval}, \mathsf{F}.\mathtt{Constrain}, \mathsf{F}.\mathtt{ConstrainEval})$ with input space $\{0,1\}^{(d+1)\cdot l}$ and output space $\mathcal{K}$.
- An indistinguishability obfuscator iO for all polynomial-size circuits.
- Two pseudorandom generators $\mathsf{G} : \{0,1\}^l \to \{0,1\}^n$ and $\mathsf{G}' : \{0,1\}^{\frac{l}{2}} \to \{0,1\}^l$.

- A family of collision-resistant hash function $\mathcal{H}$ with input space $\{0,1\}^{d \cdot m}$ and output space $\{0,1\}^{l}$.

We construct $\mathsf{WM} = (\mathsf{WM}.\mathtt{Setup}, \mathsf{WM}.\mathtt{Mark}, \mathsf{WM}.\mathtt{Extract})$, which has a message space $\{0,1\}^{\kappa} \backslash \{0^{\kappa}\} = [1, 2^{\kappa} - 1]$, as follows:

- **Setup.** On input a security parameter λ, the setup algorithm first samples $H \xleftarrow{\$} \mathcal{H}$ and generates $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^{\lambda})$. Then it generates $(mpk, msk) \leftarrow \mathsf{PFE}.\mathtt{Setup}(1^{\lambda})$ and $sk \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, \mathtt{ID})$, where $\mathtt{ID} : \{0,1\}^{(d+1) \cdot l + \kappa} \to \{0,1\}^{(d+1) \cdot l + \kappa}$ is the identity function, i.e., for any $x \in \{0,1\}^{(d+1) \cdot l + \kappa}$, $ID(x) = x$. Next, it computes $\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}[mpk, K])$, where $\mathtt{Ext}$ is defined in Figure 1[9]. Finally, the output of the setup algorithm is (MK, EK) where $MK = (sk, K, H)$ and $EK = (H, \mathtt{E})$.
- **Mark.** On input a mark key $MK = (sk, K, H)$, a secret key $k \in \mathcal{K}$ for PRF and a message msg, the marking algorithm outputs a circuit $\mathtt{C} \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k, msg])$, where $\mathtt{M}$ is defined in Figure 1[10].
- **Extract.** On input an extraction key $EK = (H, \mathtt{E})$, a circuit $\mathtt{C}$, and a parameter q, the extraction algorithm first computes $\epsilon = 1/((\kappa + 1) \cdot q + 1)$, $T = \lambda/\epsilon^2$, and $S = q \cdot (\kappa + 1)$ and sets a variable $counter = 0$. Then it computes $\mathcal{L} = \mathtt{Trace}(0, 2^{\kappa}, 1, 0, \epsilon, T, \mathtt{E}, H, \mathtt{C})$, where $\mathtt{Trace}(\cdot)$ is defined in Figure 1.

  In this procedure, the algorithm also maintains the variable *counter* and increase it by 1 each time the function $\mathtt{Test}(\cdot)$ defined in Figure 1 is invoked. The algorithm aborts and outputs $\perp$ once $counter > S$. In case the algorithm does not abort, it checks the set $\mathcal{L}$ returned by $\mathtt{Trace}$. It outputs $\perp$ if $\mathcal{L} = \emptyset$ and outputs $\mathsf{UNMARKED}$ if $\mathcal{L} = \{0\}$. Otherwise, it outputs $\mathcal{L}$.

Theorem 6.1. *If* PRF *is a secure puncturable PRF with weak key-injectivity and constrained one-wayness,* PFE *is a secure puncturable functional encryption scheme as defined in Sec. 5.1,* F *is a secure prefix puncturable PRF,* G *and* G' *are pseudorandom generators,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* WM *is a secure watermarking scheme with collusion resistant unremovability and* δ*-unforgeability, as defined in Sec. 4, for* PRF.

We present the proof of Theorem 6.1 in Appendix C.

Here, we provide a brief overview on how to prove the collusion resistant unremovability of WM. For simplicity, we consider an adversary who only gets two circuits $\mathtt{C}_1$ and $\mathtt{C}_2$ for the same secret key k embedded with messages msg_1 and msg_2 respectively, where $msg_1 < msg_2$, and omit its advantage in viewing the public key and querying the marking oracle.

[9] The circuit $\mathtt{Ext}$, as well as all circuits $\mathtt{Ext}^{(\cdot)}$ appeared in the security proofs for WM will be padded to the same size.

[10] The circuit $\mathtt{M}$, as well as all circuits $\mathtt{M}^{(\cdot)}$ appeared in the security proof for WM will be padded to the same size.

$\mathtt{Ext}$	$\mathtt{Trace}$
Constant: mpk, K **Input:** $a_1, \ldots, a_d, b, ind, r$ 1. $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$. 2. $x = \mathsf{PFE}.\mathtt{Enc}(mpk, t_1 \| \ldots \| t_d \| b \| ind; r)$. 3. $k' = \mathsf{F}.\mathtt{Eval}(K, t_1 \| \ldots \| t_d \| b)$. 4. $y = \mathsf{PRF}.\mathtt{Eval}(k', x)$. 5. Output (x, y).	**Input:** $ind_1, ind_2, p_1, p_2, \epsilon, T, \mathtt{E}, H, \mathtt{C}$ 1. $\Delta = \lvert p_1 - p_2 \rvert$. 2. If $\Delta \leq \epsilon$: return $\emptyset$. 3. If $ind_2 - ind_1 = 1$: return $\{ind_1\}$. 4. $ind_3 = \lfloor \frac{ind_1 + ind_2}{2} \rfloor$. 5. $p_3 = \mathtt{Test}(ind_3, T, \mathtt{E}, H, \mathtt{C})$. 6. Return $\mathtt{Trace}(ind_1, ind_3, p_1, p_3, \epsilon, T, \mathtt{E}, H, \mathtt{C}) \cup \mathtt{Trace}(ind_3, ind_2, p_3, p_2, \epsilon, T, \mathtt{E}, H, \mathtt{C})$.
$\mathtt{M}$	$\mathtt{Test}$
Constant: sk, K, H, k, msg **Input:** x 1. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$. 2. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_d))) = b)$ (a) $k' = \mathsf{F}.\mathtt{Eval}(K, t_1 \| \ldots \| t_d \| b)$. (b) $y = \mathsf{PRF}.\mathtt{Eval}(k', x)$. (c) Output y. 3. Otherwise, output $\mathsf{PRF}.\mathtt{Eval}(k, x)$.	**Input:** $ind, T, \mathtt{E}, H, \mathtt{C}$ 1. $Acc = 0$ 2. For $i \in [1, T]$: (a) Sample $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$ and $r \xleftarrow{\$} \mathcal{R}$. (b) $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$. (c) $b = H(\mathtt{C}(\mathsf{G}(t_1)), \ldots, \mathtt{C}(\mathsf{G}(t_d)))$. (d) $(x, y) = \mathtt{E}(a_1, \ldots, a_d, b, ind, r)$. (e) If $\mathtt{C}(x) = y$: $Acc = Acc + 1$. 3. Return $\frac{Acc}{T}$.

Fig. 1 The circuit $\mathtt{Ext}$, the circuit $\mathtt{M}$, the function $\mathtt{Trace}$, and the function $\mathtt{Test}$

Following the syntax used in Sec. 2, we denote an input encrypted from $t_1 \| \ldots \| t_d \| b \| ind$ satisfying $b = H(\mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_d)))$ as a punctured point labeled with an index ind. Also, we use $\mathcal{X}_{ind}$ to denote the set of all punctured points labeled with the index ind.

First, as shown in [BCP14, NWZ16], the Trace algorithm can output a non-empty subset of $\{msg_1, msg_2\}$ if the adversary cannot distinguish 1) two punctured points labeled with different indices adaptively chosen from $(msg_1, msg_2]$ and 2) a punctured point labeled with an index adaptively chosen outside $(msg_1, msg_2]$ and a random point.

For two punctured points in $\mathcal{X}_{ind_1}$ and $\mathcal{X}_{ind_2}$ respectively, where $ind_1, ind_2 \in (msg_1, msg_2]$, both of them are properly punctured and reprogrammed in $\mathtt{C}_2$ while none of them are punctured in $\mathtt{C}_1$, thus the decryption (in both $\mathtt{C}_1$ and $\mathtt{C}_2$) do not need to distinguish them. So, their indistinguishability comes from the adaptive indistinguishability of PFE.

The adaptive indistinguishability of PFE also implies indistinguishability of two punctured points in $\mathcal{X}_{ind_1}$ and $\mathcal{X}_{ind_2}$ when both ind_1 and ind_2 are in $[1, msg_1]$ or both of them are in $(msg_2, 2^\kappa - 1]$. This could reduce the problem of claiming the pseudorandomness of a punctured point labeled with an index adaptively chosen from $[1, msg_1]$ (or $(msg_2, 2^\kappa - 1]$) to the problem of claiming the pseudorandomness of a punctured points from $\mathcal{X}_1$ (resp. $\mathcal{X}_{2^\kappa - 1}$), where the latter claim can be implied by the ciphertext pseudorandomness of PFE. In this way, pseudorandomness of punctured points in $\mathcal{X}_{ind}$ for $ind \notin (msg_1, msg_2]$ is proved.

It is worth noting that when arguing indistinguishability between a punctured point from $\mathcal{X}_1$ and a random input, we also need to show that the marked circuits are able to hide punctured points that are punctured and identically reprogrammed in all circuits. This indicates that our construction of watermarkable PRF involves a collusion resistant constraint-hiding constrained PRF implicitly.

7 Collusion Resistant Watermarking Schemes for Other Cryptographic Functionalities

In this section, we show how to construct watermarking schemes for advanced cryptographic functionalities, including the decryption algorithm of a PKE scheme and the signing algorithm of a signature scheme. The constructions are based on the observation that the PKE scheme (and the signature scheme) constructed in [SW14] has a decryption algorithm (resp. signing algorithm) that is nothing more than a puncturable PRF evaluation. The observation 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 give a detailed description for how to construct collusion resistant watermarkable PKE schemes and omit the construction for collusion resistant watermarkable signature schemes. We start by presenting the formal definition of watermarkable PKE scheme. Then we give our construction based on a puncturable PRF, an indistinguishability obfuscator, a puncturable functional encryption scheme, and some standard cryptographic primitives.

7.1 The Definition

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

Definition 7.1 (Watermarkable PKEs [CHN+16, adapted]). *Let* $\mathsf{PKE} = (\mathsf{PKE}.\mathtt{KeyGen}, \mathsf{PKE}.\mathtt{Enc}, \mathsf{PKE}.\mathtt{Dec})$ *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 mark key* MK *and the extraction key* EK*.*
- ***Mark.*** *On input the mark key* MK*, a secret key* $sk \in \mathcal{SK}$ *of* PKE*, and a message* $msg \in \mathcal{M}$*, the marking algorithm outputs a marked circuit* $\mathtt{C}$*.*
- ***Extract.*** *On input the extraction key* EK*, a circuit* $\mathtt{C}$*, and a parameter* q*, the extraction algorithm outputs either a set* $\mathcal{L} \subseteq \mathcal{M}$ *or a symbol* $\mathsf{UNMARKED}$ *or an error symbol* $\perp$*.*

Definition 7.2 (Watermarking Correctness). *Correctness of the watermarking scheme requires that for any* $sk \in \mathcal{SK}$*,* $msg \in \mathcal{M}$*, and any polynomial* $q \geq 1$*, let* $(MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$*,* $\mathtt{C} \leftarrow \mathtt{Mark}(MK, sk, msg)$*, we have:*

- ***Functionality Preserving.*** $\mathtt{C}(\cdot)$ *and* $\mathsf{PKE}.\mathtt{Dec}(sk, \cdot)$ *compute identically on all but a negligible fraction of inputs.*
- ***Extraction Correctness.*** $\Pr[\mathtt{Extract}(EK, \mathtt{C}, q) \neq \{msg\}] \leq negl(\lambda)$.

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

- **Marking Oracle $\mathcal{O}^{M}_{MK}(\cdot, \cdot)$.** On input a message $msg \in \mathcal{M}$ and a secret key key $sk \in \mathcal{SK}$, the oracle returns a circuit $\mathtt{C} \leftarrow \mathtt{Mark}(MK, sk, msg)$.
- **Challenge Oracle $\mathcal{O}^{C}_{MK}(\cdot)$.** On input a polynomial-size set M of messages from $\mathcal{M}$, the oracle first generates a key pair $(sk^*, pk^*) \leftarrow \mathsf{PKE}.\mathtt{KeyGen}(1^\lambda)$. Then, for each $msg^*_i \in \mathsf{M}$, it computes $\mathtt{C}^*_i \leftarrow \mathtt{Mark}(MK, sk^*, msg^*_i)$. Finally, it returns the set $\{\mathtt{C}^*_1, \ldots, \mathtt{C}^*_Q\}$, where $Q = \|\mathsf{M}\|$, and the public key pk^*.

Definition 7.3 (Collusion Resistant Unremovability). *The watermarking scheme for a PKE is collusion resistant unremovable if for any polynomial q, for all PPT and unremoving-admissible adversaries $\mathcal{A}$, we have* $\Pr[\mathtt{ExptUR}_{\mathcal{A},q}(\lambda) = 1] \leq negl(\lambda)$*, where we define the experiment* ExptUR *and unremoving-admissibility as follows:*

1. *The challenger samples $(MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$ and returns EK to $\mathcal{A}$.*
2. *Then, $\mathcal{A}$ is allowed to make multiple queries to the marking oracle.*
3. *Next, $\mathcal{A}$ submits a set M^* of Q messages in $\mathcal{M}$ to the challenge oracle and gets a set C^* of circuits as well as a public key pk^* back.*
4. *Then, $\mathcal{A}$ is further allowed to make multiple queries to the marking oracle.*
5. *Finally $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$. The experiment outputs 0 if*
   (a) $q < Q$ and either $\mathtt{Extract}(EK, \tilde{\mathtt{C}}, q)$ is a non-empty subset of M^* or it equals to the error symbol $\perp$.
   (b) $q \geq Q$ and $\mathtt{Extract}(EK, \tilde{\mathtt{C}}, q)$ is a non-empty subset of M^*.
   Otherwise, the experiment outputs 1.

Here, an adversary $\mathcal{A}$ is unremoving-admissible if there exists circuit $\mathtt{C}^*_i \in \mathsf{C}^*$ that $\mathtt{C}^*_i$ and $\tilde{\mathtt{C}}$ compute identically on all but a negligible fraction of inputs.

Definition 7.4 (δ-Unforgeability). *The watermarking scheme for a PKE is δ-unforgeable if for any polynomial $q \geq 1$ and for all PPT and δ-unforging-admissible adversaries $\mathcal{A}$, we have* $\Pr[\mathtt{ExptUF}_{\mathcal{A},q}(\lambda) = 1] \leq negl(\lambda)$*, where we define the experiment* ExptUF *and unforging-admissiability as follows:*

1. *The challenger samples $(MK, EK) \leftarrow \mathtt{Setup}(1^\lambda)$ and returns EK to $\mathcal{A}$.*
2. *Then, $\mathcal{A}$ is allowed to make multiple queries to the marking oracle.*
3. *Finally, $\mathcal{A}$ submits a circuit $\tilde{\mathtt{C}}$. The experiment outputs 0 if $\mathtt{Extract}(EK, \tilde{\mathtt{C}}, q) = \mathsf{UNMARKED}$; otherwise, the experiment output 1.*

Here, let Q' be the number of queries $\mathcal{A}$ made to the marking oracle, then an adversary $\mathcal{A}$ is δ-unforging-admissible if for all $i \in [1, Q']$, its submitted circuit $\tilde{\mathtt{C}}$ and the circuit $\mathtt{C}_i$ compute differently on at least a δ fraction of inputs, where $\mathtt{C}_i$ is the output of the marking oracle on the ith query.

7.2 The Construction

Let λ be the security parameter. Let δ be a positive real value and $d = \lambda/\delta = poly(\lambda)$. Let n, m, l, κ be positive integers that are polynomial in λ and $n = l + poly(\lambda)$. Our watermarkable PKE scheme is built from the following building blocks:

- A family of puncturable PRF $\mathsf{PRF} = (\mathsf{PRF}.\mathtt{KeyGen}, \mathsf{PRF}.\mathtt{Eval}, \mathsf{PRF}.\mathtt{Constrain}, \mathsf{PRF}.\mathtt{ConstrainEval})$ with key space $\mathcal{K}$, input space $\{0,1\}^n$, and output space $\{0,1\}^m$.
- A puncturable functional encryption scheme $\mathsf{PFE} = (\mathsf{PFE}.\mathtt{Setup}, \mathsf{PFE}.\mathtt{KeyGen}, \mathsf{PFE}.\mathtt{Puncture}, \mathsf{PFE}.\mathtt{Enc}, \mathsf{PFE}.\mathtt{Dec})$ with plaintext space $\{0,1\}^{(d+1)\cdot l+\kappa}$, ciphertext space $\{0,1\}^n$ and encryption randomness space $\mathcal{R}$. Also, we require that it supports a family $\mathcal{F}$ of polynomial-size circuit with output space $\{0,1\}^{(d+1)\cdot l+\kappa}$.
- A family of prefix puncturable PRF $\mathsf{F} = (\mathsf{F}.\mathtt{KeyGen}, \mathsf{F}.\mathtt{Eval}, \mathsf{F}.\mathtt{Constrain}, \mathsf{F}.\mathtt{ConstrainEval})$ with input space $\{0,1\}^{(d+1)\cdot l}$ and output space $\mathcal{K}$.
- An indistinguishability obfuscator iO for all polynomial-size circuits.
- Three pseudorandom generators $\mathsf{G} : \{0,1\}^l \to \{0,1\}^n$, $\mathsf{G}' : \{0,1\}^{\frac{l}{2}} \to \{0,1\}^l$, and $\tilde{\mathsf{G}} : \{0,1\}^\lambda \to \{0,1\}^n$.
- A family of collision-resistant hash function $\mathcal{H}$ with input space $\{0,1\}^{d\cdot m}$ and output space $\{0,1\}^l$.

For completeness, we first recall how PKE scheme PKE is constructed in [SW14].

- **KeyGen.** On input a security parameter λ, the key generation algorithm first samples $k \xleftarrow{\$} \mathcal{K}$. Then, it computes $\mathtt{P} \leftarrow \mathsf{iO}(\mathtt{Encrypt}[k])$, where $\mathtt{Encrypt}$ is defined in Figure 2 and is properly padded. Finally, the output of the key generation algorithm is (pk, sk) where $pk = \mathtt{P}$ and $sk = k$.
- **Enc.** On input a public key $pk = \mathtt{P}$ and a message $msg \in \{0,1\}^m$, the encryption algorithm samples $r \xleftarrow{\$} \{0,1\}^\lambda$ and outputs $\mathtt{P}(msg, r)$.
- **Dec.** On input a secret key $sk = k$ and a ciphertext $ct = (x, z)$, the decryption algorithm outputs $msg = \mathsf{PRF}.\mathtt{Eval}(k, x) \oplus z$.

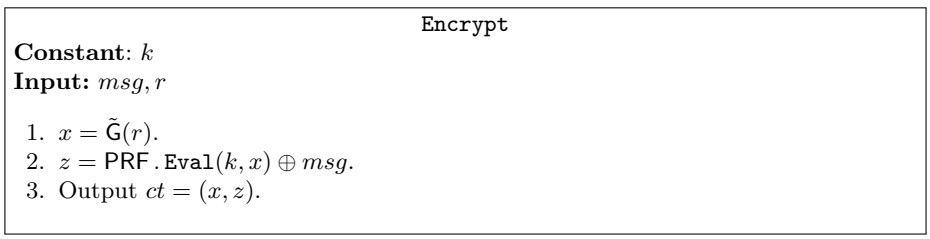

Fig. 2 The circuit $\mathtt{Encrypt}$.

Next, we construct the watermarking scheme $\mathsf{WM} = (\mathsf{WM}.\mathtt{Setup}, \mathsf{WM}.\mathtt{Mark}, \mathsf{WM}.\mathtt{Extract})$ for the above constructed PKE scheme, which has a message space $\{0,1\}^\kappa \backslash \{0^\kappa\} = [1, 2^\kappa - 1]$, as follows:

- **Setup.** On input a security parameter λ, the setup algorithm first samples $H \xleftarrow{\$} \mathcal{H}$ and generates $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then it generates $(mpk, msk) \leftarrow \mathsf{PFE}.\mathtt{Setup}(1^\lambda)$ and $sk \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, \mathtt{ID})$, where $\mathtt{ID} : \{0,1\}^{(d+1)\cdot l+\kappa} \to \{0,1\}^{(d+1)\cdot l+\kappa}$ is the identity function, i.e., for any $x \in \{0,1\}^{(d+1)\cdot l+\kappa}$, $ID(x) = x$. Next, it computes $\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}[mpk, K])$, where $\mathtt{Ext}$ is defined in Figure 3 and is properly padded. Finally, the output of the setup algorithm is (MK, EK) where $MK = (sk, K, H)$ and $EK = (H, \mathtt{E})$.
- **Mark.** On input a mark key $MK = (sk, K, H)$, a secret key $k \in \mathcal{K}$ for PKE and a message msg, the marking algorithm outputs a circuit $\mathtt{C} \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k, msg])$, where $\mathtt{M}$ is defined in Figure 3 and is properly padded.
- **Extract.** On input an extraction key $EK = (H, \mathtt{E})$, a circuit $\mathtt{C}$, and a parameter q, the extraction algorithm first computes $\epsilon = 1/((\kappa+1)\cdot q + 1)$, $T = \lambda/\epsilon^2$, and $S = q \cdot (\kappa + 1)$ and sets a variable $counter = 0$. Then it computes $\mathcal{L} = \mathtt{Trace}(0, 2^\kappa, 1, 0, \epsilon, T, \mathtt{E}, H, \mathtt{C})$, where $\mathtt{Trace}(\cdot)$ is defined in Figure 3.
  In this procedure, the algorithm also maintains the variable *counter* and increase it by 1 each time the function $\mathtt{Test}(\cdot)$ defined in Figure 3 is invoked. The algorithm aborts and outputs $\perp$ once *counter* exceeds S. In case the algorithm does not abort, it checks the set $\mathcal{L}$ returned by $\mathtt{Trace}$. It outputs $\perp$ if $\mathcal{L} = \emptyset$ and outputs $\mathsf{UNMARKED}$ if $\mathcal{L} = \{0\}$. Otherwise, it outputs $\mathcal{L}$.

Theorem 7.1. *If* PRF *is a secure puncturable PRF with weak key-injectivity and constrained one-wayness,* PFE *is a secure puncturable functional encryption scheme as defined in Sec. 5.1,* F *is a secure prefix puncturable PRF,* G, G' *and* $\tilde{\mathsf{G}}$ *are pseudorandom generators,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* WM *is a secure watermarking scheme with collusion resistant unremovability and* δ*-unforgeability for* PKE.

Proof. Proof of Theorem 7.1 can be proceeded similiarly as the proof of Theorem 6.1, so we omit its details here.

One subtle issue in the proof is that the adversary can additionally obtain a public key from the challenge oracle, which is an obfuscated circuit containing the challenge key k^*. So, we need to further argue that the public key will not leak additional information of k^*. Recall that through the whole security proof, either k^* or its equivalent version or its contrained version punctured on a random point will appear in the view of the adversary. In the first case, the public key will not provide additional information about k^*. In the second case, k^* can be replaced with its equivalent version in the public key and due to the indistinguishability of iO, this cannot be detected by the adversary. In the third case, k^* can be replaced with its contrained version in the public key. Since the probability that the random punctrued point falls in the range of $\tilde{\mathsf{G}}$ is negligible, by the indistinguishability of iO, this will also not affect the adversary's advantage. □

Remark 7.1. We remark that the above strategy is not fully applicable in the watermarkable signature setting. This is because in the verification key of the

Ext	Trace
Constant: mpk, K **Input:** $a_1, \ldots, a_d, b, ind, r$ 1. $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$. 2. $x = \mathsf{PFE}.\mathtt{Enc}(mpk, t_1 \| \ldots \| t_d \| b \| ind; r)$. 3. $k' = \mathsf{F}.\mathtt{Eval}(K, t_1 \| \ldots \| t_d \| b)$. 4. $y = \mathsf{PRF}.\mathtt{Eval}(k', x)$. 5. Output (x, y).	**Input:** $ind_1, ind_2, p_1, p_2, \epsilon, T, \mathtt{E}, H, \mathtt{C}$ 1. $\Delta = \lvert p_1 - p_2 \rvert$. 2. If $\Delta \leq \epsilon$: return $\emptyset$. 3. If $ind_2 - ind_1 = 1$: return $\{ind_1\}$. 4. $ind_3 = \lfloor \frac{ind_1 + ind_2}{2} \rfloor$. 5. $p_3 = \mathtt{Test}(ind_3, T, \mathtt{E}, H, \mathtt{C})$. 6. Return $\mathtt{Trace}(ind_1, ind_3, p_1, p_3, \epsilon, T, \mathtt{E}, H, \mathtt{C}) \cup \mathtt{Trace}(ind_3, ind_2, p_3, p_2, \epsilon, T, \mathtt{E}, H, \mathtt{C})$.
M	**Test**
Constant: sk, K, H, k, msg **Input:** $ct = (x, z)$ 1. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$. 2. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_d))) = b)$ (a) $k' = \mathsf{F}.\mathtt{Eval}(K, t_1 \| \ldots \| t_d \| b)$. (b) $y = \mathsf{PRF}.\mathtt{Eval}(k', x)$. (c) Output $y \oplus z$. 3. Otherwise, output $\mathsf{PRF}.\mathtt{Eval}(k, x) \oplus z$.	**Input:** $ind, T, \mathtt{E}, H, \mathtt{C}$ 1. $Acc = 0$ 2. For $i \in [1, T]$: (a) Sample $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$ and $r \xleftarrow{\$} \mathcal{R}$. (b) Sample $z_1, \ldots, z_d, z^* \xleftarrow{\$} \{0,1\}^m$. (c) $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$. (d) $b = H(\mathtt{C}(\mathsf{G}(t_1), z_1) \oplus z_1, \ldots, \mathtt{C}(\mathsf{G}(t_d), z_d) \oplus z_d)$. (e) $(x, y) = \mathtt{E}(a_1, \ldots, a_d, b, ind, r)$. (f) If $\mathtt{C}(x, z^*) \oplus z^* = y$: $Acc = Acc + 1$. 3. Return $\frac{Acc}{T}$.

Fig. 3 The circuit `Ext`, the circuit `M`, the function `Trace`, and the function `Test` for the watermarkable PKE scheme.

signature scheme constructed in [SW14], the pseudorandom random function will compute on all points in its domain (rather than points in the range of a pseudorandom generator), thus, we cannot argue indistinguishability between a verification key generated from a normal key and that generated from a constrained key. To circumvent this problem, we modify the construction of signature scheme slightly and use a watermarked PRF key in the obfuscated circuit of the verification key. But this will lead to a weaker watermarkable signature scheme, which needs the marking key of the watermarking scheme when generating a signing key/verification key pair of the signature scheme.

8 Conclusion and Future Works

In this work, we initiate the study of collusion resistant watermarking by defining and constructing collusion resistant watermarking schemes for common cryptographic functionalities, including PRF, PKE, and signature.

One may note that watermarking schemes constructed in this work only achieve a $negl(\cdot)$-unremovability, which guarantees that no attacker can remove or modify the embedded message in a watermarked program via altering the program on a *negligible* fraction of inputs. A stronger form of unremovability, which is called ϵ-unremovability, considers attackers that can alter the watermarked

program on a ϵ fraction of inputs for some non-negligible ϵ. In this setting, since the attacker is able to reset the outputs on a non-negligible fraction of inputs, internal variables generated during the extraction procedure may significantly depart from what is expected. In previous works with ϵ-unremovability (e.g., [CHN+16, QWZ18, KW19]), this issue is tackled by repeating some subprocedure multiple times and deciding based on majority. Unfortunately, in our construction, as the extraction algorithm needs to analyze the fraction of reprogrammed points in a set, it seems implausible to use the "repeating-and-choosing-majority" trick. How to construct collusion resistant watermarking schemes with ϵ-unremovability for non-negligible ϵ is an interesting open problem.

Another interesting direction is to explore the possibility of instantiating a collusion resistant watermarkable PRF from standard assumptions. As discussed in Sec. 1.1, a collusion resistant watermarkable PRF can be approximately viewed as a collusion resistant constraint-hiding constrained PRF, which can imply indistinguishability obfuscator. However, we have not provided a formal reduction. It is interesting to formally construct an indistinguishability obfuscator from a collusion resistant watermarkable PRF or construct a collusion resistant watermarkable PRF from standard assumptions.

Besides, it is also interesting to construct collusion resistant watermarking schemes with other desirable features, e.g., constructing collusion resistant watermarking schemes with public marking.

Acknowledgement. We appreciate the anonymous reviewers for their valuable suggestions. Part of this work was supported by the National Natural Science Foundation of China (Grant No. 61572294, 61602396, 61632020, U1636205), Early Career Scheme research grant (ECS Grant No. 25206317) from the Research Grant Council of Hong Kong, the Innovation and Technology Support Programme of Innovation and Technology Fund of Hong Kong (Grant No. ITS/356/17), and the MonashU-PolyU-Collinstar Capital Joint Lab on Blockchain. Junzuo Lai was supported by National Natural Science Foundation of China (Grant No. 61922036, 61572235), and Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2015A030306045).

References

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

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

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

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

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

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

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos. In *TCC*, pages 474–502. Springer, 2016.

[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.

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

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, pages 171–190. IEEE, 2015.

[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.

[CHV15] Aloni Cohen, Justin Holmgren, and Vinod Vaikuntanathan. Publicly verifiable software watermarking. Cryptology ePrint Archive, Report 2015/373, 2015. `https://eprint.iacr.org/2015/373`.

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

[CVW+18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In *TCC*, 2018.

[DSDCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, pages 566–598. Springer, 2001.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[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.

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

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

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, 2018.

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

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. In *ASIACRYPT*, pages 341–358. Springer, 2010.

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

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

[KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In *EUROCRYPT*, pages 603–648. Springer, 2018.

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

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

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

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

[NW15] Ryo Nishimaki and Daniel Wichs. Watermarking cryptographic programs against arbitrary removal strategies. Cryptology ePrint Archive, Report 2015/344, 2015. https://eprint.iacr.org/2015/344.

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

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

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

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

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

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

[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 Cryptographic Primitives

In this section, we review cryptographic primitives employed in this work.

Pseudorandom Generators. Let λ be the security parameter, and l be a polynomial in λ that is larger than λ. A pseudorandom generator $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^l$ is a polynomial-time computable function that for all PPT adversary $\mathcal{A}$,

$$| \Pr[x \xleftarrow{\$} \{0,1\}^\lambda : \mathcal{A}(\mathsf{G}(x)) = 1] - \Pr[y \xleftarrow{\$} \{0,1\}^l : \mathcal{A}(y) = 1] | \leq negl(\lambda)$$

Collision Resistant Hash Families. Let λ be the security parameter, and m, n be polynomials in λ that $m > n$. A collision resistant hash family $\mathcal{H}$ is a family of polynomial-time computable function $H : \{0,1\}^m \to \{0,1\}^n$ that for all PPT adversary $\mathcal{A}$,

$$\Pr[H \xleftarrow{\$} \mathcal{H}, \mathcal{A}(1^\lambda, H) = (x_0, x_1) : x_0 \neq x_1 \wedge H(x_0) = H(x_1)] \leq negl(\lambda)$$

Adaptively statistically sound NIZK Proofs. The notion of NIZK proof was proposed by Blum et al. in [BFM88]. As shown in [FLS99, DSDCO+01, Gro10], an adaptively statistically sound NIZK proof can be constructed from (doubly-enhanced) trapdoor permutations, which can be instantiated from the factoring assumption or from the indistinguishability obfuscator plus one-way function [BPW16].

The NIZK proof for a language $\mathcal{L}$ consists of three PPT algorithms:

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

Also, it satisfies the following conditions:

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

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

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

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

  where σ is the state of $\mathcal{A}_1$ and $\mathcal{A}_1$ is required to output a valid statement/witness pair.

Puncturable Pseudorandom Function with Constrained One-wayness and Weak Key-Injectivity. The notion of puncturable pseudorandom function was first formalized by Sahai and Waters in [SW14]. They also show that a PRF constructed via the GGM-framework [GGM84] is a puncturable PRF. In this work, we will use a slightly stronger version of puncturable PRF, namely, puncturable PRF with weak key-injectivity[11] and constrained one-wayness, which is defined in Definition A.1. In [CHN+16], a puncturable PRF with weak key-injectivity is constructed from the LWE assumption under the GGM-framework. Moreover, it can be easily verified that a PRF constructed under the GGM-framework also has constrained one-wayness. So one can instantiate the puncturable PRF with weak key-injectivity and constrained one-wayness from the LWE assumption.

Definition A.1. *A puncturable PRF family with weak key-injectivity, constrained one-wayness, 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 $k \in \mathcal{K}$*[12]*.*
- ***Eval.*** *On input a secret key $k \in \mathcal{K}$ and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs a string $y \in \{0,1\}^m$.*
- ***Constrain.*** *On input a secret keys $k \in \mathcal{K}$ and a polynomial-size set $\mathcal{S} \subseteq \{0,1\}^n$, the constrain algorithm outputs a punctured key ck.*
- ***ConstrainEval.*** *On input a punctured key ck and an input $x \in \{0,1\}^n$, the constrained evaluation algorithm outputs an string $y \in \{0,1\}^m \cup \{\perp\}$.*

Also, it satisfies the following conditions:

[11] This is in fact the "key-injectivity" property defined in [CHN+16], here we call this property weak key-injectivity to distinguish it from the "key-injectivity" property defined in [KW17].

[12] Here, we require that the key generation algorithm will output a uniform key in $\mathcal{K}$.

- ***Correctness.*** *For any $k \in \mathcal{K}$, any polynomial size set $\mathcal{S} \subseteq \{0,1\}^n$, and any $x \in \{0,1\}^n \backslash \mathcal{S}$, let $ck \leftarrow \texttt{Constrain}(k, \mathcal{S})$, then we have $\texttt{ConstrainEval}(ck, x) = \texttt{Eval}(k, x)$.*
- ***Weak Key-Injectivity.*** *Let $k_1 \leftarrow \texttt{KeyGen}(1^\lambda)$, then we have*

$$\Pr[\exists k_2 \in \mathcal{K}, x \in \{0,1\}^n, s.t.\, k_1 \neq k_2 \wedge \texttt{Eval}(k_1, x) = \texttt{Eval}(k_2, x)] \leq negl(\lambda)$$

- ***Pseudorandomness.*** *For all PPT adversary $\mathcal{A}$,*

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

where $FUN_{n,m}$ denotes the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$, the oracle $\mathcal{O}_k^{PR}(\cdot)$ takes as input a string $x \in \{0,1\}^n$ and returns $\texttt{Eval}(k, x)$, and the oracle $\mathcal{O}_f^{R}(\cdot)$ takes as input a string $x \in \{0,1\}^n$ and returns $f(x)$.
- ***Constrained One-wayness.*** *For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\Pr\left[\begin{array}{l}(x, \sigma) \leftarrow \mathcal{A}_1(1^\lambda); \\ k \leftarrow \texttt{KeyGen}(1^\lambda); \\ ck \leftarrow \texttt{Constrain}(k, \{x\}); \\ y = \texttt{Eval}(k, x);\end{array} : \mathcal{A}_2(\sigma, ck, y) = k\right] \leq negl(\lambda)$$

where σ is the state of $\mathcal{A}_1$.
- ***Constrained Pseudorandomness.*** *For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have*

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

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

Prefix Puncturable Pseudorandom Function. The notion of prefix puncturable PRF was formally introduced in [NW15]. It was also shown that the GGM-framework can lead to a prefix puncturable PRF. Now, we recall its definition.

Definition A.2. *A prefix puncturable PRF family 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 $k \in \mathcal{K}$.*
- ***Eval.*** *On input a secret key $k \in \mathcal{K}$ and an input $x \in \{0,1\}^n$, the evaluation algorithm outputs a string $y \in \{0,1\}^m$.*

- ***Constrain.*** *On input a secret keys $k \in \mathcal{K}$ and a string $t \in \{0,1\}^{\leq n}$, 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 a string $y \in \{0,1\}^m \cup \{\perp\}$.*

Also, it satisfies the following conditions:

- ***Correctness.*** *For any $k \in \mathcal{K}$, any $t \in \{0,1\}^{\leq n}$, and any $x \in \{0,1\}^n \backslash t \| \{0,1\}^{n-\|t\|}$, let $ck \leftarrow \mathtt{Constrain}(k,t)$, then we have $\mathtt{ConstrainEval}(ck,x) = \mathtt{Eval}(k,x)$.*
- ***Constrained Pseudorandomness.*** *For any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\Pr\left[\begin{array}{l}(t,\sigma) \leftarrow \mathcal{A}_1(1^\lambda);\\ k \leftarrow \mathtt{KeyGen}(1^\lambda);\\ ck \leftarrow \mathtt{Constrain}(k,t); : b' = b\\ b \xleftarrow{\$} \{0,1\};\\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{k,b}(\cdot)}(\sigma, ck);\end{array}\right] \leq 1/2 + negl(\lambda)$$

  where σ is the state of $\mathcal{A}_1$. Here, the oracle $\mathcal{O}_{k,0}(\cdot)$ takes as input a string $x \in \{0,1\}^n$ with prefix t and outputs $\mathtt{Eval}(k,x)$, and the oracle $\mathcal{O}_{k,1}(\cdot)$ takes as input a string $x \in \{0,1\}^n$ with prefix t and outputs $f(x)$, where f is a truly random function and is computed via lazy sampling.

Functional Encryption. The notion of functional encryption is formally introduced in [BSW11, O'N10]. In this work, we need a (two-key) adaptively secure functional encryption scheme with perfect correctness for a family $\mathcal{F}$ of polynomial-size circuit, which has been constructed in previous works, e.g., [GVW12]. Next, we recall the definition of functional encryption schemes.

Definition A.3. *A functional encryption scheme for a function family $\mathcal{F}$ with message space $\{0,1\}^m$ and ciphertext space $\{0,1\}^n$ consists of four algorithms:*

- ***Setup.*** *On input the security parameter λ, the setup algorithm outputs the master public key/master secret key pair (mpk, msk).*
- ***KeyGen.*** *On input the master secret key msk and a function $f \in \mathcal{F}$, the key generation algorithm outputs a secret key sk for f.*
- ***Enc.*** *On input the master public key mpk and the message $msg \in \{0,1\}^m$, the encryption algorithm outputs the ciphertext ct.*
- ***Dec.*** *On input a secret key sk and a ciphertext $ct \in \{0,1\}^n$, the decryption algorithm outputs a string msg.*

Also, it satisfies the following conditions:

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

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

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

where σ is the state of $\mathcal{A}_1$. Here, $\mathcal{O}_{msk}$ takes as input a function $f \in \mathcal{F}$ and outputs a secret key $sk \leftarrow \mathtt{KeyGen}(msk, f)$. We require that for all f submitted to the oracle $\mathcal{O}_{msk}$, $f(msg_0) = f(msg_1)$ and the oracle $\mathcal{O}_{msk}$ can only be queried two times.

Puncturable Encryption. The puncturable encryption scheme was first presented and constructed in [CHV15, CHN+16], and we recall its definition here.

Definition A.4. *A puncturable encryption scheme with message space $\{0,1\}^l$ and ciphertext space $\{0,1\}^n$ consists of four algorithms:*

- ***KeyGen.*** *On input the security parameter λ, the key generation algorithm outputs the public key/secret key (pk, sk).*
- ***Puncture.*** *On input a secret keys sk and two ciphertexts $c_0, c_1 \in \{0,1\}^n$, the puncture algorithm outputs a punctured secret key sk'.*
- ***Enc.*** *On input a public key pk and a message $m \in \{0,1\}^l$, the encryption algorithm outputs a ciphertext c.*
- ***Dec.*** *On input a secret key (or a punctured secret key) sk and a ciphertext $c \in \{0,1\}^n$, the decryption algorithm outputs a valid message in $\{0,1\}^l$ or a symbol $\perp$ indicating decryption failure.*

Also, it satisfies the following conditions:

- ***Correctness.*** *For any message $m \in \{0,1\}^l$, let $(pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$, and $c \leftarrow \mathtt{Enc}(pk, m)$, then we have $\Pr[\mathtt{Dec}(sk, c) = m] = 1$.*
- ***Punctured Correctness.*** *For any strings $c_0, c_1, c^* \in \{0,1\}^n$ that $c^* \notin \{c_0, c_1\}$, let $(pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$ and $sk' \leftarrow \mathtt{Puncture}(sk, \{c_0, c_1\})$, then we have $\Pr[\mathtt{Dec}(sk, c^*) = \mathtt{Dec}(sk', c^*)] = 1$.*
- ***Sparseness.*** *Let $(pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$, and let $c \xleftarrow{\$} \{0,1\}^n$, then we have $\Pr[\mathtt{Dec}(sk, c) \neq \perp] \leq negl(\lambda)$.*

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

$$\Pr\left[\begin{array}{l}(m^*, \sigma) \leftarrow \mathcal{A}_1(1^\lambda);\\ (pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda);\\ c^* \leftarrow \mathtt{Enc}(pk, m^*);\\ r^* \xleftarrow{\$} \{0,1\}^n;\\ sk' \leftarrow \mathtt{Puncture}(sk, \{c^*, r^*\});\\ b \xleftarrow{\$} \{0,1\};\\ Y_0 = (c^*, r^*);\\ Y_1 = (r^*, c^*);\end{array} : \mathcal{A}_2(\sigma, pk, sk', Y_b) = b\right] \leq 1/2 + negl(\lambda)$$

where σ *is the state of* $\mathcal{A}_1$*.*

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 A.5 ([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* $\mathtt{C} \in \mathcal{C}_\lambda$*, we have*

$$\Pr[\mathtt{C}' \leftarrow \mathsf{iO}(\mathtt{C}) : \mathtt{C}' \equiv \mathtt{C}] = 1$$

- ***Indistinguishability.*** *For any PPT adversary* $\mathcal{A}$*, for all security parameters* $\lambda \in \mathbb{N}$*, and all pairs of circuits* $\mathtt{C}_0, \mathtt{C}_1 \in \mathcal{C}_\lambda$ *that* $\mathtt{C}_0 \equiv \mathtt{C}_1$*, we have*

$$|\Pr[\mathcal{A}(\mathsf{iO}(\mathtt{C}_0)) = 1] - \Pr[\mathcal{A}(\mathsf{iO}(\mathtt{C}_1)) = 1]| \leq negl(\lambda)$$

B Proof of Theorem 5.1

In this section, we present the proof of Theorem 5.1.

Proof. Correctness of PFE comes from correctness of PE, completeness of NIZK and correctness of FE directly. Sparseness of PFE comes from sparseness of PE directly. Punctured correctness of PFE comes from punctured correctness of PE directly. Ciphertext pseudorandomness of PFE comes from ciphertext pseudorandomness of PE by a direct reduction. It remains to show the iO-compatible correctness and the adaptive indistinguishability of PFE.

iO-Compatible Correctness. First, for any master public key/master secret key pair $(MPK, MSK) = ((mpk, crs, pk), (msk, sk, mpk, crs))$, we define

$$\mathcal{V}_{(MPK,MSK)} = \{CT \in \{0,1\}^l \mid \exists (ct, \pi) \in \{0,1\}^{n+n'}, \\ (ct, \pi) = \mathsf{PE}.\mathtt{Dec}(sk, CT) \wedge \mathsf{NIZK}.\mathtt{Verify}(crs, (mpk, ct), \pi) = 1\}$$

and

$$\mathcal{I}_{(MPK,MSK)} = \{0,1\}^l \backslash \mathcal{V}_{(MPK,MSK)}$$

For any $CT \in \mathcal{I}_{(MPK,MSK)}$, let $ct \| \pi = \mathsf{PE}.\mathtt{Dec}(sk, CT)$, then we have either $ct \| \pi = \perp$ or $\mathsf{NIZK}.\mathtt{Verify}(crs, (mpk, ct), \pi) = 0$. In each case, the decryption algorithm will output $\perp$ on any secret key SK generated from MSK.

Next, we consider ciphertexts in $\mathcal{V}_{(MPK,MSK)}$. First, by the adaptively statistical soundness of NIZK, with all but negligible probability over the choice of crs, for any $CT \in \mathcal{V}_{(MPK,MSK)}$, let $ct \| \pi = \mathsf{PE}.\mathtt{Dec}(sk, CT)$, we have $(mpk, ct) \in \mathcal{L}$, i.e., there exists $(msg, r) \in \{0,1\}^m \times \mathcal{R}$ that $ct = \mathsf{FE}.\mathtt{Enc}(mpk, msg; r)$. Next, by the perfect correctness of FE, for any $f \in \mathcal{F}$, let $sk \leftarrow \mathsf{FE}.\mathtt{KeyGen}(msk, f)$, then we have $\mathsf{FE}.\mathtt{Dec}(sk, ct) = f(msg)$. Thus, as long as a "good" common reference string is chosen, which occurs with all but negligible probability, even an unbounded adversary cannot generate a ciphertext in $\mathcal{V}_{(MPK,MSK)}$ that causes a conflict when decrypting with different secret keys.

This completes the proof of iO-compatible correctness.

Adaptive Indistinguishability. To prove adaptive indistinguishability of PFE, we define the following games between a challenger and a PPT adversary $\mathcal{A}$:

- *Game 0.* This is the real experiment in the definition of adaptive indistinguishability. In more detail, the challenger proceeds as follows:
  1. In the beginning, the challenger generates $(mpk, msk) \leftarrow \mathsf{FE}.\mathtt{Setup}(1^\lambda)$, $crs \leftarrow \mathsf{NIZK}.\mathtt{KeyGen}(1^\lambda)$, and $(pk, sk) \leftarrow \mathsf{PE}.\mathtt{KeyGen}(1^\lambda)$. Then it returns the master public key $MPK = (mpk, crs, pk)$ to $\mathcal{A}$.
  2. Next, the challenger answers oracle queries for $\mathcal{A}$ and on input a function $f \in \mathcal{F}$, it generates $fsk \leftarrow \mathsf{FE}.\mathtt{KeyGen}(msk, f)$ and returns $SK = (fsk, sk, mpk, crs)$ to $\mathcal{A}$.
  3. Finally, on input two messages msg_0^*, msg_1^*, the challenger first samples $b \xleftarrow{\$} \{0,1\}$. Then it samples $r^* \in \mathcal{R}$ and computes $ct^* = \mathsf{FE}.\mathtt{Enc}(mpk, msg_b^*; r^*)$ and $\pi^* \leftarrow \mathsf{NIZK}.\mathtt{Prove}(crs, (mpk, ct^*), (msg_b^*, r^*))$. Finally, it returns $CT^* \leftarrow \mathsf{PE}.\mathtt{Enc}(pk, ct^* \| \pi^*)$ to $\mathcal{A}$ and outputs 1 if $\mathcal{A}$ succeeds in guessing b.
- *Game 1.* This is identical to Game 0 except that the challenger uses the simulator of NIZK to generate crs and π^*:

$$(crs, state) \leftarrow \mathsf{NIZK}.\mathtt{S}_1(1^\lambda)$$

$$\pi^* \leftarrow \mathsf{NIZK}.\mathtt{S}_2(1^\lambda, (mpk, ct^*), state)$$

Indistinguishability between Game 0 and Game 1 comes from adaptive zero-knowledge of NIZK directly. Moreover, negligibility of $\mathcal{A}$'s advantage in Game 1 comes from the adaptive indistinguishability of FE via a direct reduction.

This completes the proof of adaptive indistinguishability of PFE. □

C Proof of Theorem 6.1

In this section, we present the proof of Theorem 6.1. Before starting the description of the proof, we first recall a few useful notions and facts about the trace function defined in Figure 1. Note that those notions and facts are explained in previous works [BCP14,NWZ16]. Here, we briefly review them for completeness.

First, during each invocation of the extraction algorithm, the test algorithm will be invoked at most once on each input ind, so we use $\mathfrak{p}_{ind}$ to denote the result of $\mathtt{Test}(ind, -, -, -, -)$[13] (we also set $\mathfrak{p}_0 = 1$ and $\mathfrak{p}_{2^\kappa} = 0$). We use an interval $[ind_1, ind_2)$ to denote an invocation of the algorithm $\mathtt{Trace}(ind_1, ind_2, -, -, -, -, -, -, -, -)$. The trace algorithm will be invoked recursively, and we can use a binary tree to denote all potential calls of intervals. The root of the tree is the interval $[0, 2^\kappa)$ and the leaf nodes are intervals $[a, a+1)$ for $a \in [0, 2^\kappa - 1]$. For each non-leaf node $[ind_1, ind_2)$, it has two children, namely, $[ind_1, \lfloor \frac{ind_1+ind_2}{2} \rfloor)$ and $[\lfloor \frac{ind_1+ind_2}{2} \rfloor, ind_2)$, each of which is half in size of its parent $[ind_1, ind_2)$, and both children will be called if and only if $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}|$ is large enough. It is easy to see that the tree is of height $\kappa + 1$ and all intervals at the same level are disjoint. Here, we use the notion "level" to denote the distance between a node and the root node, i.e., the root node is at level 0 while the children of a level-i node is at level $i + 1$. Now, we are ready to describe a useful lemma that will be extensively used in proving the correctness and security of our watermarking scheme:

Lemma C.1 ([BCP14, NWZ16]). *Let $\mathcal{C} \subseteq [0, 2^\kappa)$ be a set with polynomial elements. Let $q \geq \|\mathcal{C}\|$ be a polynomial. Consider an invocation of the algorithm $\mathtt{Trace}(0, 2^\kappa, 1, 0, \epsilon, -, -, -, -)$ where $\epsilon = 1/((\kappa + 1) \cdot q + 1)$, and let $\mathcal{L}$ be the output. Now, if $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| \leq \epsilon$ for every interval $[ind_1, ind_2)$ that is called during this procedure and satisfies $[ind_1, ind_2) \cap \mathcal{C} = \emptyset$, then we have:*

1. *The $\mathtt{Test}$ algorithm will be invoked at most $q \cdot (\kappa + 1)$ times.*
2. *$\mathcal{L} \subseteq \mathcal{C}$.*
3. *$\mathcal{L} \neq \emptyset$.*

Proof. First, the $\mathtt{Test}$ algorithm will be called in an interval $[ind_1, ind_2)$ if and only if $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| > \epsilon$, so only those intervals containing at least one element in $\mathcal{C}$ will invoke the $\mathtt{Test}$ algorithm. Thus, at each level of the tree, there are at most q intervals that are able to run the $\mathtt{Test}$ algorithm. Therefore, the $\mathtt{Test}$ algorithm will be invoked at most $q \cdot (\kappa + 1)$ times.

Then, note that the output of the $\mathtt{Trace}$ algorithm is in fact a set consisting of all called leaf-nodes $[ind, ind+1)$ satisfying $|\mathfrak{p}_{ind} - \mathfrak{p}_{ind+1}| > \epsilon$. So no elements outside $\mathcal{C}$ will be included in the output set.

Finally, to prove the third statement, we first prove the following claim:

Claim 1. *Let $[ind_1, ind_2)$ be a node at level h, if the $\mathtt{Trace}$ algorithm is invoked at this interval, $k > 0$ and $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| > \epsilon \cdot (\kappa + 1 - h) \cdot k$, then the algorithm will output a non-empty set, where $k = [ind_1, ind_2) \cap \mathcal{C}$.*

[13] For simplicity of notation, in this section, we use $-$ to denote inputs that are not needed to be specified, and its value is defined by the context.

Proof. We prove Claim 1 recursively. First, in case $h = \kappa$, i.e., the interval $[ind_1, ind_2)$ is a leaf-node, the claim holds obviously.

Then assuming the claim holds for $h \geq l+1$, we prove that it holds for $h = l$, where $0 \leq l \leq \kappa - 1$. Let $[ind_1, ind_2)$ be a called node at level l that $k = [ind_1, ind_2) \cap \mathcal{C}$ is positive and $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| > \epsilon \cdot (\kappa + 1 - l) \cdot k$. Then, it will invoke its two children since $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| > \epsilon \cdot (\kappa + 1 - l) \cdot k > \epsilon$. Also, let $ind_3 = \lfloor \frac{ind_1 + ind_2}{2} \rfloor$, $k_l = [ind_1, ind_3) \cap \mathcal{C}$, and $k_r = [ind_3, ind_2) \cap \mathcal{C}$. Obviously, we have $k = k_l + k_r$.

Now, if both k_l and k_r are positive. Then we have either $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_3}| > \epsilon \cdot (\kappa + 1 - l) \cdot k_l$ or $|\mathfrak{p}_{ind_3} - \mathfrak{p}_{ind_2}| > \epsilon \cdot (\kappa + 1 - l) \cdot k_r$ (Otherwise, $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| \leq |\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_3}| + |\mathfrak{p}_{ind_3} - \mathfrak{p}_{ind_2}| \leq \epsilon \cdot (\kappa + 1 - l) \cdot (k_l + k_r) = \epsilon \cdot (\kappa + 1 - l) \cdot k$), and w.l.o.g, we assume $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_3}| > \epsilon \cdot (\kappa + 1 - l) \cdot k_l$. Then, we have $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_3}| > \epsilon \cdot (\kappa + 1 - (l+1)) \cdot k_l$, and by assumption, the algorithm running on $[ind_1, ind_3)$ will output a non-empty set.

Otherwise, i.e., either k_l or k_r is 0, w.l.o.g, we assume $k_r = 0$. Then we have $|\mathfrak{p}_{ind_3} - \mathfrak{p}_{ind_2}| \leq \epsilon$, which implies that $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_3}| > \epsilon \cdot (\kappa + 1 - l) \cdot k - \epsilon \geq \epsilon \cdot (\kappa + 1 - (l+1)) \cdot k$. Also, by assumption, the algorithm running on $[ind_1, ind_3)$ will output a non-empty set.

To summarize, in both cases, the algorithm will invoke its children and at least one of them will output a non-empty set, thus, the algorithm on $[ind_1, ind_2)$ will also output a non-empty set. □

Now, considering the case $h = 0$ for Claim 1, since $1 = \epsilon \cdot ((\kappa + 1) \cdot q + 1) > \epsilon \cdot (\kappa + 1) \cdot \|\mathcal{C}\|$, the output will be non-empty. □

Note that the proof of the second statement in Lemma C.1, i.e., $\mathcal{L} \subseteq \mathcal{C}$, does not rely on the condition that $q \geq \|\mathcal{C}\|$. So we have:

Lemma C.2. *Let $\mathcal{C} \subseteq [0, 2^\kappa)$ be a set with polynomial elements. Let $q < \|\mathcal{C}\|$ be a polynomial. Consider an invocation of the algorithm* $\texttt{Trace}(0, 2^\kappa, 1, 0, \epsilon, -, -, -, -)$ *where $\epsilon = 1/((\kappa+1) \cdot q + 1)$, and let $\mathcal{L}$ be the output. Now, if $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| \leq \epsilon$ for every interval $[ind_1, ind_2)$ that is called during this procedure and satisfies $[ind_1, ind_2) \cap \mathcal{C} = \emptyset$, then we have $\mathcal{L} \subseteq \mathcal{C}$.*

Remark C.1. It is worth noting that the `Test` algorithm in the above two lemmas can be replaced with any algorithm that outputs real values between [0,1], and the lemmas still hold.

Now, we are ready to prove that WM is a secure watermarking scheme for PRF. In particular, we need to argue its correctness (Appendix C.1), collusion resistant unremovability (Appendix C.3), and δ-unforgeability (Appendix C.2).

C.1 Proof of Correctness

Functionality Preserving. Functionality preserving of WM comes from the sparseness of PFE and the correctness of iO directly.

Extraction Correctness. For any key $k \in \mathcal{K}$, message $msg \in [1, 2^\kappa)$, and polynomial $q \geq 1$, let $(MK, EK) \leftarrow \mathsf{WM}.\mathtt{Setup}(1^\lambda)$ and $\mathtt{C} \leftarrow \mathsf{WM}.\mathtt{Mark}(MK, k, msg)$.

We start the proof of extraction correctness by inspecting the output of $\mathtt{Test}(ind, -, -, -, \mathtt{C})$ for $ind \in [1, 2^\kappa)$. In more detail, let $\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{r}, \tilde{t}_1, \ldots, \tilde{t}_d, \tilde{b}, \tilde{x}, \tilde{y}$ be internal variables used in one iteration of the for loop in the $\mathtt{Test}(ind, -, -, -, \mathtt{C})$ algorithm, we first examine if $\mathtt{C}(\tilde{x}) = \tilde{y}$.

First, if $ind > msg$, by the correctness of PFE and iO, $\mathtt{C}(\tilde{x}) = \mathsf{PRF}.\mathtt{Eval}(k, \tilde{x})$. Also, by the pseudorandomness of F, the (pseudo)-random key $k' = \mathsf{F}.\mathtt{Eval}(K, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b})$ equals to k with only a negligible probability. Then, by the weak key-injectivity of PRF, the probability that $\mathsf{PRF}.\mathtt{Eval}(k, \tilde{x}) = \mathsf{PRF}.\mathtt{Eval}(k', \tilde{x})$ is negligible. Thus, the probability that $\mathtt{C}(\tilde{x})$ equals to $\tilde{y} = \mathsf{PRF}.\mathtt{Eval}(k', \tilde{x})$ is negligible.

Then, if $ind \leq msg$, again, by the correctness of PFE and iO, $\mathtt{C}(\tilde{x}) = \tilde{y}$ if

$$\forall i \in [1, d],\ \mathtt{C}(\mathsf{G}(\mathsf{G}'(\tilde{a}_i))) = \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(\mathsf{G}'(\tilde{a}_i))) \tag{1}$$

By the pseudorandomness of G and G', which implies the pseudorandomness of $\mathsf{G}(\mathsf{G}'(\cdot))$ and the sparseness of PFE, Equation (1) holds with all but negligible probability.

Finally, by the union bound, with all but negligible probability, we have $\mathtt{Test}(ind, -, -, -, \mathtt{C}) = 0$ if $ind > msg$ and $\mathtt{Test}(ind, -, -, -, \mathtt{C}) = 1$ if $ind \leq msg$.

Now, with all but negligible probability, for any called interval $[ind_1, ind_2)$ that does not contain msg, $\mathtt{Test}(ind_1, -, -, -, \mathtt{C}) = \mathtt{Test}(ind_2, -, -, -, \mathtt{C})$, thus, by Lemma C.1, the extraction algorithm will return a non-empty subset of $\{msg\}$, i.e., $\{msg\}$, with all but negligible probability. This completes the proof of extraction correctness.

C.2 Proof of Unforgeability

In this section, we prove the δ-unforgeability of WM. We start with an auxiliary lemma, which indicates that for any PPT adversary, it cannot predict the reprogrammed value on a random punctured point.

Lemma C.3. *If PFE has sparseness and correctness, F is a secure prefix puncturable PRF, G and G' are pseudorandom generators, $\mathcal{H}$ is a family of collision-resistant hash function, and iO is a secure indistinguishability obfuscator, then for any PPT unforging-admissible adversary $\mathcal{A}$, the probability that $\mathtt{AUF}_{\mathcal{A}}(1^\lambda) = 1$ is negligible, where $\mathtt{AUF}$ is defined as follows:*

1. *In the beginning, the challenger first samples $H \xleftarrow{\$} \mathcal{H}$ and generates $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then it generates $(mpk, msk) \leftarrow \mathsf{PFE}.\mathtt{Setup}(1^\lambda)$ and $sk \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, \mathtt{ID})$ for the identity function $\mathtt{ID}$. Next, it computes $\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}[mpk, K])$, and returns $EK = (H, \mathtt{E})$ to $\mathcal{A}$.*

2. *Next, the challenger answers marking oracle queries from $\mathcal{A}$ and on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns* $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k_\iota, msg_\iota])$.
3. *Finally, $\mathcal{A}$ will output a circuit $\tilde{\mathtt{C}}$ together with an integer $ind^* \in [1, 2^\kappa)$, and then the challenger works as follows:*
   (a) *Sample* $\tilde{a}_1, \ldots, \tilde{a}_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$ *and* $r \xleftarrow{\$} \mathcal{R}$.
   (b) $\tilde{t}_1 = \mathsf{G}'(\tilde{a}_1), \ldots, \tilde{t}_d = \mathsf{G}'(\tilde{a}_d)$.
   (c) $\tilde{b} = H(\tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_1)), \ldots, \tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_d)))$.
   (d) $(\tilde{x}, \tilde{y}) = \mathtt{E}(\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{b}, ind^*, r)$.
   (e) *If* $\tilde{\mathtt{C}}(\tilde{x}) = \tilde{y}$, *then outputs 1; otherwise, output 0.*

We prove Lemma C.3 later in Appendix C.2.1. Next, we argue the δ-unforgeability of WM based on Lemma C.3.

Let $\tilde{\mathtt{C}}$ be the circuit returned in the δ-unforgeability experiment. Then by Lemma C.3, for any $ind \in [1, 2^\kappa)$ that $\mathtt{Test}(ind, -, -, -, \tilde{\mathtt{C}})$ is invoked when running the extraction algorithm on $\tilde{\mathtt{C}}$, $\mathtt{Test}(ind, -, -, -, \tilde{\mathtt{C}}) = 0$ with all but negligible probability. That is to say, with all but negligible probability, for any invoked interval $[ind_1, ind_2)$ that does not contain 0, $\mathtt{Test}(ind_1, -, -, -, \mathtt{C}) = \mathtt{Test}(ind_2, -, -, -, \mathtt{C})$, thus, by Lemma C.1, with all but negligible probability, the trace algorithm will return $\{0\}$ after invoking the $\mathtt{Test}$ algorithm κ times, which implies that the extraction algorithm will output $\mathsf{UNMARKED}$. This completes the proof of δ-unforgeability.

C.2.1 Proof of Lemma C.3 In this section, we prove Lemma C.3. First, we define the following games between a challenger and a PPT unforging-admissible adversary $\mathcal{A}$:

- *Game 0.* This is the real experiment $\mathtt{AUF}$.
- *Game 1.* This is identical to Game 0 except that in step 3, after computing $\tilde{t}_1, \ldots, \tilde{t}_d$ and $\tilde{b}$, the challenger further checks if $\tilde{t}_1, \ldots, \tilde{t}_d$ and $\tilde{b}$ define "marked points" for circuits returned by the marking oracle. More precisely, assuming $\mathcal{A}$ has made Q marking oracle queries, then for $i \in [1, Q]$, the challenger computes

$$b_i = H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d)))$$

  where k_i is the secret key submitted in the ith marking oracle query. Next, it aborts and outputs 2 if there exists $i \in [1, Q]$ that $b_i = \tilde{b}$; otherwise, it proceeds identically as in Game 0.
- *Game 2.* This is identical to Game 1 except that the challenger computes $\tilde{x}$ and $\tilde{y}$ as follows:
  1. $\tilde{x} \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \| ind^*)$.
  2. $\tilde{k}' = \mathsf{F}.\mathtt{Eval}(K, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b})$
  3. $\tilde{y} = \mathsf{PRF}.\mathtt{Eval}(\tilde{k}', \tilde{x})$.
- *Game 3.* This is identical to Game 2 except that the challenger modifies the way to generate $\tilde{t}_1, \ldots, \tilde{t}_d$. More precisely, it samples $\tilde{t}_1, \ldots, \tilde{t}_d \xleftarrow{\$} \{0,1\}^l$ in the beginning of step 1 instead of computing them from $\tilde{a}_1, \ldots, \tilde{a}_d$ in step 3.

- *Game 4.* This is identical to Game 3 except that the challenger uses a punctured version of K instead of using K directly in obfuscated circuits. More precisely, the challenger computes

$$CK \leftarrow \mathsf{F}.\mathtt{Constrain}(K, \tilde{t}_1 \| \dots \| \tilde{t}_d)$$

after generating K (recall that this means CK cannot compute on inputs with prefix $\tilde{t}_1 \| \dots \| \tilde{t}_d$). Then it computes

$$\mathsf{E} \leftarrow \mathsf{iO}(\mathtt{Ext}^1[mpk, CK])$$

where the circuit $\mathtt{Ext}^1$ is defined in Figure 4, and returns $EK = (H, \mathsf{E})$ to $\mathcal{A}$. Besides, in step 2, on receiving a marking oracle query (k_ι, msg_ι), the challenger first computes

$$b_\iota = H(\mathsf{PRF}.\mathtt{Eval}(k_\iota, \mathsf{G}(\tilde{t}_1)), \dots, \mathsf{PRF}.\mathtt{Eval}(k_\iota, \mathsf{G}(\tilde{t}_d)))$$

Then it sets $\alpha_\iota = \tilde{t}_1 \| \dots \| \tilde{t}_d \| b_\iota$ and computes $\beta_\iota = \mathsf{F}.\mathtt{Eval}(K, \alpha_\iota)$. After that, it returns

$$\mathsf{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}^1[sk, CK, H, k_\iota, msg_\iota, \alpha_\iota, \beta_\iota])$$

to $\mathcal{A}$, where the circuit $\mathtt{M}^1$ is defined in Figure 4.
- *Game 5.* This is identical to Game 4 except that the challenger computes $\beta_\iota = f(\alpha_\iota)$ with a truly random function f.[14] The challenger also computes $\tilde{k}' = f(\tilde{t}_1 \| \dots \tilde{t}_d \| \tilde{b})$.
- *Game 6.* This is identical to Game 5 except that the challenger samples $\tilde{y}$ uniformly at random from $\{0,1\}^m$.

Next, we prove the indistinguishability of each consecutive pair of games defined above and show that the adversary $\mathcal{A}$ will win in the final game (Game 6) with a negligible probability. For simplicity of notation, we use $\mathcal{E}_i$ to denote the output of Game i.

Claim 2. *If* PFE *has sparseness,* iO *has correctness, both* G *and* G' *are pseudorandom generators and* $\mathcal{H}$ *is a family of collision-resistant hash functions, then* $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. For $i \in [1, Q]$, we use $\mathbf{Bad}_i$ to denote the event that $b_i = \tilde{b}$ in Game 1. It is obvious that the outputs of Game 0 and Game 1 are identical as long as for all $i \in [1, Q]$, $\mathbf{Bad}_i$ does not occur. So it is sufficient to prove that

$$\Pr[\exists i \in [1, Q],\ \mathbf{Bad}_i] \leq negl(\lambda) \tag{2}$$

By the union bound, Equation (2) holds as long as for any $i \in [1, Q]$, $\Pr[\mathbf{Bad}_i] \leq negl(\lambda)$. Next, we will analyze the probability that $\mathbf{Bad}_i$ occurs for any fixed $i \in [1, Q]$.

[14] f is computed via lazy sampling, i.e., if α_ι is fresh, then β_ι is sampled uniformly from $\mathcal{K}$, and if there exists $\iota' < \iota$ that $\alpha_\iota = \alpha_{\iota'}$, then β_ι is set to be $\beta_{\iota'}$.

$\mathtt{Ext}^1$

Constant: mpk, CK
Input: $a_1, \ldots, a_d, b, ind, r$

1. $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$.
2. $x = \mathsf{PFE}.\mathtt{Enc}(mpk, t_1 \| \ldots \| t_d \| b \| ind; r)$.
3. $k' = \mathsf{F}.\mathtt{ConstrainEval}(CK, t_1 \| \ldots \| t_d \| b)$.
4. $y = \mathsf{PRF}.\mathtt{Eval}(k', x)$.
5. Output (x, y).

$\mathtt{M}^1$

Constant: $sk, CK, H, k, msg, \alpha, \beta$
Input: x

1. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$.
2. If $(t_1 \| \ldots \| t_d \| b = \alpha) \wedge (ind \leq msg)$
   (a) $y = \mathsf{PRF}.\mathtt{Eval}(\beta, x)$.
   (b) Output y.
3. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_d))) = b)$
   (a) $k' = \mathsf{F}.\mathtt{ConstrainEval}(CK, t_1 \| \ldots \| t_d \| b)$.
   (b) $y = \mathsf{PRF}.\mathtt{Eval}(k', x)$.
   (c) Output y.
4. Otherwise, output $\mathsf{PRF}.\mathtt{Eval}(k, x)$.

Fig. 4 The circuits $\mathtt{Ext}^1$ and $\mathtt{M}^1$.

First, by the unforging-admissibility of $\mathcal{A}$, we have $\tilde{\mathtt{C}}$ and $\mathtt{C}_i$ differ on at least a δ fraction of inputs. Also, by sparseness of PFE and correctness of iO, $\mathtt{C}_i$ and $\mathsf{PRF}.\mathtt{Eval}(k_i, \cdot)$ differ on only a negligible fraction of inputs, thus, we have $\tilde{\mathtt{C}}$ and $\mathsf{PRF}.\mathtt{Eval}(k_i, \cdot)$ differ on at least a $\delta - negl(\lambda)$ fraction of inputs. So, for $x \xleftarrow{\$} \{0,1\}^n$, we have

$$\Pr[\tilde{\mathtt{C}}(x) = \mathsf{PRF}.\mathtt{Eval}(k_i, x)] \leq 1 - (\delta - negl(\lambda))$$

Next, let $x_1, \ldots, x_d \xleftarrow{\$} \{0,1\}^n$, and let $\mathbf{F}_i$ be the event that

$$(\mathsf{PRF}.\mathtt{Eval}(k_i, x_1), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, x_d)) = (\tilde{\mathtt{C}}(x_1), \ldots, \tilde{\mathtt{C}}(x_d))$$

Then, we have

$$\begin{aligned}
\Pr[\mathbf{F}_i] &\leq (1 - (\delta - negl(\lambda)))^d \\
&= (1 - (\delta - negl(\lambda)))^{\frac{\lambda}{\delta}} \\
&\leq (1 - (\delta - negl(\lambda)))^{\frac{\lambda}{2(\delta - negl(\lambda))}} \\
&\leq e^{-\frac{\lambda}{2}}
\end{aligned}$$

which is negligible. Then, by the pseudorandomness of $\mathsf{G}(\mathsf{G}'(\cdot))$ (which comes from the pseudorandomness of G and G'), the probability that

$$(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d))) = (\tilde{\mathsf{C}}(\mathsf{G}(t_1)), \ldots, \tilde{\mathsf{C}}(\mathsf{G}(t_d)))$$

is also negligible. Finally, by the collision-resistance of $\mathcal{H}$, the probability that $\mathbf{Bad}_i$ occurs is also negligible.

This completes the proof of Claim 2. □

Claim 3. *If* iO *has correctness,* $\Pr[\mathcal{E}_1 = 1] = \Pr[\mathcal{E}_2 = 1]$.

Proof. Since the challenger uses an obfuscated version and an unobfuscated version of the circuit $\mathtt{Ext}$ to compute $(\tilde{x}, \tilde{y})$ in Game 1 and in Game 2 respectively, equivalence of $\Pr[\mathcal{E}_1 = 1]$ and $\Pr[\mathcal{E}_2 = 1]$ comes from the correctness of iO directly. □

Claim 4. *If* G' *is a pseudorandom generator, then* $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| \leq negl(\lambda)$.

Proof. In both Game 2 and Game 3, $\tilde{a}_1, \ldots, \tilde{a}_d$ are not used except generating $\tilde{t}_1, \ldots, \tilde{t}_d$. So, Claim 4 comes from the pseudorandomness of G' directly. □

Claim 5. *If* iO *is a secure indistinguishability obfuscator and* F *has correctness, then* $|\Pr[\mathcal{E}_3 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. As the only difference between Game 3 and Game 4 is that the adversary $\mathcal{A}$ gets obfuscations of different circuits in these two games, it is sufficient to prove that with all but negligible probability, all circuits $\mathcal{A}$ gets in Game 4 are functionally equivalent to their counterparts in Game 3. More concretely, we need to prove

$$\Pr[\mathtt{Ext}[mpk, K] \not\equiv \mathtt{Ext}^1[mpk, CK]] \leq negl(\lambda) \tag{3}$$

and for any $i \in [1, Q]$,

$$\Pr[\mathtt{M}[sk, K, H, k_i, msg_i] \not\equiv \mathtt{M}^1[sk, CK, H, k_i, msg_i, \alpha_i, \beta_i]] \leq negl(\lambda) \tag{4}$$

We start by proving Equation (3). For any input $(a_1, \ldots, a_d, b, ind, r)$, the two circuits $\mathtt{Ext}[mpk, K]$ and $\mathtt{Ext}^1[mpk, CK]$ compute differently only if $(\mathsf{G}'(a_1), \ldots, \mathsf{G}'(a_d)) = (\tilde{t}_1, \ldots, \tilde{t}_d)$. Such input exists with only a negligible probability since for any $j \in [1, d]$, $\tilde{t}_j$ is chosen uniformly at random from $\{0,1\}^l$ and the probability that it falls in the range of G', which contains only $2^{\frac{l}{2}}$ elements, is negligible.

Then we prove Equation (4). For any fixed $i \in [1, Q]$ and for any input x, let $\mu \| ind$ be the decryption of x, then we consider the following three cases:

1. **Case I:** $\mu = \alpha_i$ and $ind \leq msg_i$. In this case, the circuit $\mathtt{M}^1[sk, CK, H, k_i, msg_i, \alpha_i, \beta_i]$ will output $\mathsf{PRF}.\mathtt{Eval}(\beta_i, x)$ on input x, where $\beta_i = \mathsf{F}.\mathtt{Eval}(K, \mu)$. Besides, since $b_i = H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d)))$,

x could pass the check in step 2 of the circuit $\mathtt{M}[sk, K, H, k_i, msg_i]$. Thus, $\mathtt{M}[sk, K, H, k_i, msg_i]$ will also output $\mathsf{PRF}.\mathtt{Eval}(\mathsf{F}.\mathtt{Eval}(K, \mu), x)$ on input x. Therefore, $\mathtt{M}^{\mathtt{1}}[sk, CK, H, k_i, msg_i, \alpha_i, \beta_i](x) = \mathtt{M}[sk, K, H, k_i, msg_i](x)$ in this case.

2. **Case II:** $\mu \| ind \neq \perp$, $\mu \neq \alpha_i$, $ind \leq msg_i$, and $b = H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(t_d)))$, where $(t_1 \| \ldots \| t_d \| b) = \mu$. In this case, the circuit $\mathtt{M}^{\mathtt{1}}[sk, CK, H, k_i, msg_i, \alpha_i, \beta_i]$ will output $\mathsf{PRF}.\mathtt{Eval}(k'_1, x)$ on input x, where $k'_1 = \mathsf{F}.\mathtt{ConstrainEval}(CK, \mu)$. Also, we have

$$t_1 \| \ldots \| t_d \neq \tilde{t}_1 \| \ldots \| \tilde{t}_d$$

This is because otherwise, we will have

$$\begin{aligned} b &= H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(t_d))) \\ &= H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d))) \\ &= b_i \end{aligned}$$

which implies that $\mu = \alpha_i$ and contradicts the condition that $\mu \neq \alpha_i$. As a result, we have $k'_1 = \mathsf{F}.\mathtt{Eval}(K, \mu)$. Moreover, the circuit $\mathtt{M}[sk, K, H, k_i, msg_i]$ will also output $\mathsf{PRF}.\mathtt{Eval}(\mathsf{F}.\mathtt{Eval}(K, \mu), x)$ on input x. Therefore, $\mathtt{M}^{\mathtt{1}}[sk, CK, H, k_i, msg_i, \alpha_i, \beta_i](x) = \mathtt{M}[sk, K, H, k_i, msg_i](x)$ in this case.

3. **Case III:** Otherwise, in both circuits, the output will be $\mathsf{PRF}.\mathtt{Eval}(k_i, x)$.

In summary, for any $i \in [1, Q]$ and for any input, the output of the circuit $\mathtt{M}[sk, K, H, k_i, msg_i]$ and that of the circuit $\mathtt{M}^{\mathtt{1}}[sk, CK, H, k_i, msg_i, \alpha_i, \beta_i]$ are identical and thus Equation (4) follows.

This completes the proof of Claim 5. □

Claim 6. *If* F *is a secure prefix puncturable PRF, then* $|\Pr[\mathcal{E}_4 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$.

Proof. Indistinguishability between Game 4 and Game 5 comes from the puncturable pseudorandomness of F directly. □

Claim 7. *If* PRF *has pseudorandomness, then* $|\Pr[\mathcal{E}_5 = 1] - \Pr[\mathcal{E}_6 = 1]| \leq negl(\lambda)$.

Proof. Since in step 3 of Game 5 (and Game 6), the experiment aborts and outputs 2 if there exists $i \in [1, Q]$ that $\tilde{b} = b_i$, we have for any $i \in [1, Q]$, $\tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \neq \alpha_i$ in case the experiment does not abort. Thus, $\tilde{k}'$ will be sampled freshly and uniformly at random in Game 5 (and Game 6). Consequently, indistinguishability between Game 5 and Game 6 comes from the pseudorandomness of PRF directly. □

Claim 8. $\Pr[\mathcal{E}_6 = 1] \leq negl(\lambda)$.

Proof. In Game 6, $\tilde{y}$ is sampled uniformly and is independent of the view of $\mathcal{A}$. Thus, the probability that $\mathcal{A}$ succeeds in guessing $\tilde{y}$ in advance is negligible, which implies that the probability that $\mathcal{A}$ wins in Game 6 is negligible. □

Combining Claim 2 to Claim 8, we can conclude that the probability that $\mathcal{A}$ wins in Game 0 (i.e., in the real experiment $\mathtt{AUF}$) is also negligible. This completes the proof of Lemma C.3.

C.3 Proof of Unremovability

In this section, we prove the collusion resistant unremovability of WM. We start with defining a few auxiliary lemmas (Lemma C.4 to Lemma C.6) in Appendix C.3.1. Then we prove the collusion resistant unremovability of WM using these lemmas in Appendix C.3.2. Finally, we prove Lemma C.4 to Lemma C.6 in Appendix C.3.3 to Appendix C.3.5 respectively.

C.3.1 Definitions of Auxiliary Lemmas Before describing these lemmas, we first define the map ρ from $[1, 2^\kappa - 1]$ to $\{1, msg_1^*, \ldots, msg_Q^*, 2^\kappa - 1\}$:

$$\rho(ind) = \begin{cases} 1 & \text{if } ind \leq msg_1^*. \\ 2^\kappa - 1 & \text{if } ind > msg_Q^*. \\ msg_i^* & \text{if } \exists i \in [2, Q], msg_{i-1}^* < ind \leq msg_i^*. \end{cases}$$

where $msg_1^*, \ldots, msg_Q^*$ are messages submitted to the challenge oracle in the collusion resistant unremovability experiment and satisfies $msg_1^* < \ldots < msg_Q^*$ [15]. Note that for any $ind \in [1, 2^\kappa - 1]$ and any $i \in [1, Q]$, $ind \leq msg_i^*$ iff $\rho(ind) \leq msg_i^*$. Also, $\rho(\rho(ind)) = \rho(ind)$.

Next, we state the auxiliary lemmas in Lemma C.4 to Lemma C.6 as follows. Note that the messages $msg_1^*, \ldots, msg_Q^*$ can divide the whole message space $[1, 2^\kappa - 1]$ into $Q + 1$ sets, namely, $[1, msg_1^*], (msg_1^*, msg_2^*], \ldots, (msg_{Q-1}^*, msg_Q^*], (msg_Q^*, 2^\kappa - 1]$. Then, Lemma C.4 indicates that it is hard to distinguish punctured points labeled with messages from the same set; Lemma C.5 claims pseudorandomness of punctured points labeled with messages from the last set $(msg_Q^*, 2^\kappa - 1]$; and Lemma C.6 claims pseudorandomness of punctured points labeled with messages from the first set $[1, msg_1^*]$.

Lemma C.4. *If PFE is a secure puncturable functional encryption, PRF has correctness, constrained one-wayness and weak key-injectivity, G and G' are pseudorandom generators, $\mathcal{H}$ is a family of collision-resistant hash function, and iO is a secure indistinguishability obfuscator, then for any PPT unremoving-admissible adversary $\mathcal{A}$, the probability that $\mathtt{AUR}^1_{\mathcal{A}}(1^\lambda) = 1$ is negligibly close to $1/2$, where $\mathtt{AUR}^1$ is defined as follows:*

1. *In the beginning of the experiment, the challenger first samples $H \xleftarrow{\$} \mathcal{H}$ and generates $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then it generates $(mpk, msk) \leftarrow \mathsf{PFE}.\mathtt{Setup}(1^\lambda)$ and $sk \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, \mathtt{ID})$ for the identity function $\mathtt{ID}$. Next, it computes $\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}[mpk, K])$, and returns $EK = (H, \mathtt{E})$ to $\mathcal{A}$.*

[15] In this section, we always suppose $msg_1^* < \ldots < msg_Q^*$.

2. *Next, the challenger answers marking oracle queries from $\mathcal{A}$ and on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k_\iota, msg_\iota])$.*
3. *Once $\mathcal{A}$ makes a challenge oracle query with Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger samples $k^* \leftarrow \mathsf{PRF}.\mathtt{KeyGen}(1^\lambda)$, generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.*
4. *Then, the challenger answers marking oracle queries from $\mathcal{A}$ in the same way as in step 2.*
5. *Next, $\mathcal{A}$ will output a circuit $\tilde{\mathtt{C}}$ together with an integer $ind^* \in [1, 2^\kappa)$, and then the challenger works as follows:*
   (a) *Sample $\tilde{a}_1, \ldots, \tilde{a}_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$ and $r \xleftarrow{\$} \mathcal{R}$.*
   (b) *$\tilde{t}_1 = \mathsf{G}'(\tilde{a}_1), \ldots, \tilde{t}_d = \mathsf{G}'(\tilde{a}_d)$.*
   (c) *$\tilde{b} = H(\tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_1)), \ldots, \tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_d)))$.*
   (d) *$(\tilde{x}_0, \tilde{y}_0) = \mathtt{E}(\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{b}, ind^*, r)$.*
   (e) *$(\tilde{x}_1, \tilde{y}_1) = \mathtt{E}(\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{b}, \rho(ind^*), r)$.*
   (f) *$\beta \xleftarrow{\$} \{0,1\}$.*
   (g) *Return $\tilde{x}_\beta$ to $\mathcal{A}$.*
6. *Finally, $\mathcal{A}$ outputs a bit β'. Then, the challenger outputs 1 if $\beta = \beta'$ and outputs 0 otherwise.*

Lemma C.5. *If PFE is a secure puncturable functional encryption, PRF has constrained one-wayness and weak key-injectivity, G and G' are pseudorandom generators, $\mathcal{H}$ is a family of collision-resistant hash function, and iO is a secure indistinguishability obfuscator, then for any PPT unremoving-admissible adversary $\mathcal{A}$, the probability that $\mathtt{AUR}^2_{\mathcal{A}}(1^\lambda) = 1$ is negligibly close to $1/2$, where $\mathtt{AUR}^2$ is defined as follows:*

1. *In the beginning of the experiment, the challenger first samples $H \xleftarrow{\$} \mathcal{H}$ and generates $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then it generates $(mpk, msk) \leftarrow \mathsf{PFE}.\mathtt{Setup}(1^\lambda)$ and $sk \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, \mathtt{ID})$ for the identity function $\mathtt{ID}$. Next, it computes $\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}[mpk, K])$, and returns $EK = (H, \mathtt{E})$ to $\mathcal{A}$.*
2. *Next, the challenger answers marking oracle queries from $\mathcal{A}$ and on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k_\iota, msg_\iota])$.*
3. *Once $\mathcal{A}$ makes a challenge oracle query with Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger samples $k^* \leftarrow \mathsf{PRF}.\mathtt{KeyGen}(1^\lambda)$, generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.*
4. *Then, the challenger answers marking oracle queries from $\mathcal{A}$ in the same way as in step 2.*
5. *Next, $\mathcal{A}$ will output a circuit $\tilde{\mathtt{C}}$ together with an integer $ind^* \in (msg_Q^*, 2^\kappa)$[16], and then the challenger works as follows:*
   (a) *Sample $\tilde{a}_1, \ldots, \tilde{a}_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$ and $r \xleftarrow{\$} \mathcal{R}$.*
   (b) *$\tilde{t}_1 = \mathsf{G}'(\tilde{a}_1), \ldots, \tilde{t}_d = \mathsf{G}'(\tilde{a}_d)$.*
   (c) *$\tilde{b} = H(\tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_1)), \ldots, \tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_d)))$.*

[16] Here we suppose $msg_Q^* < 2^\kappa - 1$ as otherwise Lemma C.5 holds trivially

(d) $(\tilde{x}_0, \tilde{y}_0) = \mathtt{E}(\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{b}, ind^*, r)$.
(e) $\tilde{x}_1 \xleftarrow{\$} \{0,1\}^n$.
(f) $k' = \mathsf{F}.\mathtt{Eval}(K, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b})$
(g) $\tilde{y}_1 = \mathsf{PRF}.\mathtt{Eval}(k', \tilde{x}_1)$
(h) $\beta \xleftarrow{\$} \{0,1\}$.
(i) Return $(\tilde{x}_\beta, \tilde{y}_\beta)$ to $\mathcal{A}$.

6. Finally, $\mathcal{A}$ outputs a bit β'. Then, the challenger outputs 1 if $\beta = \beta'$ and outputs 0 otherwise.

Lemma C.6. *If* PFE *is a secure puncturable functional encryption,* PRF *is a secure puncturable PRF with constrained one-wayness and weak key-injectivity,* F *is a secure prefix puncturable PRF,* G *and* G' *are pseudorandom generators,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then for any PPT unremoving-admissible adversary* $\mathcal{A}$*, the probability that* $\mathtt{AUR}^3_{\mathcal{A}}(1^\lambda) = 1$ *is negligibly close to* $1/2$*, where* $\mathtt{AUR}^3$ *is defined as follows:*

1. *In the beginning of the experiment, the challenger first samples* $H \xleftarrow{\$} \mathcal{H}$ *and generates* $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$*. Then it generates* $(mpk, msk) \leftarrow \mathsf{PFE}.\mathtt{Setup}(1^\lambda)$ *and* $sk \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, \mathtt{ID})$ *for the identity function* $\mathtt{ID}$*. Next, it computes* $\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}[mpk, K])$*, and returns* $EK = (H, \mathtt{E})$ *to* $\mathcal{A}$*.*
2. *Next, the challenger answers marking oracle queries from* $\mathcal{A}$ *and on receiving a query* (k_ι, msg_ι) *(for the* ι*th marking oracle query), it returns* $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k_\iota, msg_\iota])$*.*
3. *Once* $\mathcal{A}$ *makes a challenge oracle query with* Q *messages* $\{msg_1^*, \ldots, msg_Q^*\}$*, the challenger samples* $k^* \leftarrow \mathsf{PRF}.\mathtt{KeyGen}(1^\lambda)$*, generates* $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k^*, msg_i^*])$ *for* $i \in [1, Q]$ *and returns* $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ *back.*
4. *Then, the challenger answers marking oracle queries from* $\mathcal{A}$ *in the same way as in step 2.*
5. *Next,* $\mathcal{A}$ *will output a circuit* $\tilde{\mathtt{C}}$ *together with an integer* $ind^* \in [1, msg_1^*]$*, and then the challenger works as follows:*
   (a) *Sample* $\tilde{a}_1, \ldots, \tilde{a}_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$ *and* $r \xleftarrow{\$} \mathcal{R}$*.*
   (b) $\tilde{t}_1 = \mathsf{G}'(\tilde{a}_1), \ldots, \tilde{t}_d = \mathsf{G}'(\tilde{a}_d)$*.*
   (c) $\tilde{b} = H(\tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_1)), \ldots, \tilde{\mathtt{C}}(\mathsf{G}(\tilde{t}_d)))$*.*
   (d) $(\tilde{x}_0, \tilde{y}_0) = \mathtt{E}(\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{b}, ind^*, r)$*.*
   (e) $\tilde{x}_1 \xleftarrow{\$} \{0,1\}^n$*.*
   (f) $\beta \xleftarrow{\$} \{0,1\}$*.*
   (g) *Return* $\tilde{x}_\beta$ *to* $\mathcal{A}$*.*
6. *Finally,* $\mathcal{A}$ *outputs a bit* β'*. Then, the challenger outputs 1 if* $\beta = \beta'$ *and outputs 0 otherwise.*

Remark C.2. Note that in both experiment $\mathtt{AUR}^1$ and experiment $\mathtt{AUR}^3$, the challenge $\tilde{x}_\beta$ could be generated publicly, so the adversary $\mathcal{A}$ can not win in these two games even multiple challenges are given. Unfortunately, in experiment $\mathtt{AUR}^2$, K is needed to generate the challenge $(\tilde{x}_\beta, \tilde{y}_\beta)$. However, as in the proof of Lemma C.5, security of F is not used, the indistinguishability holds even K is public. So, in experiment $\mathtt{AUR}^2$, multiple challenges also can not increase $\mathcal{A}$'s advantage noticeably. In summary, all three lemmas hold in the multiple challenges settings.

C.3.2 Unremovability from Lemma C.4 to Lemma C.6 In this section, we prove collusion resistant unremovability of WM based on these three lemmas defined in Appendix C.3.1.

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

- *Game 0.* This is the real experiment $\mathtt{ExptUR}_{\mathcal{A},q}$. In more detail, the challenger proceeds as follows:
  1. In the beginning, the challenger first samples $H \xleftarrow{\$} \mathcal{H}$ and generates $K \leftarrow \mathsf{F}.\mathtt{KeyGen}(1^\lambda)$. Then it generates $(mpk, msk) \leftarrow \mathsf{PFE}.\mathtt{Setup}(1^\lambda)$ and $sk \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, \mathtt{ID})$ for the identity function $\mathtt{ID}$. Next, it computes $\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}[mpk, K])$, and returns $EK = (H, \mathtt{E})$ to $\mathcal{A}$.
  2. Next, the challenger answers marking oracle queries from $\mathcal{A}$ and on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k_\iota, msg_\iota])$.
  3. Once $\mathcal{A}$ makes a challenge oracle query with Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger samples $k^* \leftarrow \mathsf{PRF}.\mathtt{KeyGen}(1^\lambda)$ generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.
  4. Then, the challenger answers marking oracle queries from $\mathcal{A}$ in the same way as in step 2.
  5. Finally, on input a circuit $\tilde{\mathtt{C}}$, the challenger first computes $\epsilon = 1/((\kappa+1)\cdot q+1)$, $T = \lambda/\epsilon^2$, and $S = q\cdot(\kappa+1)$, and sets a variable $counter = 0$. Next, it computes $\mathcal{L} = \mathtt{Trace}(0, 2^\kappa, 1, 0, \epsilon, T, \mathtt{E}, H, \tilde{\mathtt{C}})$. During this procedure, the challenger also maintains the variable *counter* and increase it by 1 each time the function $\mathtt{Test}(\cdot)$ is invoked. The challenger aborts once *counter* exceeds S. Finally,
     (a) If $q \geq Q$: then the challenger outputs 0 if it does not abort, $\mathcal{L} \subseteq \{msg_1^*, \ldots, msg_Q^*\}$ and $\mathcal{L} \neq \emptyset$. Otherwise, it outputs 1.
     (b) If $q < Q$: then the challenger outputs 0 if it aborts or if $\mathcal{L} \subseteq \{msg_1^*, \ldots, msg_Q^*\}$. Otherwise, it outputs 1.
- *Game 1.* This is identical to Game 0 except that for $ind \leq msg_Q^*$, the challenger uses $\mathtt{Test}^1(ind, -, -, -, \tilde{\mathtt{C}}, \mathtt{C}_Q^*)$ instead of $\mathtt{Test}(ind, -, -, -, \tilde{\mathtt{C}})$ in the $\mathtt{Trace}$ algorithm, where $\mathtt{Test}^1$ is defined in Figure 5.
- *Game 2.* This is identical to Game 1 except that for $ind \leq msg_1^*$, the challenger uses $\mathtt{Test}^2(ind, -, -, -, \tilde{\mathtt{C}}, \mathtt{C}_Q^*)$ instead of $\mathtt{Test}^1(ind, -, -, -, \tilde{\mathtt{C}}, \mathtt{C}_Q^*)$ in the $\mathtt{Trace}$ algorithm, where $\mathtt{Test}^2$ is defined in Figure 5.
- *Game 3.* This is identical to Game 2 except that for $ind \leq msg_1^*$, instead of computing $p_3 = \mathtt{Test}^2(ind, -, -, -, \tilde{\mathtt{C}}, \mathtt{C}_Q^*)$ in the $\mathtt{Trace}$ algorithm, the challenger sets p_3 to be 1 directly.
- *Game 4* This is identical to Game 3 except that for $msg_1^* < ind \leq msg_Q^*$, the challenger uses $\mathtt{Test}^1(\rho(ind), -, -, -, \tilde{\mathtt{C}}, \mathtt{C}_Q^*)$ instead of $\mathtt{Test}^1(ind, -, -, -, \tilde{\mathtt{C}}, \mathtt{C}_Q^*)$ in the $\mathtt{Trace}$ algorithm.
- *Game 5.* This is identical to Game 4 except that for $ind > msg_Q^*$, the challenger uses $\mathtt{Test}^3(ind, -, -, -, \tilde{\mathtt{C}}, K)$ instead of $\mathtt{Test}(ind, -, -, -, \tilde{\mathtt{C}})$ in the $\mathtt{Trace}$ algorithm, where $\mathtt{Test}^3$ is defined in Figure 5.

- *Game 6.* This is identical to Game 5 except that for $ind > msg^*_Q$, the challenger uses $\texttt{Test}^3(ind, -, -, -, \mathsf{PRF}.\texttt{Eval}(k^*, \cdot), K)$ instead of $\texttt{Test}^3(ind, -, -, -, \tilde{\mathsf{C}}, K)$ in the `Trace` algorithm.
- *Game 7.* This is identical to Game 6 except that for $ind > msg^*_Q$, the challenger uses $\texttt{Test}^4(ind, -, -, -, \mathsf{PRF}.\texttt{Eval}(k^*, \cdot), k^*, K)$ instead of $\texttt{Test}^3(ind, -, -, -, \mathsf{PRF}.\texttt{Eval}(k^*, \cdot), K)$ in the `Trace` algorithm, where $\texttt{Test}^4$ is defined in Figure 5.
- *Game 8.* This is identical to Game 7 except that for $ind > msg^*_Q$, instead of computing $p_3 = \texttt{Test}^4(ind, -, -, -, \mathsf{PRF}.\texttt{Eval}(k^*, \cdot), k^*, K)$ in the `Trace` algorithm, the challenger sets p_3 to be 0 directly.

$\texttt{Test}$

Input: $ind, T, \mathsf{E}, H, \mathsf{C}$

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) Sample $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$, $r \xleftarrow{\$} \mathcal{R}$.
   (b) $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$.
   (c) $b = H(\mathsf{C}(\mathsf{G}(t_1)), \ldots, \mathsf{C}(\mathsf{G}(t_d)))$.
   (d) $(x, y) = \mathsf{E}(a_1, \ldots, a_d, b, ind, r)$.
   (e) If $\mathsf{C}(x) = y$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

$\texttt{Test}^1$

Input: $ind, T, \mathsf{E}, H, \mathsf{C}, \underline{\mathsf{C}'}$

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) Sample $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$, $r \xleftarrow{\$} \mathcal{R}$.
   (b) $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$.
   (c) $b = H(\mathsf{C}(\mathsf{G}(t_1)), \ldots, \mathsf{C}(\mathsf{G}(t_d)))$.
   (d) $(x, y) = \mathsf{E}(a_1, \ldots, a_d, b, ind, r)$.
   (e) If $\underline{\mathsf{C}(x) = \mathsf{C}'(x)}$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

$\texttt{Test}^2$

Input: $ind, T, \mathsf{E}, H, \mathsf{C}, \mathsf{C}'$

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) $\underline{x \xleftarrow{\$} \{0,1\}^n}$.
   (b) If $\mathsf{C}(x) = \mathsf{C}'(x)$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

$\texttt{Test}^3$

Input: $ind, T, \mathsf{E}, H, \mathsf{C}, \underline{K}$

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) Sample $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$
   (b) $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$.
   (c) $b = H(\mathsf{C}(\mathsf{G}(t_1)), \ldots, \mathsf{C}(\mathsf{G}(t_d)))$.
   (d) $\underline{x \xleftarrow{\$} \{0,1\}^n}$.
   (e) $\underline{k' = \mathsf{F}.\texttt{Eval}(K, t_1 \| \ldots \| t_d \| b)}$.
   (f) $\underline{y = \mathsf{PRF}.\texttt{Eval}(k', x)}$.
   (g) If $\mathsf{C}(x) = y$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

$\texttt{Test}^4$

Input: $ind, T, \mathsf{E}, H, \mathsf{C}, \underline{k}, K$

1. $Acc = 0$
2. For $i \in [1, T]$:
   (a) Sample $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$
   (b) $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$.
   (c) $b = H(\mathsf{C}(\mathsf{G}(t_1)), \ldots, \mathsf{C}(\mathsf{G}(t_d)))$.
   (d) $x \xleftarrow{\$} \{0,1\}^n$.
   (e) $k' = \mathsf{F}.\texttt{Eval}(K, t_1 \| \ldots \| t_d \| b)$.
   (f) $\underline{\text{If } k = k' \text{: return } 0.}$
   (g) $y = \mathsf{PRF}.\texttt{Eval}(k', x)$.
   (h) If $\mathsf{C}(x) = y$: $Acc = Acc + 1$.
3. Return $\frac{Acc}{T}$.

Fig. 5 Variants of function `Test` that are used in Game 0 to Game 8 defined above. For completeness, we also include the original function `Test` here.

Next, we prove the indistinguishability of each consecutive pair of games and show that the adversary $\mathcal{A}$ will win in the final game (Game 8) with a negligible probability. For simplicity of notation, we use $\mathcal{E}_i$ to denote the output of Game i.

Claim 9. *If both* G *and* G' *are pseudorandom generators,* iO *has correctness and* PFE *has sparseness and correctness, then* $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. For any $ind \leq msg_Q^*$ that $\mathtt{Test}(ind, -, -, -, \tilde{\mathsf{C}})$ is called in step 5 of Game 0, let $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}$, $r \xleftarrow{\$} \mathcal{R}$, $t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d)$, $b = H(\tilde{\mathsf{C}}(\mathsf{G}(t_1)), \ldots, \tilde{\mathsf{C}}(\mathsf{G}(t_d)))$ and $(x, y) = \mathtt{E}(a_1, \ldots, a_d, b, ind, r)$. Also, let $\mu = t_1 \| \ldots \| t_d \| b$. It is sufficient to prove that $\mathtt{C}_Q^*(x) = y$ with all but negligible probability.

First, by the unremoving-admissibility of $\mathcal{A}$, the sparseness of PFE, and the correctness of iO, $\tilde{\mathsf{C}}$ differs with $\mathsf{PRF}.\mathtt{Eval}(k^*, \cdot)$ on only a negligible fraction of inputs. Then, by the pseudorandomness of G and G', which implies the pseudorandomness of $\mathsf{G}(\mathsf{G}'(\cdot))$, $b = H(\mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(t_d)))$ with all but negligible probability. Thus, by the correctness of PFE and iO, $\mathtt{C}_Q^*(x) = \mathsf{PRF}.\mathtt{Eval}(\mathsf{F}.\mathtt{Eval}(K, \mu), x)$. Besides, by the correctness of iO, we also have $y = \mathsf{PRF}.\mathtt{Eval}(\mathsf{F}.\mathtt{Eval}(K, \mu), x)$. This completes the proof of Claim 9. □

Claim 10. *If Lemma C.6 holds, then* $|\Pr[\mathcal{E}_1 = 1] - \Pr[\mathcal{E}_2 = 1]| \leq negl(\lambda)$.

Proof. Claim 10 comes from the multiple challenges version of Lemma C.6 by a direct reduction, where the latter is implied by Lemma C.6 as we have discussed in Remark C.2. □

Claim 11. *If* PFE *has sparseness and* iO *has correctness, then* $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| \leq negl(\lambda)$.

Proof. First, by sparseness of PFE and correctness of iO, for any $i \in [1, Q]$, $\mathtt{C}_i^*$ differs with $\mathtt{C}_Q^*$ on only a negligible fraction of inputs. Then, by the unremoving-admissibility of $\mathcal{A}$, $\tilde{\mathsf{C}}$ differs with $\mathtt{C}_Q^*$ on only a negligible fraction of inputs. Therefore, for a randomly sampled x, $\tilde{\mathsf{C}}(x) \neq \mathtt{C}_Q^*(x)$ with only a negligible probability. Finally, by the union bound, with all but negligible probability, for all $ind \leq msg_1^*$ that $\mathtt{Test}^2(ind, -, -, -, \tilde{\mathsf{C}}, \mathtt{C}_Q^*)$ is invoked in step 5 of Game 2, we have $\mathtt{Test}^2(ind, -, -, -, \tilde{\mathsf{C}}, \mathtt{C}_Q^*) = 1$. This completes the proof of Claim 11. □

Claim 12. *If Lemma C.4 holds, then* $|\Pr[\mathcal{E}_3 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. Claim 12 comes from the multiple challenges version of Lemma C.4 by a direct reduction, where the latter is implied by Lemma C.4 as we have discussed in Remark C.2. □

Claim 13. *If Lemma C.5 holds, then* $|\Pr[\mathcal{E}_4 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$.

Proof. Claim 13 comes from the multiple challenges version of Lemma C.5 by a direct reduction, where the latter is implied by Lemma C.5 as we have discussed in Remark C.2. □

Claim 14. *If both* G *and* G' *are pseudorandom generators,* PFE *has sparseness,* iO *has correctness, then* $|\Pr[\mathcal{E}_5 = 1] - \Pr[\mathcal{E}_6 = 1]| \leq negl(\lambda)$.

Proof. First, by the unremoving-admissibility of $\mathcal{A}$, sparseness of PFE and correctness of iO, $\tilde{\mathsf{C}}$ differs with $\mathsf{PRF}.\mathtt{Eval}(k^*,\cdot)$ on only a negligible fraction of inputs. Thus, for a random or pseudorandom input x, $\tilde{\mathsf{C}}(x) \neq \mathsf{PRF}.\mathtt{Eval}(k^*, x)$ with only a negligible probability. Therefore, with all but negligible probability, replacing $\tilde{\mathsf{C}}$ with $\mathsf{PRF}.\mathtt{Eval}(k^*,\cdot)$ will not change the output of the $\mathtt{Test}$ algorithm. □

Claim 15. *If* F *has pseudorandomness, then* $|\Pr[\mathcal{E}_6 = 1] - \Pr[\mathcal{E}_7 = 1]| \leq negl(\lambda)$.

Proof. Let $a_1, \ldots, a_d \xleftarrow{\$} \{0,1\}^{\frac{l}{2}}, t_1 = \mathsf{G}'(a_1), \ldots, t_d = \mathsf{G}'(a_d), b = H(\mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(t_d)))$ and $k' = \mathsf{F}.\mathtt{Eval}(K, t_1\| \ldots \|t_d\|b)$. It is sufficient to prove that $k^* = k'$ with only a negligible probability.

This comes from the pseudorandomness of F directly since both $t_1\| \ldots \|t_d\|b$ and k^* can be computed without knowing the secret key K of F. □

Claim 16. *If* PRF *has weak key-injectivity, then* $|\Pr[\mathcal{E}_7 = 1] - \Pr[\mathcal{E}_8 = 1]| \leq negl(\lambda)$.

Proof. It is sufficient to prove that $\mathtt{Test}^4(ind, -, -, -, \mathsf{PRF}.\mathtt{Eval}(k^*,\cdot), k^*, K)$ equals to 0 with all but negligible probability for any $ind > msg_Q^*$ that $\mathtt{Test}^4(ind, -, -, -, \mathsf{PRF}.\mathtt{Eval}(k^*,\cdot), k^*, K)$ is invoked.

We consider two cases. First, if the check in step 2.(f) is validated during the execution of the algorithm, then the function will output 0. Otherwise, by the weak key-injectivity of PRF, with all but negligible probability, the variable Acc will never increase at step 2.(h), so the output will also be 0.

This completes the proof of Claim 16. □

Claim 17. $\Pr[\mathcal{E}_8 = 1] \leq negl(\lambda)$.

Proof. Now, in Game 8, for any invoked interval $[ind_1, ind_2)$ (recall that we use an interval $[ind_1, ind_2)$ to denote the invocation of the algorithm $\mathsf{Trace}(ind_1, ind_2, -, -, -, -, -, -, -)$) that does not contain any message submitted to the challenge oracle, it must belong to one of the following three cases:

1. $0 \leq ind_1 < ind_2 \leq msg_1^*$. In this case, both $\mathfrak{p}_{ind_1}$ and $\mathfrak{p}_{ind_2}$ will be 1.
2. $\exists i \in [2, Q], msg_{i-1}^* < ind_1 < ind_2 \leq msg_i^*$. In this case, we have $\rho(ind_1) = \rho(ind_2)$, so $\mathfrak{p}_{ind_1}$ and $\mathfrak{p}_{ind_2}$ are outputs of the algorithm $\mathtt{Test}^1$ on the same input. Now, let p be the probability that the check in step 2.(e) of $\mathtt{Test}^1(\rho(ind_1), T, -, -, -, -)$ can be validated in one iteration and $\mathfrak{Acc}$ be a random variable for the final value of variable Acc when running $\mathtt{Test}^1(\rho(ind_1), T, -, -, -, -)$. Then by the chernoff bounds, we have

$$Pr[\mathfrak{Acc} \geq (1 + \frac{\epsilon}{2p}) \cdot p \cdot T] \leq e^{-\frac{\epsilon^2}{8p+2\epsilon}T} = e^{-\frac{\lambda}{8p+2\epsilon}}$$

$$Pr[\mathfrak{Acc} \leq (1 - \frac{\epsilon}{2p}) \cdot p \cdot T] \leq e^{-\frac{\epsilon^2}{8p}T} = e^{-\frac{\lambda}{8p}}$$

Since both p and ϵ do not exceed 1, both probabilities are negligible. Therefore, we have $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| \leq \epsilon$ with all but negligible probability.

3. $msg^*_Q < ind_1 < ind_2 \leq 2^\kappa$. In this case, both $\mathfrak{p}_{ind_1}$ and $\mathfrak{p}_{ind_2}$ will be 0.

In all three cases, we have $|\mathfrak{p}_{ind_1} - \mathfrak{p}_{ind_2}| \leq \epsilon$ with all but negligible probability. Thus, if $q \geq Q$, by Lemma C.1, with all but negligible probability, the challenger can obtain the set $\mathcal{L}$ by invoking the $\mathtt{Test}$ algorithm no more than $q \cdot (\kappa + 1)$ times and $\mathcal{L}$ is a non-empty subset of the set $\{msg^*_1, \ldots, msg^*_Q\}$. On the other hand, if $q < Q$, by Lemma C.2, with all but negligible probability, the challenger can obtain a subset of $\{msg^*_1, \ldots, msg^*_Q\}$ if it does not abort. In summary, in both cases, namely, the case $q \geq Q$ and the case $q < Q$, the challenger will output 0 with all but negligible probability. This completes the proof of Claim 17. □

Combining Claim 9 to Claim 17, we can conclude that the probability that $\mathcal{A}$ wins in Game 0 (i.e., in the real experiment $\mathtt{ExptUR}$) is also negligible. This completes the proof of collusion resistant unremovability.

C.3.3 Proof of Lemma C.4 In this section, we prove Lemma C.4. First, we define the following games between a challenger and a PPT unremoving-admissible adversary $\mathcal{A}$:

- *Game 0.* This is the real experiment $\mathtt{AUR}^1$.
- *Game 1.* In Game 1, the challenger changes the way for generating variables in step 5. In particular, in step 5, on receiving the circuit $\tilde{\mathtt{C}}$ and ind^*, the challenger proceeds as follows:
  1. Sample $\tilde{t}_1, \ldots, \tilde{t}_d \xleftarrow{\$} \{0,1\}^l$.
  2. $\tilde{b} = H(\mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_d)))$.
  3. $\tilde{x}_0 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \| ind^*)$
  4. $\tilde{x}_1 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \| \rho(ind^*))$
  5. $\beta \xleftarrow{\$} \{0,1\}$.
  6. Return $\tilde{x}_\beta$ to $\mathcal{A}$.
- *Game 2.* This is identical to Game 1 except that the challenger samples $k^*, \tilde{t}_1, \ldots, \tilde{t}_d$ and computes $\tilde{b}$ immediately after it generates H, K, mpk, sk in step 1.
- *Game 3.* This is identical to Game 2 except that the challenger further checks if k^* has been submitted to the marking oracle. In particular, in step 2 and in step 4, on receiving a query (k_ι, msg_ι), the challenger checks if $k_\iota = k^*$. It aborts and outputs 2 if this is the case; otherwise, it proceeds identically as in Game 2.
- *Game 4.* This is identical to Game 3 except that the challenger changes the way to answer the marking oracle and the challenge oracle. In particular, it generates $sk_1 \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, f_1)$ and $sk_2 \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, f_2)$ in step 2 and step 3 respectively, where f_1 and f_2 is defined as:

$$f_1(t_1 \| \ldots \| t_d \| b \| ind) = \begin{cases} t_1 \| \ldots \| t_d \| b \| 0 & \text{if } t_1 \| \ldots \| t_d \| b = \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \\ t_1 \| \ldots \| t_d \| b \| ind & \text{otherwise} \end{cases}$$

$$f_2(t_1\|\ldots\|t_d\|b\|ind) = \begin{cases} t_1\|\ldots\|t_d\|b\|\rho(ind) & \text{if } t_1\|\ldots\|t_d\|b = \tilde{t}_1\|\ldots\|\tilde{t}_d\|\tilde{b} \\ & \wedge ind \in [1, 2^\kappa) \\ t_1\|\ldots\|t_d\|b\|ind & \text{otherwise} \end{cases}$$

Then in step 2 and step 4, on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk_1, K, H, k_\iota, msg_\iota])$. Also, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk_2, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.

Next, we prove the indistinguishability of each consecutive pair of games and show that the adversary $\mathcal{A}$ will win in the final game (Game 4) with a negligible probability. For simplicity of notation, we use $\mathcal{E}_i$ to denote the output of Game i.

Claim 18. *If both* G *and* G' *are pseudorandom generators,* iO *has correctness and* PFE *has sparseness, then* $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_1 = 1]| \leq negl(\lambda)$.

Proof. To prove Claim 18, we further define the following auxiliary games:

- *Game* 0.1. This is identical to Game 0 except that the challenger samples $r_1 \xleftarrow{\$} \mathcal{R}$ and $r_2 \xleftarrow{\$} \mathcal{R}$ in the beginning of step 5. Then it computes $\tilde{x}_0$ and $\tilde{x}_1$ as follows:
$$(\tilde{x}_0, \tilde{y}_0) = \mathtt{E}(\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{b}, ind^*, r_1)$$
$$(\tilde{x}_1, \tilde{y}_1) = \mathtt{E}(\tilde{a}_1, \ldots, \tilde{a}_d, \tilde{b}, \rho(ind^*), r_2)$$
- *Game* 0.2. This is identical to Game 0.1 except that the challenger computes $\tilde{b}$ as follows:
$$\tilde{b} = H(\mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_d)))$$
- *Game* 0.3. This is identical to Game 0.2 except that the challenger computes $\tilde{x}_0$ and $\tilde{x}_1$ as follows:
$$\tilde{x}_0 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1\|\ldots\|\tilde{t}_d\|\tilde{b}\|ind^*)$$
$$\tilde{x}_1 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1\|\ldots\|\tilde{t}_d\|\tilde{b}\|\rho(ind^*))$$
- *Game* 0.4. This is identical to Game 0.3 except that the challenger samples $\tilde{t}_1 \xleftarrow{\$} \{0,1\}^l, \ldots, \tilde{t}_d \xleftarrow{\$} \{0,1\}^l$.

Next, we will argue the indistinguishability between each consecutive pair of games:

- **Game** 0 **and Game** 0.1**.** As either $\tilde{x}_0$ or $\tilde{x}_1$ will be hidden from $\mathcal{A}$, the changes in Game 0.1 are purely conceptual and will not affect the output of the game.

- **Game** 0.1 **and Game** 0.2**.** First, by the unremoving-admissibility of $\mathcal{A}$, the sparseness of PFE, and the correctness of iO, $\tilde{\mathtt{C}}$ differs with $\mathsf{PRF}.\mathtt{Eval}(k^*, \cdot)$ on only a negligible fraction of inputs. Then, by the pseudorandomness of G and G', which implies the pseudorandomness of $\mathsf{G}(\mathsf{G}'(\cdot))$, $\tilde{\mathtt{C}}(\mathsf{G}(t_j)) = \mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(t_j))$ with all but negligible probability for $j \in [1, d]$. Thus, by the union bound, with all but negligible probability, $\tilde{b}$ is identically computed in Game 0.1 and Game 0.2, and the indistinguishability between these two games follows.
- **Game** 0.2 **and Game** 0.3**.** Since the challenger uses an obfuscated version and an unobfuscated version of the circuit $\mathtt{Ext}$ to compute $\tilde{x}_0, \tilde{x}_1$ in Game 0.2 and in Game 0.3 respectively, equivalence of $\Pr[\mathcal{E}_{0.2} = 1]$ and $\Pr[\mathcal{E}_{0.3} = 1]$ comes from the correctness of iO directly.
- **Game** 0.3 **and Game** 0.4**.** As in both games, $\tilde{a}_1, \ldots, \tilde{a}_d$ are not used except generating $\tilde{t}_1, \ldots, \tilde{t}_d$, indistinguishability of Game 0.3 and Game 0.4 comes from the pseudorandomness of G' directly.
- **Game** 0.4 **and Game** 1**.** It is easy to verify that Game 0.4 and Game 1 are identical and the indistinguishability follows.

□

Claim 19. $\Pr[\mathcal{E}_1 = 1] = \Pr[\mathcal{E}_2 = 1]$.

Proof. Obviously, the changes in Game 2 are purely conceptual and will not affect the output of the game, thus, Claim 19 follows. □

Claim 20. *If* PRF *has correctness and constrained one-wayness, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_2 = 1] - \Pr[\mathcal{E}_3 = 1]| \leq negl(\lambda)$.

Proof. To prove Claim 20, we further define the following auxiliary games:

- *Game* 2.1. This is identical to Game 2 except that the challenger does not use k^* directly. More precisely, after sampling k^*, the challenger further generates $ck^* \leftarrow \mathsf{PRF}.\mathtt{Constrain}(k^*, 0^n)$ and computes $y_0 = \mathsf{PRF}.\mathtt{Eval}(k^*, 0^n)$. Then it uses $\mathsf{PRF}_{ck^*, y_0}(\cdot)$ instead of $\mathsf{PRF}.\mathtt{Eval}(k^*, \cdot)$ when computing $\tilde{b}$, where for any constrained key ck of PRF and for any $y \in \{0,1\}^m$, $\mathsf{PRF}_{ck,y}(\cdot)$ is defined as follows:

$$\mathsf{PRF}_{ck,y}(x) = \begin{cases} \mathsf{PRF}.\mathtt{ConstrainEval}(ck, x) & \text{If} x \neq 0^n \\ y & \text{Otherwise} \end{cases}$$

  Besides, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}^2[sk, K, H, ck^*, y_0, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back, where the circuit $\mathtt{M}^2$ is defined in Figure 6.
- *Game* 2.2. This is identical to Game 2.1 except that in step 2 and in step 4, on receiving a query (k_ι, msg_ι), the challenger checks if $k_\iota = k^*$. It aborts and outputs 2 if this is the case; otherwise, it proceeds identically as in Game 2.1.

$\mathtt{M}^2$

Constant: sk, K, H, ck, y, msg
Input: x

1. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$.
2. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}_{ck,y}(\mathsf{G}(t_1)), \ldots, \mathsf{PRF}_{ck,y}(\mathsf{G}(t_d))) = b)$
   (a) $k' = \mathsf{F}.\mathtt{Eval}(K, t_1 \| \ldots \| t_d \| b)$.
   (b) Output $\mathsf{PRF}.\mathtt{Eval}(k', x)$.
3. Otherwise, output $\mathsf{PRF}_{ck,y}(x)$.

Fig. 6 The circuit $\mathtt{M}^2$.

Next, we will argue the indistinguishability between each consecutive pair of games.

- **Game** 2 **and Game** 2.1**.** By the correctness of PRF,

$$\mathsf{PRF}.\mathtt{Eval}(k^*, \cdot) \equiv \mathsf{PRF}_{ck^*, y_0}(\cdot) \tag{5}$$

  so $\tilde{b}$ is identically computed in these two games. Equation (5) also implies that the two circuits $\mathtt{M}[sk, K, H, k^*, msg^*]$ and $\mathtt{M}^2[sk, K, H, ck^*, y_0, msg^*]$ are identically evaluated on all inputs. Then, by the indistinguishability of iO, the views of $\mathcal{A}$ in Game 2 and Game 2.1 are indistinguishable.
- **Game** 2.1 **and Game** 2.2**.** Game 2.1 and Game 2.2 are identically proceeded as long as for all k_ι submitted to the marking oracle, $k_\iota \neq k^*$. This occurs with all but negligible probability due to the constrained one-wayness of PRF.
- **Game** 2.2 **and Game** 3**.** Indistinguishability of Game 2.2 and Game 3 can be argued in a similar way as that of Game 2 and Game 2.1, and we just omit its details.

□

Claim 21. *If* PFE *has iO-compatible correctness,* PRF *has weak key-injectivity,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_3 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. As the only difference between Game 3 and Game 4 is that the adversary $\mathcal{A}$ gets obfuscations of different circuits in these two games, it is sufficient to prove that with all but negligible probability, all circuits $\mathcal{A}$ gets in Game 4 are functionally equivalent to their counterparts in Game 3. More concretely, let Q' be the number of marking oracle queries that is not aborted, we need to prove that:

$$\forall i \in [1, Q'], \Pr[\mathtt{M}[sk, K, H, k_i, msg_i] \not\equiv \mathtt{M}[sk_1, K, H, k_i, msg_i]] \leq negl(\lambda) \tag{6}$$

$$\forall i \in [1, Q], \Pr[\mathtt{M}[sk, K, H, k^*, msg_i^*] \not\equiv \mathtt{M}[sk_2, K, H, k^*, msg_i^*]] \leq negl(\lambda) \tag{7}$$

First, by the iO-compatible correctness of PFE, with all but negligible probability, we have

$$\forall x \in \mathcal{I}_{(mpk,msk)}, \mathsf{PFE}.\mathtt{Dec}(sk, x) = \mathsf{PFE}.\mathtt{Dec}(sk_1, x) = \mathsf{PFE}.\mathtt{Dec}(sk_2, x) = \perp$$

$$\forall x \in \mathcal{V}_{(mpk,msk)}, \mathsf{PFE}.\mathtt{Dec}(sk_1, x) = f_1(\mathsf{PFE}.\mathtt{Dec}(sk, x))$$

$$\forall x \in \mathcal{V}_{(mpk,msk)}, \mathsf{PFE}.\mathtt{Dec}(sk_2, x) = f_2(\mathsf{PFE}.\mathtt{Dec}(sk, x))$$

Moreover, as $k_i \neq k^*$ for $i \in [1, Q']$, then by the weak key-injectivity of PRF and the collision resistance of $\mathcal{H}$, we have

$$\tilde{b} \neq H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d)))$$

with all but negligible probability. Next, we suppose the above four equations hold, which occurs with all but negligible probability.

We start by proving Equation (6). For any $i \in [1, Q']$ and for any input x, let $\mu_1 \| ind_1 = \mathsf{PFE}.\mathtt{Dec}(sk, x)$ and $\mu_2 \| ind_2 = \mathsf{PFE}.\mathtt{Dec}(sk_1, x)$. Then we consider the following three cases:

1. **Case I:** $x \in \mathcal{I}_{(mpk,msk)}$. In this case, both $\mu_1 \| ind_1$ and $\mu_2 \| ind_2$ are $\perp$. Thus, both circuits will output $\mathsf{PRF}.\mathtt{Eval}(k_i, x)$.
2. **Case II:** $x \in \mathcal{V}_{(mpk,msk)}$ and $\mu_1 = \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b}$. In this case, we have $\mu_2 = \mu_1$. As $\tilde{b} \neq H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d)))$, we have both $\mathtt{M}[sk_1, K, H, k_i, msg_i](x) = \mathsf{PRF}.\mathtt{Eval}(k_i, x)$ and $\mathtt{M}[sk, K, H, k_i, msg_i](x) = \mathsf{PRF}.\mathtt{Eval}(k_i, x)$.
3. **Case III:** $x \in \mathcal{V}_{(mpk,msk)}$ and $\mu_1 \neq \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b}$. In this case, we have $\mu_2 \| ind_2 = \mu_1 \| ind_1$. Thus, the outputs of the two circuits are also equal on input x.

In summary, for any $i \in [1, Q']$ and for any input, the output of the circuit $\mathtt{M}[sk, K, H, k_i, msg_i]$ and that of the circuit $\mathtt{M}[sk_1, K, H, k_i, msg_i]$ are identical.

Then, we prove Equation (7). For any $i \in [1, Q]$ and for any input x, let $\mu_1 \| ind_1 = \mathsf{PFE}.\mathtt{Dec}(sk, x)$ and $\mu_2 \| ind_2 = \mathsf{PFE}.\mathtt{Dec}(sk_2, x)$. Then we consider the following three cases:

1. **Case I:** $x \in \mathcal{I}_{(mpk,msk)}$. In this case, both $\mu_1 \| ind_1$ and $\mu_2 \| ind_2$ are $\perp$. Thus, both circuits will output $\mathsf{PRF}.\mathtt{Eval}(k^*, x)$.
2. **Case II:** $x \in \mathcal{V}_{(mpk,msk)}$, $\mu_1 = \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b}$ and $ind \in [1, 2^\kappa)$. In this case, we have $\mu_2 = \mu_1$ and $ind_2 = \rho(ind_1)$. As $ind_1 \leq msg_i^*$ iff $\rho(ind_1) \leq msg_i^*$, the two circuits $\mathtt{M}[sk, K, H, k^*, msg_i^*]$ and $\mathtt{M}[sk_2, K, H, k^*, msg_i^*]$ are identically evaluated.
3. **Case III:** Otherwise, we have $\mu_2 \| ind_2 = \mu_1 \| ind_1$. Thus, the outputs of the two circuits are also equal on input x.

In summary, for any $i \in [1, Q]$ and for any input, the output of the circuit $\mathtt{M}[sk, K, H, k^*, msg_i^*]$ and that of the circuit $\mathtt{M}[sk_2, K, H, k^*, msg_i^*]$ are identical.

As with all but negligible probability, we have both Equation (6) and Equation (7) hold, Claim 21 follows. □

Claim 22. *If* PFE *is a secure puncturable functional encryption scheme, then* $\Pr[\mathcal{E}_4 = 1] \leq negl(\lambda)$.

Proof. Now, in the view of $\mathcal{A}$, only sk_1 and sk_2 are available and each of them decrypts identically on $\tilde{x}_0$ and $\tilde{x}_1$. Thus, by the adaptive indistinguishability of PFE, the adversary can win in Game 4 with only a negligible probability. □

Combining Claim 18 to Claim 22, we can conclude that the probability that $\mathcal{A}$ wins in Game 0 (i.e., in the real experiment $\mathtt{AUR}^1$) is also negligible. This completes the proof of Lemma C.4.

C.3.4 Proof of Lemma C.5 In this section, we prove Lemma C.5. First, we define the following games between a challenger and a PPT unremoving-admissible adversary $\mathcal{A}$:

- *Game 0.* This is the real experiment $\mathtt{AUR}^2$.
- *Game 1.* In Game 1, the challenger changes the way for generating variables in step 5. In particular, in step 5, on receiving the circuit $\tilde{\mathsf{C}}$ and ind^*, the challenger proceeds as follows:
  1. Sample $\tilde{t}_1, \ldots, \tilde{t}_d \xleftarrow{\$} \{0,1\}^l$.
  2. $\tilde{b} = H(\mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_d)))$.
  3. $\tilde{x}_0 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \| ind^*)$
  4. $\tilde{x}_1 \xleftarrow{\$} \{0,1\}^n$.
  5. $k' = \mathsf{F}.\mathtt{Eval}(K, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b})$
  6. $\tilde{y}_0 = \mathsf{PRF}.\mathtt{Eval}(k', \tilde{x}_0)$
  7. $\tilde{y}_1 = \mathsf{PRF}.\mathtt{Eval}(k', \tilde{x}_1)$
  8. $\beta \xleftarrow{\$} \{0,1\}$.
  9. Return $(\tilde{x}_\beta, \tilde{y}_\beta)$ to $\mathcal{A}$.
- *Game 2.* This is identical to Game 1 except that the challenger samples $k^*, \tilde{t}_1, \ldots, \tilde{t}_d$ and computes $\tilde{b}$ immediately after it generates H, K, mpk, sk in step 1.
- *Game 3.* This is identical to Game 2 except that the challenger further checks if k^* has been submitted to the marking oracle. In particular, in step 2 and in step 4, on receiving a query (k_ι, msg_ι), the challenger checks if $k_\iota = k^*$. It aborts and outputs 2 if this is the case; otherwise, it proceeds identically as in Game 2.
- *Game 4.* This is identical to Game 3 except that the challenger changes the way to answer the marking oracle and the challenge oracle. In particular, it generates $sk_1 \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, f_1)$ and $sk_2 \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, f_2)$ in step 2 and step 3 respectively, where f_1 and f_2 is defined as:

$$f_1(t_1 \| \ldots \| t_d \| b \| ind) = \begin{cases} t_1 \| \ldots \| t_d \| b \| 0 & \text{if } t_1 \| \ldots \| t_d \| b = \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \\ t_1 \| \ldots \| t_d \| b \| ind & \text{otherwise} \end{cases}$$

$$f_2(t_1\|\ldots\|t_d\|b\|ind) = \begin{cases} t_1\|\ldots\|t_d\|b\|\rho(ind) & \text{if } t_1\|\ldots\|t_d\|b = \tilde{t}_1\|\ldots\|\tilde{t}_d\|\tilde{b} \\ & \wedge ind \in [1, 2^\kappa) \\ t_1\|\ldots\|t_d\|b\|ind & \text{otherwise} \end{cases}$$

Then in step 2 and step 4, on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk_1, K, H, k_\iota, msg_\iota])$. Also, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk_2, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.

- *Game 5.* This is identical to Game 4 except that the challenger computes $\tilde{x}_0$ as follows:
$$\tilde{x}_0 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1\|\ldots\|\tilde{t}_d\|\tilde{b}\|\rho(ind^*))$$
Recall that as $ind^* \in (msg_Q^*, 2^\kappa)$, $\rho(ind^*) = 2^\kappa - 1$.
- *Game 6.* This is identical to Game 5 except that the challenger changes the way to answer the marking oracle and the challenge oracle. In particular, in step 2 and step 4, on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k_\iota, msg_\iota])$. Also, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.
- *Game 7.* This is identical to Game 6 except that the challenger computes $\tilde{x}_0$ and $\tilde{x}_1$ in step 1.
- *Game 8.* This is identical to Game 7 except that the challenger changes the way to answer the marking oracle and the challenge oracle. In particular, it generates $sk' \leftarrow \mathsf{PFE}.\mathtt{Puncture}(sk, \{\tilde{x}_0, \tilde{x}_1\})$ in step 1.
Then, in step 2 and step 4, on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it first computes $y_{0,\iota} = \mathsf{PRF}.\mathtt{Eval}(k_\iota, \tilde{x}_0)$ and $y_{1,\iota} = \mathsf{PRF}.\mathtt{Eval}(k_\iota, \tilde{x}_1)$ and sets $\mathcal{P}_\iota = \{(\tilde{x}_0, y_{0,\iota}), (\tilde{x}_1, y_{1,\iota})\}$. Then it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}^3[sk', K, H, k_\iota, msg_\iota, \mathcal{P}_\iota])$, where $\mathtt{M}^3$ is defined in Figure 7.
Also, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger first computes $y_0^* = \mathsf{PRF}.\mathtt{Eval}(k^*, \tilde{x}_0)$ and $y_1^* = \mathsf{PRF}.\mathtt{Eval}(k^*, \tilde{x}_1)$ and sets $\mathcal{P}^* = \{(\tilde{x}_0, y_0^*), (\tilde{x}_1, y_1^*)\}$. Then it generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}^3[sk', K, H, k^*, msg_i^*, \mathcal{P}^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.

Next, we prove the indistinguishability of each consecutive pair of games and show that the adversary $\mathcal{A}$ will win in the final game (Game 8) with a negligible probability. For simplicity of notation, we use $\mathcal{E}_i$ to denote the output of Game i.

Claim 23. *If* PRF *has correctness, weak key-injectivity and constrained one-wayness,* PFE *has sparseness and iO-compatible correctness, both* G *and* G' *are pseudorandom generators,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_4 = 1]| \leq negl(\lambda)$.

Proof. Claim 23 can be proved in a similar way as the proofs of Claim 18 to Claim 21. It is worth noting that, as security of F is not used in the proof of

$\mathtt{M}^3$

Constant: $sk, K, H, k, msg, \mathcal{P}$
Input: x

1. For $(x_i, y_i) \in \mathcal{P}$:
   (a) If $x = x_i$, then outputs y_i
2. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$.
3. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_d))) = b)$
   (a) $k' = \mathsf{F}.\mathtt{Eval}(K, t_1 \| \ldots \| t_d \| b)$.
   (b) Output $\mathsf{PRF}.\mathtt{Eval}(k', x)$.
4. Otherwise, output $\mathsf{PRF}.\mathtt{Eval}(k, x)$.

Fig. 7 The circuit $\mathtt{M}^3$.

Claim 23, simulators of reductions in this proof can always handle the secret key K of F and thus is able to generate $\tilde{y}_0$ and $\tilde{y}_1$ for the adversary. □

Claim 24. *If* PFE *is a secure puncturable functional encryption scheme, then* $|\Pr[\mathcal{E}_4 = 1] - \Pr[\mathcal{E}_5 = 1]| \leq negl(\lambda)$.

Proof. In Game 4 (and Game 5), only sk_1 and sk_2 are available in the view of $\mathcal{A}$ and each of them decrypts identically on $\tilde{x}_0$ from these two games. Thus, Indistinguishability between Game 4 and Game 5 comes from the adaptive indistinguishability of PFE by a direct reduction. □

Claim 25. *If* PFE *has iO-compatible correctness,* PRF *has weak key-injectivity,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_5 = 1] - \Pr[\mathcal{E}_6 = 1]| \leq negl(\lambda)$.

Proof. Proof of Claim 25 is similar to the proof of Claim 21, and we just omit the details here. □

Claim 26. $\Pr[\mathcal{E}_6 = 1] = \Pr[\mathcal{E}_7 = 1]$.

Proof. Obviously, the changes in Game 7 are purely conceptual and will not affect the output of the game, thus, Claim 26 follows. □

Claim 27. *If* PFE *has punctured correctness and sparseness,* PRF *has weak key-injectivity,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_7 = 1] - \Pr[\mathcal{E}_8 = 1]| \leq negl(\lambda)$.

Proof. As the only difference between Game 7 and Game 8 is that the adversary $\mathcal{A}$ gets obfuscations of different circuits in these two games, it is sufficient to prove that with all but negligible probability, all circuits $\mathcal{A}$ gets in Game 8 are functionally equivalent to their counterparts in Game 7. More concretely, let Q' be the number of marking oracle queries that is not aborted, we need to prove that:

$$\forall i \in [1, Q'], \Pr[\mathtt{M}[sk, K, H, k_i, msg_i] \not\equiv \mathtt{M}^3[sk', K, H, k_i, msg_i, \mathcal{P}_i]] \leq negl(\lambda) \quad (8)$$

$$\forall i \in [1, Q], \Pr[\mathtt{M}[sk, K, H, k^*, msg_i^*] \not\equiv \mathtt{M}^3[sk', K, H, k^*, msg_i^*, \mathcal{P}^*]] \leq negl(\lambda) \tag{9}$$

First, by the punctured correctness of PFE, with all but negligible probability, we have

$$\forall x \in \{0,1\}^n \backslash \{\tilde{x}_0, \tilde{x}_1\}, \mathsf{PFE}.\mathtt{Dec}(sk, x) = \mathsf{PFE}.\mathtt{Dec}(sk', x)$$

In addition, by the correctness of PFE, with all but negligible probability, we have

$$\mathsf{PFE}.\mathtt{Dec}(sk, \tilde{x}_0) = \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \| 1^\kappa$$

Also, by the sparseness of PFE, with all but negligible probability, we have

$$\mathsf{PFE}.\mathtt{Dec}(sk, \tilde{x}_1) = \perp$$

Besides, as $k_i \neq k^*$ for $i \in [1, Q']$, then by the weak key-injectivity of PRF and the collision resistance of $\mathcal{H}$, we have

$$\tilde{b} \neq H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d)))$$

with all but negligible probability. Next, we suppose the above four equations holds, which occurs with all but negligible probability.

We start by proving Equation (8). For any $i \in [1, Q']$ and for any input x, we consider the following three cases:

1. **Case I:** $x \notin \{\tilde{x}_0, \tilde{x}_1\}$. In this case, the check in step 1 of circuit $\mathtt{M}^3$ will not be validated and $\mathsf{PFE}.\mathtt{Dec}(sk, x) = \mathsf{PFE}.\mathtt{Dec}(sk', x)$. Thus the outputs of the two circuits $\mathtt{M}[sk, K, H, k_i, msg_i]$ and $\mathtt{M}^3[sk', K, H, k_i, msg_i, \mathcal{P}_i]$ are equal on input x.
2. **Case II:** $x = \tilde{x}_0$. In this case, $\mathtt{M}^3[sk', K, H, k_i, msg_i, \mathcal{P}_i](x) = y_{0,i} = \mathsf{PRF}.\mathtt{Eval}(k_i, x)$. Moreover, as $\tilde{b} \neq H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d)))$, we also have $\mathtt{M}[sk, K, H, k_i, msg_i](x) = \mathsf{PRF}.\mathtt{Eval}(k_i, x)$.
3. **Case III:** $x = \tilde{x}_1$. In this case, $\mathtt{M}^3[sk', K, H, k_i, msg_i, \mathcal{P}_i](x) = y_{1,i} = \mathsf{PRF}.\mathtt{Eval}(k_i, x)$. Moreover, as $\mathsf{PFE}.\mathtt{Dec}(sk, \tilde{x}_1) = \perp$, we also have $\mathtt{M}[sk, K, H, k_i, msg_i](x) = \mathsf{PRF}.\mathtt{Eval}(k_i, x)$.

In summary, for any $i \in [1, Q']$ and for any input, the output of the circuit $\mathtt{M}[sk, K, H, k_i, msg_i]$ and that of the circuit $\mathtt{M}^3[sk', K, H, k_i, msg_i, \mathcal{P}_i]$ are identical.

Then, we prove Equation (9). For any $i \in [1, Q]$ and for any input x, we consider the following three cases:

1. **Case I:** $x \notin \{\tilde{x}_0, \tilde{x}_1\}$. In this case, the check in step 1 of circuit $\mathtt{M}^3$ will not be validated and $\mathsf{PFE}.\mathtt{Dec}(sk, x) = \mathsf{PFE}.\mathtt{Dec}(sk', x)$. Thus the outputs of the two circuits $\mathtt{M}[sk, K, H, k^*, msg_i^*]$ and $\mathtt{M}^3[sk', K, H, k^*, msg_i^*, \mathcal{P}^*]$ are equal on input x.
2. **Case II:** $x = \tilde{x}_0$. In this case, $\mathtt{M}^3[sk', K, H, k^*, msg_i^*, \mathcal{P}^*](x) = y_0^* = \mathsf{PRF}.\mathtt{Eval}(k^*, x)$. Moreover, as $2^\kappa - 1 > msg_Q^*$ (otherwise, the lemma holds trivially), we also have $\mathtt{M}[sk, K, H, k^*, msg_i^*](x) = \mathsf{PRF}.\mathtt{Eval}(k^*, x)$.

3. **Case III:** $x = \tilde{x}_1$. In this case, $\mathtt{M}^3[sk', K, H, k^*, msg_i^*, \mathcal{P}^*](x) = y_1^* = \mathsf{PRF}.\mathtt{Eval}(k^*, x)$. Moreover, as $\mathsf{PFE}.\mathtt{Dec}(sk, \tilde{x}_1) = \perp$, we also have $\mathtt{M}[sk, K, H, k^*, msg_i^*](x) = \mathsf{PRF}.\mathtt{Eval}(k^*, x)$.

In summary, for any $i \in [1, Q]$ and for any input, the output of the circuit $\mathtt{M}[sk, K, H, k^*, msg_i^*]$ and that of the circuit $\mathtt{M}^3[sk', K, H, k^*, msg_i^*, \mathcal{P}^*]$ are identical.

As with all but negligible probability, we have both Equation (8) and Equation (9) hold, Claim 27 follows.

□

Claim 28. *If* PFE *has ciphertext pseudorandomness, then* $\Pr[\mathcal{E}_8 = 1] \leq negl(\lambda)$.

Proof. In Game 8, the two challenge inputs $\tilde{x}_0$ and $\tilde{x}_1$ are delt with identically once generated. Thus, Claim 28 comes from the ciphertext pseudorandomness of PFE via a direct reduction. □

Combining Claim 23 to Claim 28, we can conclude that the probability that $\mathcal{A}$ wins in Game 0 (i.e., in the real experiment $\mathtt{AUR}^2$) is also negligible. This completes the proof of Lemma C.5.

C.3.5 Proof of Lemma C.6 In this section, we prove Lemma C.6. First, we define the following games between a challenger and a PPT unremoving-admissible adversary $\mathcal{A}$:

- *Game 0.* This is the real experiment $\mathtt{AUR}^3$.
- *Game 1.* In Game 1, the challenger changes the way for generating variables in step 5. In particular, in step 5, on receiving the circuit $\tilde{\mathtt{C}}$ and ind^*, the challenger proceeds as follows:
  1. Sample $\tilde{t}_1, \ldots, \tilde{t}_d \xleftarrow{\$} \{0,1\}^l$.
  2. $\tilde{b} = H(\mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k^*, \mathsf{G}(\tilde{t}_d)))$.
  3. $\tilde{x}_0 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b} \| ind^*)$
  4. $\tilde{x}_1 \xleftarrow{\$} \{0,1\}^n$.
  5. $\beta \xleftarrow{\$} \{0,1\}$.
  6. Return $\tilde{x}_\beta$ to $\mathcal{A}$.
- *Game 2.* This is identical to Game 1 except that the challenger samples $k^*, \tilde{t}_1, \ldots, \tilde{t}_d$ and computes $\tilde{b}$ immediately after it generates H, K, mpk, sk in step 1.
- *Game 3.* This is identical to Game 2 except that the challenger further checks if k^* has been submitted to the marking oracle. In particular, in step 2 and in step 4, on receiving a query (k_ι, msg_ι), the challenger checks if $k_\iota = k^*$. It aborts and outputs 2 if this is the case; otherwise, it proceeds identically as in Game 2.
- *Game 4.* This is identical to Game 3 except that the challenger changes the way to answer the marking oracle and the challenge oracle. In particular, it

generates $sk_1 \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, f_1)$ and $sk_2 \leftarrow \mathsf{PFE}.\mathtt{KeyGen}(msk, f_2)$ in step 2 and step 3 respectively, where f_1 and f_2 is defined as:

$$f_1(t_1\| \dots \|t_d\|b\|ind) = \begin{cases} t_1\| \dots \|t_d\|b\|0 & \text{if } t_1\| \dots \|t_d\|b = \tilde{t}_1\| \dots \|\tilde{t}_d\|\tilde{b} \\ t_1\| \dots \|t_d\|b\|ind & \text{otherwise} \end{cases}$$

$$f_2(t_1\| \dots \|t_d\|b\|ind) = \begin{cases} t_1\| \dots \|t_d\|b\|\rho(ind) & \text{if } t_1\| \dots \|t_d\|b = \tilde{t}_1\| \dots \|\tilde{t}_d\|\tilde{b} \\ & \wedge ind \in [1, 2^\kappa) \\ t_1\| \dots \|t_d\|b\|ind & \text{otherwise} \end{cases}$$

Then in step 2 and step 4, on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk_1, K, H, k_\iota, msg_\iota])$. Also, in step 3, on receiving Q messages $\{msg_1^*, \dots, msg_Q^*\}$, the challenger generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk_2, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \dots, \mathtt{C}_Q^*\}$ back.

- *Game 5.* This is identical to Game 4 except that the challenger computes $\tilde{x}_0$ as follows:
$$\tilde{x}_0 \leftarrow \mathsf{PFE}.\mathtt{Enc}(mpk, \tilde{t}_1\| \dots \|\tilde{t}_d\|\tilde{b}\|\rho(ind^*))$$
Recall that as $ind^* \in [1, msg_1^*]$, $\rho(ind^*) = 1$.
- *Game 6.* This is identical to Game 5 except that the challenger changes the way to answer the marking oracle and the challenge oracle. In particular, in step 2 and step 4, on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k_\iota, msg_\iota])$. Also, in step 3, on receiving Q messages $\{msg_1^*, \dots, msg_Q^*\}$, the challenger generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}[sk, K, H, k^*, msg_i^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \dots, \mathtt{C}_Q^*\}$ back.
- *Game 7.* This is identical to Game 6 except that the challenger computes $\tilde{x}_0$ and $\tilde{x}_1$ in step 1.
- *Game 8.* This is identical to Game 7 except that the challenger changes the way to answer the challenge oracle. More precisely, in step 3, on receiving Q messages $\{msg_1^*, \dots, msg_Q^*\}$, the challenger first samples $y_1^* \xleftarrow{\$} \{0,1\}^m$ and sets $\mathcal{P}^* = \{(\tilde{x}_1, y_1^*)\}$. Then it generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}^3[sk, K, H, k^*, msg_i^*, \mathcal{P}^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \dots, \mathtt{C}_Q^*\}$ back.
- *Game 9.* This is identical to Game 8 except that the challenger changes the way to answer the challenge oracle. More precisely, in step 3, on receiving Q messages $\{msg_1^*, \dots, msg_Q^*\}$, the challenger first samples $y_0^*, y_1^* \xleftarrow{\$} \{0,1\}^m$ and sets $\mathcal{P}^* = \{(\tilde{x}_0, y_0^*), (\tilde{x}_1, y_1^*)\}$. Then it generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}^3[sk, K, H, k^*, msg_i^*, \mathcal{P}^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \dots, \mathtt{C}_Q^*\}$ back.
- *Game 10.* This is identical to Game 9 except that the challenger changes the way to answer the marking oracle and the challenge oracle. In particular, it generates $sk' \leftarrow \mathsf{PFE}.\mathtt{Puncture}(sk, \{\tilde{x}_0, \tilde{x}_1\})$ in step 1.
Then, in step 2 and step 4, on receiving a query (k_ι, msg_ι) (for the ιth marking oracle query), it first computes $y_{0,\iota} = \mathsf{PRF}.\mathtt{Eval}(k_\iota, \tilde{x}_0)$ and $y_{1,\iota} = \mathsf{PRF}.\mathtt{Eval}(k_\iota, \tilde{x}_1)$ and sets $\mathcal{P}_\iota = \{(\tilde{x}_0, y_{0,\iota}), (\tilde{x}_1, y_{1,\iota})\}$ Then it returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}^3[sk', K, H, k_\iota, msg_\iota, \mathcal{P}_\iota])$.

Also, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger first samples $y_0^*, y_1^* \xleftarrow{\$} \{0,1\}^m$ and sets $\mathcal{P}^* = \{(\tilde{x}_0, y_0^*), (\tilde{x}_1, y_1^*)\}$. Then it generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}^3[sk', K, H, k^*, msg_i^*, \mathcal{P}^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back.

Next, we prove the indistinguishability of each consecutive pair of games and show that the adversary $\mathcal{A}$ will win in the final game (Game 10) with a negligible probability. For simplicity of notation, we use $\mathcal{E}_i$ to denote the output of Game i.

Claim 29. *If* PRF *has correctness, weak key-injectivity and constrained one-wayness,* PFE *is a secure puncturable functional encryption scheme, both* G *and* G' *are pseudorandom generators,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_0 = 1] - \Pr[\mathcal{E}_7 = 1]| \leq negl(\lambda)$.

Proof. Claim 29 can be proved in a similar way as the proofs of Claim 23 to Claim 26. □

Claim 30. *If* PRF *is a secure puncturable PRF with weak key-injectivity,* PFE *has sparseness, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_7 = 1] - \Pr[\mathcal{E}_8 = 1]| \leq negl(\lambda)$.

Proof. To prove Claim 30, we further define the following auxiliary games:

- *Game* 7.1. This is identical to Game 7 except that the challenger changes the way to generate $\tilde{b}$. More precisely, after sampling k^* and $\tilde{x}_1$ in step 1, the challenger generates $ck^* \leftarrow \mathsf{PRF}.\mathtt{Constrain}(k^*, \tilde{x}_1)$. Then it computes

  $$\tilde{b} = H(\mathsf{PRF}.\mathtt{ConstrainEval}(ck^*, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{ConstrainEval}(ck^*, \mathsf{G}(\tilde{t}_d)))$$

- *Game* 7.2. This is identical to Game 7.1 except that the challenger uses ck^* instead of k^* to test if k^* has been submitted to the marking oracle. In particular, on receiving a query (k_ι, msg_ι) in step 2 and step 4, the challenger first chooses an arbitrary $x \in \{0,1\}^n$ that $x \neq \tilde{x}_1$. Then it aborts and outputs 2 if and only if

  $$\mathsf{PRF}.\mathtt{Eval}(k_\iota, x) = \mathsf{PRF}.\mathtt{ConstrainEval}(ck^*, x)$$

- *Game* 7.3. This is identical to Game 7.2 except that the challenger changes the way to answer the challenge oracle. In more detail, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger first computes $y_1^* = \mathsf{PRF}.\mathtt{Eval}(k^*, \tilde{x}_1)$ and sets $\mathcal{P}^* = \{(\tilde{x}_1, y_1^*)\}$. Then it generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}^4[sk, K, H, ck^*, msg_i^*, \mathcal{P}^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back, where the circuit $\mathtt{M}^4$ is defined in Figure 8.
- *Game* 7.4. This is identical to Game 7.3 except that the challenger samples $y_1^* \xleftarrow{\$} \{0,1\}^m$ instead of computing it from $\tilde{x}_1$.

$\mathtt{M}^4$

Constant: $sk, K, H, ck, msg, \mathcal{P}$
Input: x

1. For $(x_i, y_i) \in \mathcal{P}$:
   (a) If $x = x_i$, then outputs y_i
2. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$.
3. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}.\mathtt{ConstrainEval}(ck, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{ConstrainEval}(ck, \mathsf{G}(t_d))) = b)$
   (a) $k' = \mathsf{F}.\mathtt{Eval}(K, t_1 \| \ldots \| t_d \| b)$.
   (b) Output $\mathsf{PRF}.\mathtt{Eval}(k', x)$.
4. Otherwise, output $\mathsf{PRF}.\mathtt{ConstrainEval}(ck, x)$.

Fig. 8 The circuit $\mathtt{M}^4$.

Next, we will argue the indistinguishability between each consecutive pair of games:

- **Game 7 and Game 7.1.** By the correctness of PRF, $\tilde{b}$ is differently computed in Game 7 and Game 7.1 only if there exists $j \in [1, d]$ that $\tilde{x}_1 = \mathsf{G}(\tilde{t}_j)$. This can occur with only a negligible probability since $\tilde{x}_1$ is sampled uniformly in $\{0,1\}^n$.
- **Game 7.1 and Game 7.2.** The outputs of Game 7.1 and Game 7.2 differ only if there exists k_i that $k_i \neq k^*$ and $\mathsf{PRF}.\mathtt{Eval}(k_i, x) = \mathsf{PRF}.\mathtt{ConstrainEval}(ck^*, x) = \mathsf{PRF}.\mathtt{Eval}(k^*, x)$, which occurs with only a negligible probability due to the weak key-injectivity of PRF.
- **Game 7.2 and Game 7.3.** As the only difference between Game 7.2 and Game 7.3 is that the adversary $\mathcal{A}$ gets obfuscations of different circuits in these two games, it is sufficient to prove that with all but negligible probability, all circuits $\mathcal{A}$ gets in Game 7.3 are functionally equivalent to their counterparts in Game 7.2. More concretely, we need to prove that:

$$\forall i \in [1, Q], \Pr[\mathtt{M}[sk, K, H, k^*, msg_i^*] \not\equiv \mathtt{M}^4[sk, K, H, ck^*, msg_i^*, \mathcal{P}^*]] \leq negl(\lambda) \tag{10}$$

First, as $\tilde{x}_1$ is sampled uniformly from $\{0,1\}^n$, the probability that it falls in the range of G, which contains only 2^l elements in $\{0,1\}^n$, is negligible, thus we have

$$\forall t \in \{0,1\}^l, \mathsf{G}(t) \neq \tilde{x}_1$$

Also, by the sparseness of PFE, with all but negligible probability, we have

$$\mathsf{PFE}.\mathtt{Dec}(sk, \tilde{x}_1) = \perp$$

Next, we suppose the above two equations hold, which occurs with all but negligible probability.
Now, for any $i \in [1, Q]$ and for any input x, we consider the following two cases:

1. **Case I:** $x \neq \tilde{x}_1$. In this case, the check in step 1 of circuit $\mathtt{M}^4$ will not be validated. Moreover, $\mathsf{PRF}.\mathtt{ConstrainEval}(ck^*, \cdot)$ is not required to compute on $\tilde{x}_1$, thus the outputs of the two circuits $\mathtt{M}[sk, K, H, k^*, msg_i^*]$ and $\mathtt{M}^4[sk, K, H, ck^*, msg_i^*, \mathcal{P}^*]$ are equal on input x.
2. **Case II:** $x = \tilde{x}_1$. In this case, $\mathtt{M}^4[sk, K, H, ck^*, msg_i^*, \mathcal{P}^*](x) = y_1^* = \mathsf{PRF}.\mathtt{Eval}(k^*, x)$. Moreover, as $\mathsf{PFE}.\mathtt{Dec}(sk, \tilde{x}_1) = \perp$, we also have $\mathtt{M}[sk, K, H, k^*, msg_i^*](x) = \mathsf{PRF}.\mathtt{Eval}(k^*, x)$.

As with all but negligible probability, Equation (10) holds, indistinguishability between Game 7.2 and Game 7.3 follows.

- **Game** 7.3 **and Game** 7.4**.** Indistinguishability between Game 7.3 and Game 7.4 comes from the constrained pseudorandomness of PRF via a direct reduction.
- **Game** 7.4 **and Game** 8**.** In Game 8, we reverse changes introduced in Game 7.1 to Game 7.3. Thus, indistinguishability between Game 7.4 and Game 8 can be proved in a similar way as that of indistinguishabilities from Game 7 to Game 7.3.

□

Claim 31. *If* PRF *is a secure puncturable PRF with weak key-injectivity,* F *is a secure prefix puncturable PRF,* $\mathcal{H}$ *is a family of collision-resistant hash function,* PFE *has correctness, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_8 = 1] - \Pr[\mathcal{E}_9 = 1]| \leq negl(\lambda)$.

Proof. To prove Claim 31, we further define the following auxiliary games:

- *Game* 8.1. This is identical to Game 8 except that the challenger changes the way to generate $\mathtt{E}$. More precisely, the challenger computes

$$CK \leftarrow \mathsf{F}.\mathtt{Constrain}(K, \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b})$$

  after generating K. Then it computes $\mathtt{E}$ as

$$\mathtt{E} \leftarrow \mathsf{iO}(\mathtt{Ext}^1[mpk, CK])$$

  Recall that the circuit $\mathtt{Ext}^1$ is defined in Figure 4.
- *Game* 8.2. This is identical to Game 8.1 except that the challenger changes the way to answer marking oracles. More precisely, in step 2 and step 4, on receiving a query (k_ι, msg_ι), the challenger returns $\mathtt{C}_\iota \leftarrow \mathsf{iO}(\mathtt{M}^5[sk, CK, H, k_\iota, msg_\iota])$, where $\mathtt{M}^5$ is defined in Figure 9.
- *Game* 8.3. This is identical to Game 8.2 except that the challenger changes the way to answer the challenge oracle. More precisely, in step 3, on receiving Q messages $\{msg_1^*, \ldots, msg_Q^*\}$, the challenger first computes $\hat{k}' = \mathsf{F}.\mathtt{Eval}(K, \tilde{t}_1 \ldots \| \tilde{t}_d \| \tilde{b})$ and generates a circuit $\mathtt{F} = \mathsf{PRF}.\mathtt{Eval}(\hat{k}', \cdot)$. Then it samples $y_1^* \xleftarrow{\$} \{0,1\}^m$ and sets $\mathcal{P}^* = \{(\tilde{x}_1, y_1^*)\}$. Next, it generates $\mathtt{C}_i^* \leftarrow \mathsf{iO}(\mathtt{M}^6[sk, CK, H, k^*, msg_i^*, \tilde{\mu}, \mathtt{F}, \mathcal{P}^*])$ for $i \in [1, Q]$ and returns $\{\mathtt{C}_1^*, \ldots, \mathtt{C}_Q^*\}$ back, where $\tilde{\mu} = \tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b}$ and $\mathtt{M}^6$ is defined in Figure 9.

- *Game* 8.4. This is identical to Game 8.3 except that the challenger generates $\hat{k}' \leftarrow \mathsf{PRF}.\mathtt{KeyGen}(1^\lambda)$ instead of generating it from K and $\tilde{u}$. Recall that this is equal to sample $\hat{k}'$ uniformly at random from $\mathcal{K}$.
- *Game* 8.5. This is identical to Game 8.4 except that the challenger changes the way to generate circuit $\mathtt{F}$. In particular, it first computes $\hat{ck}' \leftarrow \mathsf{PRF}.\mathtt{Constrain}(\hat{k}', \tilde{x}_0)$ and $y_0^* = \mathsf{PRF}.\mathtt{Eval}(\hat{k}', \tilde{x}_0)$. Then, it sets

$$\mathtt{F}(x) = \begin{cases} \mathsf{PRF}.\mathtt{ConstrainEval}(\hat{ck}', x) & \text{if } x \neq \tilde{x}_0 \\ y_0^* & \text{Otherwise} \end{cases}$$

- *Game* 8.6. This is identical to Game 8.5 except that the challenger samples $y_0^* \xleftarrow{\$} \{0,1\}^m$.
- *Game* 8.7. This is identical to Game 8.6 except that the challenger sets $\mathcal{P}^*$ as $\mathcal{P}^* = \{(\tilde{x}_0, y_0^*), (\tilde{x}_1, y_1^*)\}$.

$\mathtt{M}^5$

Constant: sk, CK, H, k, msg
Input: x

1. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$.
2. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_d))) = b)$
   (a) $k' = \mathsf{F}.\mathtt{ConstrainEval}(CK, t_1 \| \ldots \| t_d \| b)$.
   (b) Output $\mathsf{PRF}.\mathtt{Eval}(k', x)$.
3. Otherwise, output $\mathsf{PRF}.\mathtt{Eval}(k, x)$.

$\mathtt{M}^6$

Constant: $sk, CK, H, k, msg, \mu, \mathtt{F}, \mathcal{P}$
Input: x

1. For $(x_i, y_i) \in \mathcal{P}$:
   (a) If $x = x_i$, then outputs y_i
2. $(t_1 \| \ldots \| t_d \| b \| ind) = \mathsf{PFE}.\mathtt{Dec}(sk, x)$.
3. If $(t_1 \| \ldots \| t_d \| b \| ind \neq \perp) \wedge (ind \leq msg) \wedge (H(\mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k, \mathsf{G}(t_d))) = b)$
   (a) If $t_1 \| \ldots \| t_d \| b = \mu$:
      i. Output $\mathtt{F}(x)$.
   (b) $k' = \mathsf{F}.\mathtt{ConstrainEval}(CK, t_1 \| \ldots \| t_d \| b)$.
   (c) Output $\mathsf{PRF}.\mathtt{Eval}(k', x)$.
4. Otherwise, output $\mathsf{PRF}.\mathtt{Eval}(k, x)$.

Fig. 9 The circuits $\mathtt{M}^5$ and $\mathtt{M}^6$.

Next, we will argue the indistinguishability between each consecutive pair of games:

- **Game** 8 **and Game** 8.1**.** As the only difference between Game 8 and Game 8.1 is that the adversary $\mathcal{A}$ gets obfuscations of different circuits in these

two games, it is sufficient to prove that with all but negligible probability, all circuits $\mathcal{A}$ gets in Game 8.1 are functionally equivalent to their counterparts in Game 8. More concretely, we need to prove

$$\Pr[\mathtt{Ext}[mpk, K] \not\equiv \mathtt{Ext}^{1}[mpk, CK]] \leq negl(\lambda) \tag{11}$$

For any input $(a_1, \ldots, a_d, b, ind, r)$, the two circuits $\mathtt{Ext}[mpk, K]$ and $\mathtt{Ext}^{1}[mpk, CK]$ compute differently only if $(\mathsf{G}'(a_1), \ldots, \mathsf{G}'(a_d)) = (\tilde{t}_1, \ldots, \tilde{t}_d)$. Such input exists with only a negligible probability since for any $j \in [1, d]$, $\tilde{t}_j$ is chosen uniformly at random from $\{0,1\}^l$ and the probability that it falls in the range of G', which contains only $2^{\frac{l}{2}}$ elements, is negligible.

- **Game** 8.1 **and Game** 8.2**.** As the only difference between Game 8.1 and Game 8.2 is that the adversary $\mathcal{A}$ gets obfuscations of different circuits in these two games, it is sufficient to prove that with all but negligible probability, all circuits $\mathcal{A}$ gets in Game 8.2 are functionally equivalent to their counterparts in Game 8.1. More concretely, let Q' be the number of marking oracle queries that is not aborted, we need to prove that:

$$\forall i \in [1, Q'], \Pr[\mathtt{M}[sk, K, H, k_i, msg_i] \not\equiv \mathtt{M}^{5}[sk, CK, H, k_i, msg_i]] \leq negl(\lambda) \tag{12}$$

  First, as $k_i \neq k^*$ for $i \in [1, Q']$, by the weak key-injectivity of PRF and the collision resistance of $\mathcal{H}$, we have

$$\tilde{b} \neq H(\mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_1)), \ldots, \mathsf{PRF}.\mathtt{Eval}(k_i, \mathsf{G}(\tilde{t}_d)))$$

  with all but negligible probability. Thus, with all but negligible probability, in circuit $\mathtt{M}^{5}[sk, CK, H, k_i, msg_i]$, $\mathsf{PRF}.\mathtt{ConstrainEval}(CK, \cdot)$ is not required to compute on $\tilde{t}_1 \| \ldots \| \tilde{t}_d \| \tilde{b}$. Then, by the correctness of PRF, equivalence of $\mathtt{M}[sk, K, H, k_i, msg_i]$ and $\mathtt{M}^{5}[sk, CK, H, k_i, msg_i]$ follows.
- **Game** 8.2 **and Game** 8.3**.** It is easy to see that by the correctness of F,

$$\mathtt{M}^{3}[sk, K, H, k^*, msg_i^*, \mathcal{P}^*] \equiv \mathtt{M}^{6}[sk, CK, H, k^*, msg_i^*, \mu, \mathtt{F}, \mathcal{P}^*]$$

  Thus, indistinguishability of Game 8.2 and Game 8.3 comes from indistinguishability of iO directly.
- **Game** 8.3 **and Game** 8.4**.** Indistinguishability of Game 8.3 and Game 8.4 comes from punctured pseudorandomness of F by a direct reduction.
- **Game** 8.4 **and Game** 8.5**.** By the correctness of PRF, the circuit $\mathtt{F}$ in Game 8.4 and that in Game 8.5 are functionally equivalent. Thus, the circuit $\mathtt{M}^{6}[sk, CK, H, k^*, msg_i^*, \mu, \mathtt{F}, \mathcal{P}^*]$ in Game 8.4 and that in Game 8.5 are also functionally equivalent. As a result, indistinguishability of Game 8.4 and Game 8.5 comes from indistinguishability of iO directly.
- **Game** 8.5 **and Game** 8.6**.** Indistinguishability of Game 8.5 and Game 8.6 comes from punctured pseudorandomness of PRF by a direct reduction.
- **Game** 8.6 **and Game** 8.7**.** To prove indistinguishability of Game 8.6 and Game 8.7, we need to show that $\mathtt{M}^{6}[sk, CK, H, k^*, msg_i^*, \mu, \mathtt{F}, \mathcal{P}^*]$ is identically evaluated in these two games. First, as $\tilde{x}_1$ is sampled uniformly, with

all but negligible probability, we have $\tilde{x}_0 \neq \tilde{x}_1$. Thus, it is sufficient to prove that in Game 8.6 we also have $\mathtt{M}^6[sk, CK, H, k^*, msg_i^*, \mu, \mathtt{F}, \mathcal{P}^*](\tilde{x}_0) = y_0^*$. This comes from the correctness of PFE straightforwardly.

- **Game** 8.7 **and Game** 9**.** In Game 9, we reverse changes introduced in Game 8.1 to Game 8.5. Thus, indistinguishability between Game 8.7 and Game 9 can be proved in a similar way as that of indistinguishabilities from Game 8 to Game 8.5.

□

Claim 32. *If* PFE *has punctured correctness and sparseness,* PRF *has weak key-injectivity,* $\mathcal{H}$ *is a family of collision-resistant hash function, and* iO *is a secure indistinguishability obfuscator, then* $|\Pr[\mathcal{E}_9 = 1] - \Pr[\mathcal{E}_{10} = 1]| \leq negl(\lambda)$.

Proof. Proof of Claim 32 can be proved in a similar way as the proof of Claim 27 and we omit the details here. □

Claim 33. *If* PFE *has ciphertext pseudorandomness, then* $\Pr[\mathcal{E}_{10} = 1] \leq negl(\lambda)$.

Proof. In Game 10, the two challenge inputs $\tilde{x}_0$ and $\tilde{x}_1$ are delt with identically once generated. Thus, Claim 33 comes from the ciphertext pseudorandomness of PFE via a direct reduction. □

Combining Claim 29 to Claim 33, we can conclude that the probability that $\mathcal{A}$ wins in Game 0 (i.e., in the real experiment $\mathtt{AUR}^3$) is also negligible. This completes the proof of Lemma C.6.