Watermarking PRFs Under Standard Assumptions: Public Marking and Security with Extraction Queries

Willy Quach(✉), Daniel Wichs, and Giorgos Zirdelis

Northeastern University, Boston, USA
quach.w@husky.neu.edu, wichs@ccs.neu.edu, zirdelis.g@husky.neu.edu

Abstract. A software watermarking scheme can embed some information called a mark into a program while preserving its functionality. No adversary can remove the mark without damaging the functionality of the program. Cohen et al. (STOC '16) gave the first positive results for watermarking, showing how to watermark certain *pseudorandom function (PRF) families* using indistinguishability obfuscation (iO). Their scheme has a secret marking procedure to embed marks in programs and a public extraction procedure to extract the marks from programs; security holds even against an attacker that has access to a marking oracle. Kim and Wu (CRYPTO '17) later constructed a PRF watermarking scheme under only the LWE assumption. In their scheme, both the marking and extraction procedures are secret, but security only holds against an attacker with access to a marking oracle *but not* an extraction oracle. In fact, it is possible to completely break the security of the latter scheme using extraction queries, which is a significant limitation in any foreseeable application.

In this work, we construct a new PRF watermarking scheme with the following properties.

- The marking procedure is public and therefore anyone can embed marks in PRFs from the family. Previously we had no such construction even using obfuscation.
- The extraction key is secret, but marks remain unremovable even if the attacker has access to an extraction oracle. Previously we had no such construction under standard assumptions.
- Our scheme is simple, uses generic components and can be instantiated under many different assumptions such as DDH, Factoring or LWE.

The above benefits come with one caveat compared to prior work: the PRF family that we can watermark depends on the public parameters of the watermarking scheme and the watermarking authority has a secret key which can break the security of all of the PRFs in the family. Since the watermarking authority is usually assumed to be trusted, this caveat appears to be acceptable.

Supported by NSF grants CNS-1314722, CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship.

A. Beimel and S. Dziembowski (Eds.): TCC 2018, LNCS 11240, pp. 669–698, 2018.
https://doi.org/10.1007/978-3-030-03810-6_24

1 Introduction

Watermarking allows us to embed some special information called a *mark* into digital objects such as images, movies, music files, or software. There are two basic requirements: firstly, a marked object should not be significantly different from the original object, and secondly, it should be impossible to remove an embedded mark without somehow "destroying" the object.

The works of Barak et al. [2,3] and Hopper, Molnar and Wagner [14] initiated the first theoretical study of program watermarking including rigorous definitions. However, positive results for watermarking remained elusive. A few early works [17,18,20] gave very partial results showing that certain cryptographic functions can be watermarked, but security only held against restricted adversaries with limited ability to modify the program. For example, in such schemes it is easy to remove the watermark by obfuscating the program without changing its functionality. The first positive result for watermarking against arbitrary removal strategies was given in the work of Cohen et al. [10] who showed how to watermark certain families of pseudo-random functions (PRFs). However, this result relies on the heavy hammer of *indistinguishability obfuscation (iO)* [2,3,12]. Later, the work of Kim and Wu [16] constructed a PRF watermarking scheme under only the learning-with-errors (LWE) assumption, but at the cost of weakening security. We first describe the problem of watermarking PRFs in more detail, then come back to discuss the above two works and finally present our new contributions.

Watermarking PRFs. A watermarking scheme for a PRF family $\{F_k\}$ consists of two procedures Mark and $\mathsf{Extract}$. The Mark procedure takes as input a PRF F_k from the family and outputs a program P which is a marked version of the PRF. We want *approximate correctness*, meaning that $F_k(x) = P(x)$ for all but a negligible fraction of inputs x and these should be hard to find. The $\mathsf{Extract}$ procedure takes as input a program P' and determines whether it is marked or unmarked. The main security property that we desire is *unremovability*: if we choose F_k randomly from the family and give the marked version P to an adversary, the adversary should be unable to come up with any program P' that even ε-approximates P for some small ε (meaning that $P(x) = P'(x)$ for an ε fraction of inputs x) yet the extraction procedure fails to recognize P' as marked. Each of the procedures $\mathsf{Mark}, \mathsf{Extract}$ may either be "public" meaning that it only relies on the public parameters of the watermarking scheme, or it may be "secret" meaning that it requires a secret key of the watermarking scheme. If one (or both) of the procedures is secret then the unremovability security property should hold even if the adversary gets oracle access to that procedure. We can also consider "message embedding" schemes, where the marking procedure additionally takes a message and the extraction procedure recovers the message from a marked

program – the unremovability property should then ensure that the adversary cannot remove the mark or modify the embedded message.[1]

There are several reason why watermarking PRFs is interesting. Firstly, watermarking in general is a poorly understood cryptographic concept yet clearly desirable in practice – therefore any kind of positive result is fascinating since it helps us get a better understanding of this elusive notion. Secondly, software watermarking only makes sense for unlearnable functions (as formalized in [10]) so we need to focus on cryptographic programs such as PRFs rather than (e.g.,) tax preparation software. Lastly, PRFs are a basic building block for more advanced cryptosystems and therefore watermarking PRFs will also allow us to watermark more advanced primitives that rely on PRFs, such as symmetric-key encryption or authentication schemes. See [10] for further discussion and potential applications of watermarked PRFs.

Prior Work. The work of Cohen et al. [10] showed how to watermark any family of puncturable PRFs using indistinguishability obfuscation (iO). They constructed a watermarking scheme with secret marking and public extraction, where the unremovability property holds even if the adversary has access to the marking oracle. The use of obfuscation may have appeared inherent in that result. However, Kim and Wu [16] (building on [5]) surprisingly showed how to remove it and managed to construct a watermarking scheme for a specific PRF family under only the learning-with-errors (LWE) assumption. In their scheme, both the marking and the extraction procedures are secret, but the unremovability security property only holds if the adversary has access to the marking oracle *but not* the extraction oracle. In particular, an adversary that can test whether arbitrary programs are marked or unmarked can completely break the security of the watermarking scheme. Since the entire point of watermarking is to use the extraction procedure on programs that may potentially have been constructed by an adversary, it is hard to justify that the adversary does not get access to the extraction oracle. Therefore this should be considered as a significant limitation of that scheme in any foreseeable application.

Our Results. In this work, we construct a watermarking scheme for a PRF family under standard assumptions. In particular, we only rely on CCA-secure public-key encryption with pseudorandom ciphertexts, which can be instantiated under most standard public-key assumptions such as DDH, LWE or Factoring. Our watermarking scheme has public marking and secret extraction, and the

[1] Some previous watermarking schemes also required an *unforgeability* property, which roughly says that an adversary should not be able to produce any marked functions on his own – in fact, he should not be able to come up with a function which is marked but is far from any of the marked functions that were output by the marking oracle. This property appears to be orthogonal to the main watermarking requirement of unremovability and we do not consider it here. In particular, it crucially requires a scheme with a secret marking procedure whereas here we construct watermarking scheme with a public marking procedure.

unremovability security property holds even if the adversary has access to the extraction oracle. We emphasize that:

- This is the first watermarking scheme with a public marking procedure. Previously such schemes were not known even under iO.
- This is the first watermarking scheme under standard assumptions where unremovability holds in the presence of the extraction oracle. Previously we only had such schemes under iO, in which case it was possible to even get public extraction, but not under any standard assumptions.
- This is the first watermarking scheme altogether under assumptions other than LWE or iO.

Our basic scheme is not message embedding (whereas the constructions of [10,16] are), but we also show how to get a message embedding scheme by additionally relying on generic constraint-hiding constrained PRFs, which we currently have under LWE [4,8,9,19].

Additionally, we allow an adversary who tries to remove the mark of some program P to change a *large* fraction of its outputs, matching the security guarantee of [10] based on iO. In comparison, the work of [16] based on standard assumptions restricts an adversary to only modify a *very small* fraction of its inputs. More precisely, while [16] only allows an adversary to only change a *negligible* fraction of the outputs of P, our construction without message embedding allows him to modify *almost all* of these outputs, as long as a polynomial fraction remains the same; and our construction with message embedding allows an adversary to change almost half of the outputs which is essentially optimal (as shown in [10])

Our scheme comes with one caveat that was not present in prior works. The PRF family that we watermark depends on the public-parameters of the watermarking scheme and it is possible to break the PRF security of this family given the watermarking secret key. In particular, this means that the watermarking authority which sets up the scheme can break the PRF security of all functions in the family, even ones that were never marked. However, we ensure that PRF security continues to hold even given the public parameters of the watermarking scheme and oracle access to the extraction procedure. Therefore the PRFs remain secure for everyone else *except* the watermarking authority. Technically, this caveat makes our results incomparable with those in prior works. However, we argue that since the watermarking authority is anyway assumed to be a trusted party in order for the watermarking guarantees to be meaningful, this caveat doesn't significantly detract from the spirit of the problem and our solutions with this caveat are still very meaningful.

1.1 Our Techniques

Watermarking Unpredictable Functions. To give the intuition behind our scheme, we first describe a simplified construction which allows us to watermark an *unpredictable* (but not yet pseudorandom) function family.[2]

The public parameters of the watermarking scheme consist of a public-key pk for a CCA secure public-key encryption scheme and the watermarking secret key is the corresponding decryption key sk. Let $\{f_s\}$ be an arbitrary puncturable PRF (pPRF) family. We are able to watermark a function family $\{F_k\}$ which is defined as follows:

- The key $k = (s, z, r)$ consists of a pPRF key s, a random pPRF input z and encryption randomness r.
- The function is defined as $F_k(x) = (f_s(x), \mathsf{ct})$ where $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}((f_s(z), z)\,;\, r)$.

Note that this is not yet a PRF family since the ct part of the output is always the same no matter what x is. However, the first part of the output ensures that the function is unpredictable. We now describe the marking and extraction procedures.

- To mark a function F_k with $k = (s, z, r)$ we create a key $\widetilde{k} = (s\{z\}, \mathsf{ct})$ where $s\{z\}$ is a PRF key which is punctured at the point z and $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}((f_s(z), z)\,;\, r)$. We define the marked function as $F_{\widetilde{k}}(x) = (f_{s\{z\}}(x), \mathsf{ct})$.
- The extraction procedure gets a circuit C and let $C(x) = (C_1(x), C_2(x))$ denote the first and second part of the output respectively. The extraction procedure computes $C_2(x_i) = \mathsf{ct}_i$ for many random values x_i and attempts to decrypt $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_i) = (y_i, z_i)$. If for at least one i the decryption succeeds and it holds that $C_1(z_i) \neq y_i$ then the procedure outputs marked, else it outputs $\mathsf{unmarked}$.

There are several properties to check. Firstly, note that marking procedure does not require any secret keys and that the marked function satisfies $F_k(x) = F_{\widetilde{k}}(x)$ for all $x \neq z$. In other words, the marking procedure only introduces only a single difference at a random point z.[3] Secondly, for any function F_k in the family which was not marked, the extraction procedure correctly outputs $\mathsf{unmarked}$ and for any function $F_{\widetilde{k}}(x)$ that was marked it correctly outputs marked.[4]

To argue that marks are unremovable, assume that we choose a random function F_k in the family, mark it, and give the adversary the marked function

[2] A function family is unpredictable if, given arbitrarily many oracle calls to a random function from the family on various inputs x_i, it is hard to predict the output of the function on any fresh input x^* which was not queried.

[3] Moreover, we show the following. Given oracle access to a random unmarked function $F_k(\cdot)$ and its marked version $F_{\widetilde{k}}(\cdot)$ as well as the extraction oracle, it is hard to find the point z on which they differ.

[4] Moreover, we can also ensure that any a-priori chosen circuit C is unmarked with high probability over the keys of the watermarking scheme. To achieve this we rely on an encryption scheme where legitimate ciphertexts are sparse.

$F_{\widetilde{k}}$ with $\widetilde{k} = (s\{z\}, \mathsf{ct})$. The adversary produces some circuit C which he gives to the extraction procedure and he wins if C agrees with $F_{\widetilde{k}}$ on a sufficiently large fraction of inputs but the extraction procedure deems C to be unmarked. If C agrees with $F_{\widetilde{k}}$ on a sufficiently large fraction of inputs then with very high probability for at least one x_i queried by the extraction procedure it holds that $C_2(x_i) = \mathsf{ct}$ and ct decrypts to $(f_s(z), z)$. In order for the extraction procedure to output $\mathsf{unmarked}$ it would have to hold that $C_1(z) = f_s(z)$ meaning that the adversary can predict $f_s(z)$. But the adversary only has a punctured pPRF key $s\{z\}$ and therefore it should be hard to predict $f_s(z)$. This argument is incomplete since the adversary also has a ciphertext ct which encrypts $f_s(z)$ and oracle access to the extraction procedure which contains a decryption key sk. To complete the argument, we rely on the CCA security of the encryption scheme to argue that extraction queries do not reveal any information about $f_s(z)$ beyond allowing the adversary to test whether $f_s(z) = y$ for various chosen values y and this is insufficient to predict $f_s(z)$.

Watermarking Pseudorandom Functions. To get a watermarking scheme for a pseudorandom function family rather than just an unpredictable one we add an additional "outer" layer of encryption. We need the outer encryption to be a"pseudorandom tagged CCA encryption" which ensures that a ciphertext encrypting some message m under a tag x looks random even given decryption queries with respect to any tags $x' \neq x$. The public parameters consist of an outer public key pk' for the "pseudorandom tagged CCA encryption" and an inner public key pk for the standard CCA encryption. The watermarking secret key consists of the decryption keys $\mathsf{sk}', \mathsf{sk}$.

We define the PRF family $\{F_k\}$ as follows:

- The key $k = (s, z, r, s')$ consists of a pPRF key s, a random pPRF input z and encryption randomness r as before. We now also include an additional PRF key s'.
- The function is defined as $F_k(x) = (f_s(x), \mathsf{ct}')$ where $\mathsf{ct}' = \mathsf{Enc}'_{\mathsf{pk}',x}(\mathsf{ct}\,;\, f_{s'}(x))$ is an encryption of ct with respect to the tag x using randomness $f_{s'}(x)$ and $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(f_s(z), z\,;\, r)$ as before. Note that the inner ciphertext ct is always the same but the outer ciphertext ct' is different for each x.

The watermarking scheme is almost the same as before except that:

- To mark a function F_k with $k = (s, z, r, s')$ we create a key $\widetilde{k} = (s\{z\}, \mathsf{ct}, s')$ where $s\{z\}$ is a pPRF key which is punctured at the point z and $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(f_s(z), z\,;\, r)$ as before. We define the marked function as $F_{\widetilde{k}}(x) = (f_{s\{z\}}(x), \mathsf{ct}')$ where $\mathsf{ct}' = \mathsf{Enc}'_{\mathsf{pk}',x}(\mathsf{ct}\,;\, f_{s'}(x))$.
- The extraction procedure is the same as before except that it also peels off the outer layer of encryption.

We now argue that the function family $\{F_k\}$ is pseudorandom even given the public parameter $(\mathsf{pk}, \mathsf{pk}')$ of the watermarking scheme and access to the extraction oracle. However, note that given the watermarking secret key $\mathsf{sk}, \mathsf{sk}'$,

it is easy to completely break PRF security by testing if the outer ciphertexts decrypts correctly to the same value every time. To argue pseudorandomness, we rely on the security of the outer encryption. Note that the outer ciphertexts are tagged with various tags x_i corresponding to the adversary's PRF queries. However, the extraction oracle only decrypts with respect to tags x'_i corresponding to random inputs that it chooses on each extraction query. Therefore, with exponentially small probability there will be an overlap between the values x'_i and x_i, and thus we can switch all of the ciphertexts returned by the PRF queries with uniformly random values.

Watermarking with Message Embedding. Our watermarking construction that allows to embed a message $\mathsf{msg} \in \{0,1\}^\ell$ during marking is very similar to the non-message embedding one. The main difference is that we use a constraint-hiding constrained PRF (CHC-PRF) to embed a hidden pattern that allows the extraction procedure to recover the message. At a high level, to mark a key with some message msg, we consider for each message bit msg_j a sparse pseudorandom set V_j; and we constrain the key on V_j if $\mathsf{msg}_j = 1$. We use an additional set V_0 on which we always constrain when marking a key. Each set V_j is defined using a fresh PRF key t_j. The public parameters and the watermarking secret key are the same as before, but now our PRF key k grows linearly with the message length.

Let $\{f_s\}$ be an arbitrary constraint-hiding constrained PRF (CHC-PRF) family. We define the PRF family $\{F_k\}$ as follows:

- The key $k = (s, (t_0, t_1, \ldots, t_\ell), r, s')$ consists of a CHC-PRF key s, $\ell + 1$ PRF keys $\{t_i\}_{i \le \ell}$, encryption randomness r, and a PRF key s'.
- The function is defined as $F_k(x) = (f_s(x), \mathsf{ct}')$ where $\mathsf{ct}' = \mathsf{Enc}'_{\mathsf{pk}',x}(\mathsf{ct}; f_{s'}(x))$ is an encryption of ct with respect to the tag x using randomness $f_{s'}(x)$ and $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(s, (t_0, t_1, \ldots, t_\ell)\,; r)$. Again, the inner ciphertext ct is always the same but the outer ciphertext ct' is different for each x.

The marking and extraction procedures work as follows:

- To mark a function F_k with $k = (s, (t_0, t_1, \ldots, t_\ell), r, s')$ and message $\mathsf{msg} \in \{0,1\}^\ell$, we first define the following circuit C_{msg}. For each key t_j, let C_j be the circuit which on input $x = (a, b)$ accepts if $f_{t_j}(a) = b$. Here we implicitly define the set $V_j = \{(a, f_{t_j}(a))\}$, and thus the circuit C_j checks membership in V_j. We define C_{msg} as:

$$C_{\mathsf{msg}} = C_0 \vee \left(\bigvee_{\substack{j=1,\ldots,\ell \\ \mathsf{msg}_j = 1}} C_j \right),$$

so that C_{msg} checks membership in the union of V_0 and the V_j's for j with $\mathsf{msg}_j = 1$.

We create a key $\widetilde{k} = (s\{C_{\mathsf{msg}}\}, \mathsf{ct}, s')$ where $s\{C_{\mathsf{msg}}\}$ is a CHC-PRF key which is constrained on the circuit C_{msg} and $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(s, (t_0, t_1, \ldots, t_\ell)\,; r)$. We define the marked function as $F_{\widetilde{k}}(x) = (f_{s\{C_{\mathsf{msg}}\}}(x), \mathsf{ct}')$ where $\mathsf{ct}' = \mathsf{Enc}'_{\mathsf{pk}',x}(\mathsf{ct}\,;\, f_{s'}(x))$.

– The extraction procedure gets a circuit C and let $C(x) = (C_1(x), C_2(x))$ denote the first and second part of the output respectively. The extraction procedure computes $C_2(x_i) = \mathsf{ct}'_i$ for many random values x_i, peels off the outer layer to obtain ct_i, and attempts to decrypt $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_i) = (s, (t_0, t_1, \ldots, t_\ell))$. The extraction procedure selects the decrypted message $(s, (t_0, t_1, \ldots, t_\ell))$ which forms the majority of the decrypted messages. If such a majority doesn't exist, the extraction stops here and outputs $\mathsf{unmarked}$.
The procedure now samples many random values a_i, computes $x_i = (a_i, f_{t_0}(a_i)) \in V_0$ and tests if $C_1(x_i) \neq f_s(x_i)$. If for the majority of the values x_i it holds that $C_1(x_i) \neq f_s(x_i)$ then the procedure considers the circuit as marked and proceeds to extract a message, as described below; else it stops here and outputs $\mathsf{unmarked}$.
To extract a message $\mathsf{msg} \in \{0,1\}^\ell$ the procedure does the following:
It samples, for $j = 1, \ldots, \ell$, many random values $a_{j,i}$, computes the pseudorandom values $x_{j,i} = (a_{j,i}, f_{t_j}(a_{j,i})) \in V_j$, and checks if $C_1(x_{j,i}) \neq f_s(x_{j,i})$. If for the majority of the values $x_{j,i}$ it holds that $C_1(x_{j,i}) \neq f_s(x_{j,i})$ then it sets $\mathsf{msg}_j = 1$, otherwise sets $\mathsf{msg}_j = 0$.
It then outputs $\mathsf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\ell)$.

To show pseudorandomness of the function family $\{F_k\}$ even given the public parameters $(\mathsf{pk}, \mathsf{pk}')$, the same argument as in the non-message embedding family goes through. Moreover, for any function F_k in the family which was not marked, the extraction procedure correctly outputs $\mathsf{unmarked}$ (because of the checks on V_0). Furthermore, for any message msg any function $F_{\widetilde{k}}$ where $\widetilde{k} \leftarrow \mathsf{Mark}(k, \mathsf{msg})$, $\mathsf{Extract}(\mathsf{ek}, F_{\widetilde{k}})$ correctly outputs the original message msg. This is because with overwhelming probability, a random point in V_j is constrained if and only if $\mathsf{msg}_j = 1$, by pseudorandomness and sparsity of the V_j. Then, correctness of the CHC-PRF ensures that $\mathsf{Extract}$ computes msg_j correctly when $\mathsf{msg}_j = 0$ (as the marked key is not constrained on V_j in that case), while constrained pseudorandomness ensures correctness when $\mathsf{msg}_j = 1$ (as the marked key is then constrained on V_j).

For watermarking security, we let the adversary choose a message msg. We sample a key $k = (s, (t_0, t_1, \ldots, t_\ell), r, s')$ and give him $F_{\widetilde{k}}$ where $\widetilde{k} \leftarrow \mathsf{Mark}(k, \mathsf{msg})$. However, we now only allow the adversary to modify slightly less than half of the outputs of the marked challenge circuit $F_{\widetilde{k}}$. As shown by Cohen et al. [10], this restriction is necessary when considering watermarking schemes that allow message embedding. So now, the adversary is given a marked function $F_{\widetilde{k}}$, and produces some circuit C which agrees with $F_{\widetilde{k}}$ on more than half of its input values. We use similar arguments as in the non message embedding version, but now we additionally rely on the constraint-hiding property of the CHC-PRF to argue that the sets V_j remain pseudorandom for the adversary, even given the marked circuit $F_{\widetilde{k}}$.

Now, the extraction procedure $\mathsf{Extract}(\mathsf{ek}, C)$ samples sufficiently many random input values. Because C and $F_{\widetilde{k}}$ agree on more than half of their input values, then with overwhelming probability the majority of the random input values will agree on their output values in both C and $F_{\widetilde{k}}$ (by a standard Chernoff bound); in which case the extraction procedure recovers $(s, (t_0, \ldots, t_\ell))$. But then, by pseudorandomness of the sets V_j, we have by another Chernoff bound that with overwhelming probability, the majority of the input values sampled in V_j will agree on their output values in both C and $F_{\widetilde{k}}$. By the sparsity and pseudorandomness of the sets V_j, these input values are constrained in $F_{\widetilde{k}}$ if and only if $\mathsf{msg}_j = 1$; and the correctness and pseudorandomness of the CHC-PRF ensure that the extraction procedure outputs msg on input C with overwhelming probability.

2 Preliminaries

2.1 Notations

For any probablistic algorithm $\mathsf{alg}(\mathsf{inputs})$, we may explicit the randomness it uses by writing $\mathsf{alg}(\mathsf{inputs}; \mathsf{coins})$.

For two circuits C, D, and $\varepsilon \in [0, 1]$, we write $C \cong_\varepsilon D$ if C and D agree on an ε fraction of their inputs.

We will use the notations $\overset{\text{s}}{\approx}$ and $\overset{\text{c}}{\approx}$ to denote statistical and computational indistinguishability, respectively.

We will use the following lemma:

Lemma 1 (Chernoff Bound). *Let $X_1, \ldots X_n$ be independent Bernoulli variables of parameter $p \in [0, 1]$. Then for all $\varepsilon > 0$, we have:*

$$\Pr\left[\sum_{i=1}^{n} X_i < n \cdot (p - \varepsilon)\right] \leq e^{-2\varepsilon^2 n}.$$

In particular for $n = \lambda/\varepsilon^2$, this probability is exponentially small in λ.

2.2 Constrained PRFs

We recall the definition of two variants of contrained PRFs.

Definition 1 (Puncturable PRFs *[6, 7, 13, 15]***).** *Let $\ell_{in} = \ell_{in}(\lambda)$ and $\ell_{out} = \ell_{out}(\lambda)$ for a pair of polynomial-time computable functions $\ell_{in}(\cdot)$ and $\ell_{out}(\cdot)$. A puncturable pseudo-random function (pPRF) family is defined by the following algorithms:*

- $\mathsf{KeyGen}(1^\lambda)$ *takes as input the security parameter λ, and outputs a PRF key k.*
- $\mathsf{Eval}(k, x)$ *takes as input a key k and an input $x \in \{0,1\}^{\ell_{in}}$ and deterministically outputs a value $y \in \{0,1\}^{\ell_{out}}$.*

- $\mathsf{Puncture}(k, z)$ *takes as input a key* k *and an input* $z \in \{0,1\}^{\ell_{in}}$*, and outputs a punctured key* $k\{z\}$*.*
- $\mathsf{PunctureEval}(k\{z\}, x)$ *takes as input a constrained key* $k\{z\}$ *and an input* $x \in \{0,1\}^{\ell_{in}}$*, and outputs a value* $y \in \{0,1\}^{\ell_{out}}$*.*

We require a puncturable PRF to satisfy the following properties:

Functionality preserving under puncturing. Let $z, x \in \{0,1\}^{\ell_{in}}$ *such that* $x \neq z$*. Then:*

$$\Pr\left[\mathsf{Eval}(k,x) = \mathsf{PunctureEval}(k\{z\},x) \;\middle|\; \begin{array}{rl} k & \leftarrow \mathsf{KeyGen}(1^\lambda) \\ k\{z\} & \leftarrow \mathsf{Puncture}(k,z) \end{array}\right] = 1.$$

Pseudorandomness on punctured points. For all $z \in \{0,1\}^{\ell_{in}}$*, we have that for all PPT adversary* $\mathcal{A}$*:*

$$|\Pr\left[\mathcal{A}(k\{z\}, \mathsf{Eval}(k,z)) = 1\right] - \Pr\left[\mathcal{A}(k\{z\}, \mathcal{U}_{\ell_{out}}) = 1\right]| \leq \mathrm{negl}(\lambda),$$

where $k \leftarrow \mathsf{KeyGen}(1^\lambda)$*,* $k\{z\} \leftarrow \mathsf{Puncture}(k,z)$ *and* $\mathcal{U}_{\ell_{out}}$ *denotes the uniform distribution over* ℓ_{out} *bits.*

We have constructions of puncturable PRFs assuming the existence of one-way functions [6,7,13,15].

Definition 2 ((Selective) Constraint-hiding Constrained PRFs). *Let* $\ell_{in} = \ell_{in}(\lambda)$ *and* $\ell_{out} = \ell_{out}(\lambda)$ *for a pair of polynomial-time computable functions* $\ell_{in}(\cdot)$ *and* $\ell_{out}(\cdot)$*. A constraint-hiding constrained pseudo-random function (CHC-PRF) family is defined by the following algorithms:*

- $\mathsf{KeyGen}(1^\lambda)$ *takes as input the security parameter* λ*, and outputs a PRF key* k*.*
- $\mathsf{Eval}(k, x)$ *takes as input a key* k *and an input* $x \in \{0,1\}^{\ell_{in}}$ *and deterministically outputs a value* $y \in \{0,1\}^{\ell_{out}}$*.*
- $\mathsf{Constrain}(k, C)$ *takes as input a key* k *and a binary circuit* $C : \{0,1\}^{\ell_{in}} \to \{0,1\}$*, and outputs a constrained key* k_C*.*
- $\mathsf{ConstrainEval}(k_C, x)$ *takes as input a constrained key* k_C *and an input* $x \in \{0,1\}^{\ell_{in}}$*, and outputs a value* $y \in \{0,1\}^{\ell_{out}}$*.*

We require the algorithms $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$ *to satisfy the following property, which captures the notions of constraint-hiding, (computational) functionality preserving and constrained pseudorandomness at the same time [9,19]:*

Selective Constraint-Hiding. Consider the following experiments between an adversary $\mathcal{A}$ *and a simulator* $\mathsf{Sim} = (\mathsf{Sim}^{\mathsf{key}}, \mathsf{Sim}^{\mathsf{ch}})$*:*

$\text{EXP}^{Real}_{CH}(1^\lambda)$: | $\text{EXP}^{Ideal}_{CH}(1^\lambda)$:

1. $C \leftarrow \mathcal{A}$ | 1. $C \leftarrow \mathcal{A}$
2. $k \leftarrow \mathsf{KeyGen}(1^\lambda)$ | 2.
3. $k_C \leftarrow \mathsf{Constrain}(k, C)$ | 3. $k_C \leftarrow \mathsf{Sim}^{\mathsf{key}}(1^{|C|})$
4. *Output* $b \leftarrow \mathcal{A}^{\mathsf{Eval}(\cdot)}(k_C)$ | 4. *Output* $b \leftarrow \mathcal{A}^{\mathsf{Sim}^{\mathsf{ch}}(\cdot)}(k_C)$

where $\mathsf{Sim}^{\mathsf{ch}}(\cdot)$ *is defined as:*

$$\mathsf{Sim}^{\mathsf{ch}}(x) = \begin{cases} R(x) & \text{if } C(x) = 1 \\ \mathsf{ConstrainEval}(k_C, x) & \text{if } C(x) = 0, \end{cases}$$

where $R : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell_{out}}$ *is a random function.*

We say that $\mathcal{F}$ *is a* constraint-hiding contrained PRF *if:*

$$\left|\Pr\left[\text{EXP}^{Real}_{CH}(1^\lambda) = 1\right] - \Pr\left[\text{EXP}^{Ideal}_{CH}(1^\lambda) = 1\right]\right| \leq \text{negl}(\lambda).$$

There are several constructions of constraint-hiding constrained PRFs under LWE [4,8,9,19].

2.3 Tag-CCA Encryption with Pseudorandom Ciphertexts

Definition 3 (Tag-CCA2 Encryption with Pseudorandom Ciphertexts).

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme with the following syntax:*

- $\mathsf{KeyGen}(1^\lambda)$ *takes as input the security parameter* λ *and outputs keys* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}_{\mathsf{pk},t}(m)$ *takes as input the public key* pk*, a message* m *and a tag* t*, and outputs a ciphertext* ct.
- $\mathsf{Dec}_{\mathsf{sk},t}(\mathsf{ct})$ *takes as input the secret key* sk*, a ciphertext* ct *and a tag* t*, and outputs a message* m.

We will in the rest of the paper omit the keys as arguments to Enc *and* Dec *when they are clear from the context.*

We will consider for simplicity perfectly correct schemes, so that for all messages m *and tag* t*:*

$$\Pr[\mathsf{Dec}_{\mathsf{sk},t}(\mathsf{Enc}_{\mathsf{pk},t}(m)) = m] = 1.$$

over the randomness of KeyGen*,* Enc *and* Dec.

Denote by $\mathcal{CT} = \mathcal{CT}_{\mathsf{pk}}$ *be the ciphertext space of* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*. For security, consider for* $b \in \{0,1\}$ *the following experiments* $\mathsf{Exp}^b_{tag-CCA2}(1^\lambda)$ *between a PPT adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

$\text{EXP}^0_{tag-CCA2}(1^\lambda)$: | $\text{EXP}^1_{tag-CCA2}(1^\lambda)$:

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ | 1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. $(m^*, t^*) \leftarrow \mathcal{A}^{\mathsf{Dec}_{\mathsf{sk},\cdot}(\cdot)}$ | 2. $(m^*, t^*) \leftarrow \mathcal{A}^{\mathsf{Dec}_{\mathsf{sk},\cdot}(\cdot)}$
3. $c^* \leftarrow \mathsf{Enc}_{\mathsf{pk},t^*}(m)$ | 3. $c^* \leftarrow \mathcal{CT}$
4. *Output* $b \leftarrow \mathcal{A}^{\mathsf{Dec}_{\mathsf{sk},\cdot}(\cdot)}$ | 4. *Output* $b \leftarrow \mathcal{A}^{\mathsf{Dec}_{\mathsf{sk},\cdot}(\cdot)}$

where $\mathsf{Dec}_{\mathsf{sk},\cdot}(\cdot)$ *takes as input a tag* t *and a ciphertext* c*, and outputs* $\mathsf{Dec}_{\mathsf{sk},t}(c)$ *We say that* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is tag-CCA2 with pseudorandom ciphertexts if for all PPT* $\mathcal{A}$ *who do not make any query of the form* $(t^*, *)$ *to the decryption oracle in phases* 2 *and* 4*:*

$$\left|\Pr[\mathsf{Exp}^0(1^\lambda) = 1] - \Pr[\mathsf{Exp}^1(1^\lambda) = 1]\right| \leq \mathrm{negl}(\lambda).$$

Notice that the notion of tag-CCA2 encryption with pseudorandom ciphertexts is weaker than both CCA2 encryption with pseudorandom ciphertexts, and fully secure Identity-Based Encryption (IBE) with pseudorandom ciphertexts. To see that CCA2 schemes with pseudorandom ciphertexts imply their tag-CCA2 counterpart, notice that it suffices to encrypt the tag along with the message. Then, make the decryption output $\perp$ if the decrypted tag part does not match the decryption tag. IBEs with pseudorandom ciphertexts also directly imply a tag-CCA2 version by simply considering identities as tags.

In particular, we have construction of tag-CCA2 schemes with pseudorandom ciphertexts under various assumptions, e.g. DDH, DCR, QR [11], or LWE [1].

We will need an additional property on the encryption scheme, namely that its ciphertexts are *sparse*:

Definition 4 (Sparsity of Ciphertexts). *We say that an encryption scheme is* sparse *if for all* ct *from the ciphertext space, and all tags* t*:*

$$\Pr[\mathsf{Dec}_{\mathsf{sk},t}(\mathsf{ct}) \neq \perp \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)] \leq \mathrm{negl}(\lambda).$$

Note that we can build a sparse tag-CCA2 encryption scheme with pseudorandom ciphertexts generically from any tag-CCA2 encryption scheme with pseudorandom ciphertexts. To do so, it suffices to add a random identifier $\alpha \in \{0,1\}^\lambda$ to the public key; to encrypt some message m, encrypt instead the message (m, α) using the non-sparse encryption scheme. Then, when decrypting, output $\perp$ if the identifier α does not match. For any fixed ct, the probability that it decrypts under the new encryption scheme is negligible over the randomness of α (sampled during KeyGen).

3 Watermarking PRFs

In this section, we construct a watermarking scheme and its associated watermarkable PRF family. The marking procedure is public, and security holds even when the attacker has access to an extraction oracle. We can instantiate the primitives we require under different various assumptions, e.g. DDH, LWE, or Factoring. We do not consider the case of embedding messages in the marked circuit yet though; the extraction algorithm here simply detects if the key has been marked or not. We will study the case of message embedding in Sect. 4.

3.1 Definitions

We first define the notion of watermaking. We tailor our notation and definitions to implicitly consider the setting where marking is public and extraction is secret.[5]

Definition 5 (Watermarking Scheme). *Let $\lambda \in \mathbb{N}$ be the security parameter and $\varepsilon \in [0,1]$ be a parameter. A watermarking scheme* WatMk *for a watermarkable family of pseudorandom functions $\mathcal{F} = \{\mathcal{F}_{\mathsf{pp}} : \mathcal{X}_{\mathsf{pp}} \to \mathcal{Y}_{\mathsf{pp}}\}_{\mathsf{pp}}$ is defined by the following polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{ek})$*: On input the security parameter 1^λ, outputs the public parameters* pp *and the extraction key* ek*.*
- $\mathsf{KeyGen}(1^\lambda, \mathsf{pp}) \to k$*: On input the security parameter 1^λ and public parameters* pp*, outputs a PRF key k.*
- $F_k(x) \to y$*: On input a key k and an input $x \in \mathcal{X}_{\mathsf{pp}}$, outputs $y \in \mathcal{Y}_{\mathsf{pp}}$.*
- $\mathsf{Mark}(k) \to \widetilde{k}$*: On input and a PRF key $k \in \mathcal{F}$, outputs a marked key $\widetilde{k}$.*
- $\mathsf{Extract}(\mathsf{ek}, C) \to \{\mathsf{marked}, \mathsf{unmarked}\}$*: On input an extraction key* ek *and an arbitrary circuit C, outputs* marked *or* unmarked*.*

We will simply denote by F_k some circuit that computes $F_k(x)$ on input x (which is efficiently computable given k).

Definition 6 (Watermarking Properties). *A watermarking scheme* WatMk *has to satisfy the following properties:*

Non-triviality. *We require two properties of non-triviality.*

1. *We require that functions in $\mathcal{F}$ are unmarked:*

$$\Pr\left[\mathsf{Extract}(\mathsf{ek}, F_k) = \mathsf{unmarked} \;\middle|\; \begin{array}{l}(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda) \\ k \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp})\end{array}\right] = 1.$$

2. *Any fixed circuit C (fixed independently of* pp*) should be unmarked:*

$$\Pr\left[\mathsf{Extract}(\mathsf{ek}, C) = \mathsf{unmarked} \,|\, (\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)\right] \geq 1 - \mathrm{negl}(\lambda).$$

Strong Correctness. *It should be hard to find points on which F_k and $F_{\widetilde{k}}$ output different values, given oracle access to both circuits.*

For all PPT $\mathcal{A}$ we require:

$$\Pr\left[F_k(x) \neq F_{\widetilde{k}}(x) \;\middle|\; \begin{array}{l}(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda) \\ k \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp}) \\ \widetilde{k} \leftarrow \mathsf{Mark}(k) \\ x \leftarrow \mathcal{A}^{F_k(\cdot), F_{\widetilde{k}}(\cdot), \mathsf{Extract}(\mathsf{ek}, \cdot)}(\mathsf{pp})\end{array}\right] \leq \mathrm{negl}(\lambda).$$

[5] We can directly extend the following definitions to the weaker setting of *secret* marking, by additionally giving the adversary oracle access to the marking algorithm in the relevant properties.

In particular, for any fixed x, the probability that $F_k(x) \neq F_{\widetilde{k}}(x)$ is negligible.[6]

Extended Pseudorandomness. *We do not require PRF security to hold if the adversary is given the extraction key. We still require that the PRFs in the family remain secure even given oracle access to the extraction algorithm.*

We require that for all PPT $\mathcal{A}$:

$$\mathcal{A}^{F_k(\cdot),\,\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp}) \overset{c}{\approx} \mathcal{A}^{R(\cdot),\,\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp}),$$

where $(\mathsf{pp},\mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$, $k \leftarrow \mathsf{KeyGen}(1^\lambda,\mathsf{pp})$, and R is a random function.

ε-Unremovability. *Define the following experiment $\mathsf{Exp}^{\mathsf{remov}}_{\mathcal{A}}(1^\lambda)$ between an adversary $\mathcal{A}$ and a challenger:*

1. *The challenger generates $(\mathsf{pp},\mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$. It also samples a random $k \leftarrow \mathsf{KeyGen}(1^\lambda,\mathsf{pp})$, and gives the public parameters pp and a circuit $\widetilde{C} = F_{\widetilde{k}}$ to the adversary, where $\widetilde{k} \leftarrow \mathsf{Mark}(k)$.*
2. *The adversary $\mathcal{A}^{\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp},\widetilde{C})$ has access to an extraction oracle, which on input a circuit C, outputs $\mathsf{Extract}(\mathsf{ek},C)$.*
3. *The adversary $\mathcal{A}^{\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp},\widetilde{C})$ outputs a circuit C^*. The output of the experiment is 1 if $\mathsf{Extract}(\mathsf{ek},C^*) = \mathsf{unmarked}$; and the output of the experiment is 0 otherwise.*

We say that an adversary $\mathcal{A}$ is ε-admissible *if its output C^* in phase 3. satisfies $C^* \cong_\varepsilon \widetilde{C}$, i.e. C^* and $\widetilde{C}$ agree on an ε fraction of their inputs.*

We say that a watermarking scheme achieves ε-unremovability if for all ε-admissible PPT adversaries $\mathcal{A}$ we have:

$$\Pr[\mathsf{Exp}^{\mathsf{remov}}_{\mathcal{A}}(1^\lambda) = 1] \leq \mathrm{negl}(\lambda).$$

Extraction Correctness. We require that:

$$\Pr\left[\mathsf{Extract}(\mathsf{ek},F_{\widetilde{k}}) = \mathsf{marked} \;\middle|\; \begin{array}{l}(\mathsf{pp},\mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)\\ k \leftarrow \mathsf{KeyGen}(1^\lambda,\mathsf{pp})\\ \widetilde{k} \leftarrow \mathsf{Mark}(k)\end{array}\right] \geq 1 - \mathrm{negl}(\lambda),$$

but in this case this follows from ε-Unremovability, as otherwise an Adversary could just directly output the marked challenge in the ε-Unremovability game.

3.2 Construction

Let $\lambda \in \mathbb{N}$ be the security parameter and let $\varepsilon = 1/\mathrm{poly}(\lambda)$ be a parameter. We describe our construction of a watermarkable family $\mathcal{F}_{\mathsf{pp}}$ and its associated ε-unremovable watermarking scheme.

We will use the following primitives in our construction:

[6] In particular, this notion of correctness is stronger than simply requiring that the output of the original PRF and the marked version differ at most on a negligible fraction of inputs.

- $\mathcal{E}^{\mathsf{in}} = (\mathcal{E}^{\mathsf{in}}.\mathsf{KeyGen}, \mathsf{Enc}^{\mathsf{in}}, \mathsf{Dec}^{\mathsf{in}})$, a CCA2 secure public-key encryption scheme
- $\mathcal{E}^{\mathsf{out}} = (\mathcal{E}^{\mathsf{out}}.\mathsf{KeyGen}, \mathsf{Enc}^{\mathsf{out}}, \mathsf{Dec}^{\mathsf{out}})$, a sparse tag-CCA2 encryption scheme with pseudorandom ciphertexts
- $\mathsf{pPRF} = (\mathsf{pPRF.KeyGen}, \mathsf{pPRF.Eval}, \mathsf{Puncture}, \mathsf{PunctureEval})$, a puncturable PRF family
- $\mathsf{PRF} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$, a standard PRF family.

We will use the following notation:

- $r^{\mathsf{in}} = r^{\mathsf{in}}(\lambda)$ and $r^{\mathsf{out}} = r^{\mathsf{out}}(\lambda)$ are the number of random bits used by $\mathsf{Enc}^{\mathsf{in}}$ and $\mathsf{Enc}^{\mathsf{out}}$, respectively;
- $(\mathcal{X}, \mathcal{Y}^{(1)}) = (\mathcal{X}_{\mathsf{pp}}, \mathcal{Y}^{(1)}_{\mathsf{pp}})$ are the input and output spaces of pPRF, where we assume that $\mathcal{X}$ and $\mathcal{Y}^{(1)}$ are of size super-polynomial in λ;
- We'll suppose that PRF has input and output spaces $(\mathcal{X}, \{0,1\}^{r^{\mathsf{out}}}) = (\mathcal{X}_{\mathsf{pp}}, \{0,1\}^{r^{\mathsf{out}}})$;
- $\mathcal{CT} = \mathcal{CT}_{\mathsf{pp}}$ is the ciphertext space of $\mathcal{E}^{\mathsf{out}}$.
- We set the input space of our watermarkable PRF to be $\mathcal{X}$, and its output space to be $\mathcal{Y} = \mathcal{Y}^{(1)} \times \mathcal{CT}$. For $y \in \mathcal{Y}$, we will write $y = (y_1, y_2)$, where $y_1 \in \mathcal{Y}^{(1)}$ and $y_2 \in \mathcal{CT}$.

We now describe our construction of a watermarking scheme, with its associated watermarkable PRF family:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, sample $(\mathsf{pk}^{\mathsf{in}}, \mathsf{sk}^{\mathsf{in}}) \leftarrow \mathcal{E}^{\mathsf{in}}.\mathsf{KeyGen}(1^\lambda)$ and $(\mathsf{pk}^{\mathsf{out}}, \mathsf{sk}^{\mathsf{out}}) \leftarrow \mathcal{E}^{\mathsf{out}}.\mathsf{KeyGen}(1^\lambda)$. Output:

$$\mathsf{pp} = (\mathsf{pk}^{\mathsf{in}}, \mathsf{pk}^{\mathsf{out}});$$

$$\mathsf{ek} = (\mathsf{sk}^{\mathsf{in}}, \mathsf{sk}^{\mathsf{out}}).$$

- $\mathsf{KeyGen}(1^\lambda, \mathsf{pp})$: On input the security parameter 1^λ and the public parameters pp, sample $s \leftarrow \mathsf{pPRF.KeyGen}(1^\lambda)$, $s' \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$, $r \leftarrow \{0,1\}^{r^{\mathsf{in}}}$, and $z \leftarrow \mathcal{X}$. The key of the watermarkable PRF is:

$$k = (s, z, r, s', \mathsf{pp}).$$

For ease of notation, we will simply write $k = (s, z, r, s')$ when the public parameters pp are clear from the context.
- $F_k(x)$: On input a key k and input x, output

$$F_k(x) = \Big(\; f_s(x), \;\; \mathsf{Enc}^{\mathsf{out}}_x(\mathsf{pk}_{\mathsf{out}}, \mathsf{Enc}^{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, (f_s(z), z); r)\,;\, f'_{s'}(x)) \; \Big)$$

where $f_s(\cdot) = \mathsf{pPRF.Eval}(s, \cdot)$, $f'_{s'}(\cdot) = \mathsf{PRF.Eval}(s', \cdot)$, and $\mathsf{Enc}^{\mathsf{out}}$ encrypts $\mathsf{Enc}^{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, (f_s(z), z)\,;\, r)$ using tag x and randomness $f'_{s'}(x)$.
- $\mathsf{Mark}(k)$: On input a key $k = (s, z, r, s')$, do the following:
  - Puncture the key s at point z: $s\{z\} \leftarrow \mathsf{pPRF.Puncture}(s, z)$.
  - Compute $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, (f_s(z), z)\,;\, r)$.

- Output the marked key

$$\widetilde{k} = (s\{z\}, c^{\mathsf{in}}, s'),$$

where the associated evaluation circuit computes:

$$F_{\widetilde{k}}(x) = \left(\mathsf{PunctureEval}(s\{z\}, x)\,,\, \mathsf{Enc}^{\mathsf{out}}_{x}(\mathsf{pk}_{\mathsf{out}}, c^{\mathsf{in}}\,;\, f'_{s'}(x)) \right).$$

– $\mathsf{Extract}(\mathsf{ek}, C)$: Let $w = \lambda/\varepsilon = \mathrm{poly}(\lambda)$. On input the extraction key ek and a circuit C do the following:
  - If the input or output length of C do not match $\mathcal{X}$ and $\mathcal{Y}^{(1)} \times \mathcal{CT}$ respectively, output $\mathsf{unmarked}$.
  - For all $i \in [w]$ sample uniformly at random $x_i \leftarrow \mathcal{X}$, and do the following:
    * Parse $C(x_i) = (C_1(x_i), C_2(x_i))$ where $C_1(x_i) \in \mathcal{Y}^{(1)}$ and $C_2(x_i) \in \mathcal{CT}$.
    * Compute $c_i^{\mathsf{in}} = \mathsf{Dec}^{\mathsf{out}}_{\mathsf{sk}^{\mathsf{out}}, x_i}(C_2(x_i))$ (using secret key $\mathsf{sk}^{\mathsf{out}}$ and tag x_i);
    * If $c_i^{\mathsf{in}} \neq \bot$, compute $(y_i, z_i) = \mathsf{Dec}^{\mathsf{in}}_{\mathsf{sk}^{\mathsf{in}}}(c_i^{\mathsf{in}})$. If $C_1(z_i) \neq y_i$, output marked.
  - If the procedure does not output marked after executing the loop above, output $\mathsf{unmarked}$.

Note that when it is clear from the context, we will omit writing $\mathsf{pk}_{\mathsf{out}}, \mathsf{pk}_{\mathsf{in}}$.

3.3 Correctness Properties of the Watermarking Scheme

We first show that our watermarking scheme satisfies the non-triviality properties.

Claim (Non-triviality). Assume $\mathcal{E}^{\mathsf{in}}$ and $\mathcal{E}^{\mathsf{out}}$ are perfectly correct, and that $\mathcal{E}^{\mathsf{out}}$ is sparse. Then our watermarking scheme satisfies the non-triviality properties.

Proof. 1. Let $(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $k = (s, z, r, s') \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp})$; then $\mathsf{Extract}(\mathsf{ek}, F_k)$ always outputs $\mathsf{unmarked}$. This is because by perfect correctness of $\mathcal{E}^{\mathsf{in}}$ and $\mathcal{E}^{\mathsf{out}}$, we have that $(y_i, z_i) = (f_s(z), z)$ for all $i \in [w]$, and therefore $C_1(z_i) = y_i = f_s(z)$.

2. Fix a circuit $C = (C_1, C_2)$, and sample $(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$. By sparsity of $\mathcal{E}^{\mathsf{out}}$, we have that for all $x_i \in \mathcal{X}$, the probability that $c_i^{\mathsf{in}} := \mathsf{Dec}^{\mathsf{out}}_{\mathsf{sk}^{\mathsf{out}}, x_i}(C_2(x_i)) \neq \bot$ is negligible (over the randomness of $\mathsf{Setup}(1^\lambda)$ alone). In particular, taking a union bound over the $w = \mathrm{poly}(\lambda)$ points $\{x_i\}_{i \in [w]}$ sampled by $\mathsf{Extract}$, we have that $c_i^{\mathsf{in}} = \bot$ with overwhelming probability, and therefore

$$\Pr\left[\mathsf{Extract}(\mathsf{ek}, C) = \mathsf{unmarked} \,|\, (\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)\right] \geq 1 - \mathrm{negl}(\lambda).$$

Claim (Strong Correctness). Suppose pPRF is a punctured PRF, PRF is secure and $\mathcal{E}^{\mathsf{out}}$ is tag-CCA2 with pseudorandom ciphertexts. Then the watermarking scheme satisfies strong correctness.

Proof. We show that the view of the adversary is essentially independent of z.

First, notice that it suffices to argue strong correctness when the adversary $\mathcal{A}$ only has oracle access to $F_k(\cdot)$ but not the marked version $F_{\widetilde{k}}(\cdot)$. This is because if we have the seemingly weaker version of correctness where the adversary doesn't have oracle access to $F_{\widetilde{k}}(\cdot)$, we can simulate oracle access to $F_{\widetilde{k}}(\cdot)$ by simply forwarding the output of $F_k(\cdot)$ on the same input. Now, an adversary can only tell the difference if he makes a query on z, which breaks the weaker notion of correctness (with a polynomial loss equal to his number of PRF queries).

Therefore, we focus on proving

$$\Pr\left[F_k(x) \neq F_{\widetilde{k}}(x) \;\middle|\; \begin{array}{l}(\mathsf{pp},\mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)\\ k \leftarrow \mathsf{KeyGen}(1^\lambda,\mathsf{pp})\\ \widetilde{k} \leftarrow \mathsf{Mark}(k)\\ x \leftarrow \mathcal{A}^{F_k(\cdot),\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp})\end{array}\right] \leq \mathrm{negl}(\lambda).$$

We prove the claim by a sequence of hybrids.

Hybrid 0. In this hybrid, the adversary $\mathcal{A}$ has oracle access to $F_k(\cdot)$ and $\mathsf{Extract}(\mathsf{ek},\cdot)$ where $(\mathsf{pp},\mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $k = (s,z,r,s') \leftarrow \mathsf{KeyGen}(1^\lambda,\mathsf{pp})$.

Hybrid 1. We modify how PRF queries are answered. Now, instead of using $f'_{s'}(x)$ as randomness to encrypt $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(f_s(z),z\,;\,r)$ using $\mathsf{Enc}^{\mathsf{out}}$ with tag x, we pick a random function $R^{(1)} : \mathcal{X} \to \{0,1\}^{r^{\mathsf{out}}}$ and use $R^{(1)}(x)$ as the encryption randomness to output:

$$(f_s(x), \mathsf{Enc}^{\mathsf{out}}_x(c^{\mathsf{in}}\,;\,R^{(1)}(x)),$$

where the function $R^{(1)}$ is common across all the PRF queries.

Hybrid 2. Now we keep track of the PRF queries x from the adversary, as well as all the x_i's that are sampled during the calls to the extraction oracle. We abort the experiment if at any point there is some x that has been both queried by the adversary and sampled during an extraction call.

Hybrid 3. We now pick a random function $R^{(3)} : \mathcal{X} \to \mathcal{CT}$ and answer to PRF oracle queries x from the adversary with:

$$(f_s(x), R^{(3)}(x)).$$

Now, by functionality preserving under puncturing of pPRF, z is the only point such that $F_k(z) \neq F_{\widetilde{k}}(z)$. However the view of the adversary is independent of z, and therefore the probability that he outputs z is negligible, over the random choice of z (sampled during $\mathsf{KeyGen}(1^\lambda,\mathsf{pp})$).

We prove the indistinguishability of the hybrids in the next section, as our proof of extended pseudorandomness uses the same hybrids.

3.4 Security Properties of the Watermarking Scheme

Unremovability. We first prove that our construction is ε-unremovable (where $\varepsilon = 1/\mathrm{poly}(\lambda)$ is a parameter of our scheme).

Claim (ε-unremovability). Suppose $\mathcal{E}^{\mathsf{in}}$ is CCA2-secure, and f is a puncturable PRF. Then the watermarking scheme is ε-unremovable.

Proof. We prove the claim by a sequence of hybrids.

Hybrid 0. This is the ε-Unremovability game $\mathsf{Exp}^{\mathsf{remov}}_{\mathcal{A}}(1^\lambda)$.

Hybrid 1. We now change how extraction oracle queries are answered (including the call used to determine the output of the experiment). Let $k = (s, z, r, s') \leftarrow \mathsf{PRF}_{\mathsf{pp}}.\mathsf{KeyGen}(1^\lambda, \mathsf{pp})$ be the (unmarked) PRF key sampled to produce the challenge marked circuit, and $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(s, z\,;\, r)$ be the associated ciphertext (which is used to produce the challenge marked circuit $\widetilde{C}$). On extraction query C from the adversary, the extraction procedure samples x_i's for $i \in [w]$ as before. Denote by E the event that $\mathsf{Dec}^{\mathsf{out}}_{\mathsf{sk}^{\mathsf{out}},x_i}(C_2(x_i)) = c^{\mathsf{in}}$, i.e. the second part $C_2(x_i)$ decrypts to c^{in} when decrypting using tag x_i. If E occurs, then instead of decrypting this inner ciphertext c^{in} in the extraction procedure, we directly check $C_1(z) \neq f_s(z)$; in particular c^{in}, z and $f_s(z)$ are now hard-coded in the modified extraction procedure.

Hybrid 2. We change how extraction calls are answered and how the challenge marked circuit is generated. Let $0_{\mathcal{X}}$ and $0_{\mathcal{CT}}$ be arbitrary fixed values in $\mathcal{X}$ and $\mathcal{CT}$ respectively. We now set

$$c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(0_{\mathcal{X}}, 0_{\mathcal{CT}}),$$

which is used as the ciphertext hard-coded in the extraction oracle (used to handle event E), and used to produce challenge marked circuit $\widetilde{C}$.

Hybrid 3. We change how we answer extraction queries (including the one determining the output of the experiment). Now pick a uniformly random $R \in \mathcal{Y}^{(1)}$. Whenever E occurs during an extraction oracle call, we check $C_1(z) \neq R$ instead. In particular, the modified extraction oracle now has $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(0_{\mathcal{X}}, 0_{\mathcal{CT}})$, z, and R hard-coded.

Hybrid 4. Now if there is any extraction oracle call such that E occurs and $C_1(z) = R$, we abort the experiment.

Now, all the outputs of the extraction oracle queries are independent of R, as R only affects the output of extraction queries only when E occurs, and the extraction oracle queries now outputs marked whenever there exists some index i such that E occurs, independently of R. Recall that the adversary wins the game if he outputs a circuit C^* such that $C^* \cong_\varepsilon \widetilde{C}$ and $\mathsf{Extract}(\mathsf{ek}, C^*) = \mathsf{unmarked}$. By construction, we have that during the execution of $\mathsf{Extract}(\mathsf{ek}, C^*)$ that defines the output of the experiment, $\mathsf{Extract}$ samples at least one x_i such that $C^*(x_i) = \widetilde{C}(x_i)$ with overwhelming probability. This is because C^* and $\widetilde{C}$ agree on a fraction $\varepsilon = 1/\mathrm{poly}(\lambda)$ of inputs, so that the probability that none of the $w = \lambda/\varepsilon$ samples x_i's satisfies $C^*(x_i) = \widetilde{C}(x_i)$ is at most $(1-\varepsilon)^{\lambda/\varepsilon} \leq e^{-\lambda} = \mathrm{negl}(\lambda)$. Now by correctness of the outer encryption scheme $\mathcal{E}^{\mathsf{out}}$, we have $\mathsf{Dec}^{\mathsf{out}}_{\mathsf{sk}^{\mathsf{out}},x_i}(C^*(x_i)) = c^{\mathsf{in}}$, so that event E occurs, and $\mathsf{Extract}$ outputs $\mathsf{unmarked}$ only if $C^*_1(z) = R$. As the view of the adversary in the experiment is now

independent of R, the experiment outputs marked with overwhelming probability (over the randomness of R alone).

Indistinguishability of the Hybrids. We now show that the hybrids above are indistinguishable.

Lemma 2. *Assuming $\mathcal{E}^{\mathsf{in}}$ is perfectly correct, we have* **Hybrid 0 $\equiv$ Hybrid 1**.

The view of the adversary is identical in Hybrid 0 and Hybrid 1 by perfect correctness of the inner encryption $\mathcal{E}^{\mathsf{in}}$: in the latter we simply hardcode the result of the decryption whenever we have to decrypt c^{in} during an extraction oracle call.

Lemma 3. *Assuming $\mathcal{E}^{\mathsf{in}}$ is CCA2-secure, we have* **Hybrid 1 $\overset{\text{c}}{\approx}$ Hybrid 2**.

We build a reduction that turns any distinguisher between Hybrid 1 and Hybrid 2 to a CCA2 adversary for $\mathcal{E}^{\mathsf{in}}$. The reduction essentially does not pick any secret key for $\mathsf{Enc}^{\mathsf{in}}$ but can still answer extraction oracle queries by interacting with the CCA2 challenger. More precisely, the reduction does not sample the secret key $\mathsf{sk}^{\mathsf{in}}$ associated to the CCA2 scheme $\mathcal{E}^{in}$, but samples the other parts of $(\mathsf{pp}, \mathsf{ek})$ as in Hybrid 1. It then sends CCA2 challenge messages $(f_s(z), z)$, and $(0_{\mathcal{X}}, 0_{\mathcal{CT}})$, and gets back a challenge ciphertext c^{in}, and sets the challenge circuit as $F_{\widetilde{k}}$ where $\widetilde{k} = (s\{z\}, c^{\mathsf{in}}, s')$. To answer extraction oracle queries for the distinguisher, it uses the CCA2 challenger to get the decryption of any $c \neq c^{\mathsf{in}}$ (which correspond to sampling x_i and event E does not occur); and whenever E occurs (which correspond to having $c_i^{\mathsf{in}} = c^{\mathsf{in}}$), it uses the hard-coded values $(f_s(z), z)$ to produce the output of the extraction oracle, by checking if $C_1(z) \neq f_s(z)$ directly without decrypting c^{in}. Now if c^{in} is an encryption of $(f_s(z), z)$, the view of the distinguisher is as in Hybrid 1; and if c^{in} is an encryption of $(0_{\mathcal{X}}, 0_{\mathcal{CT}})$ then its view is as in Hybrid 2.

Lemma 4. *Assuming* pPRF *satisfies constrained pseudorandomness, we have* **Hybrid 2 $\overset{\text{c}}{\approx}$ Hybrid 3**.

This is done by a simple reduction to the constrained pseudorandomness property of pPRF: the reduction samples some random z and gets a constrained key $s\{z\}$ from the constrained pseudorandomness game. Then, it gets a value y^*, which is used whenever event E occurs, to check $C_1(z) \neq y^*$. If $y^* = f_s(z)$, the view of the adversary is as in Hybrid 2; if y^* is random, then his view is as in Hybrid 3.

Lemma 5. *We have* **Hybrid 3 $\overset{\text{s}}{\approx}$ Hybrid 4**.

For any C queried by the adversary as an extraction oracle query, the probability that E occurs and $C_1(z) = R$ is negligible over the randomness of R alone (where we use that $\mathcal{Y}^{(1)}$ has super-polynomial size). Therefore, with overwhelming probability, all extraction oracle queries where E occurs output marked, independently of R. In particular, an union bound over the polynomial number of extraction queries made by the adversary gives that the probability that the experiment aborts is negligible.

Extended Pseudorandomness. Next, we show that our construction satisfies the extended pseudorandomness property.

Claim (Extended Pseudorandomness). Suppose pPRF and PRF are secure and $\mathcal{E}^{\mathsf{out}}$ is tag-CCA2 with pseudorandom ciphertexts. Then the watermarking scheme satisfies extended pseudorandomness.

Proof. We prove the claim by a sequence of hybrids.

Hybrid 0. In this hybrid, the adversary $\mathcal{A}$ has oracle access to $F_k(\cdot)$ and $\mathsf{Extract}(\mathsf{ek}, \cdot)$ where $(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $k = (s, z, r, s') \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp})$.

Hybrid 1. We modify how PRF queries are answered. Now, instead of using $f'_{s'}(x)$ as randomness to encrypt $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(f_s(z), z; r)$ with tag x, we pick a random function $R^{(1)} : \mathcal{X} \to \{0,1\}^{r^{\mathsf{out}}}$ and use $R^{(1)}(x)$ as randomness, and output:

$$(f_s(x), \mathsf{Enc}^{\mathsf{out}}_x(c^{\mathsf{in}}; R^{(1)}(x)),$$

where the function $R^{(1)}$ is common throughout the experiment.

Hybrid 2. Now we keep track of the PRF queries x from the adversary, as well as all the x_i's that are sampled during the calls to the extraction oracle. We abort the experiment if at any point there is some x that has been both queried by the adversary and sampled during an extraction call.

Hybrid 3. We now pick a random function $R^{(3)} : \mathcal{X} \to \mathcal{CT}$ and answer to PRF oracle queries x from the adversary with:

$$(f_s(x), R^{(3)}(x)).$$

Hybrid 4. We now additionally pick a random function $R^{(4)} : \mathcal{X} \to \mathcal{Y}^{(1)}$, and answer to PRF oracle queries x from the adversary with:

$$(R^{(4)}(x), R^{(3)}(x)).$$

Hybrid 5. Now we do not abort the experiment even if some x is both queried by the adversary and sampled during an extraction call.

Now the adversary has oracle access to $R(\cdot) = (R^{(4)}(\cdot), R^{(3)}(\cdot))$ and $\mathsf{Extract}(\mathsf{ek}, \cdot)$.

Indistinguishability of the Hybrids. We now show that the hybrids above are indistinguishable.

Lemma 6. *Assuming the security of* PRF*, we have* **Hybrid 0** $\overset{\mathrm{c}}{\approx}$ **Hybrid 1**.

We build a reduction from any distinguisher to an attacker for the PRF security game for PRF. On PRF query x from the distinguisher, the reduction queries x in the PRF game, and uses the answer as encryption randomness for the outer scheme $\mathcal{E}^{\mathsf{out}}$. If the value is $f'_{s'}(x)$, the view of the distinguisher is as in Hybrid 0; if it is random $R^{(1)}(x)$ then its view is as in Hybrid 1.

Lemma 7. *We have* **Hybrid 1** $\overset{s}{\approx}$ **Hybrid 2**.

We argue that the probability that the experiment aborts is negligible.

Suppose that some x has been both queried by the adversary as a PRF query, and sampled during an extraction oracle call.

If it has been sampled by the extraction procedure after the adversary queried it, this means that the extraction procedure sampled an x_i that the adversary queried previously, which happens with probability at most $Q^{PRF}/|\mathcal{X}|$, where Q^{PRF} is the number of PRF queries the adversary makes. An union bound on the polynomial number of samples used in every extraction call and the polynomial number of extraction calls imply that the probability that this event happens is negligible (where we use that $\mathcal{X}$ has super-polynomial size).

Otherwise, it means that the adversary queries an x_i that has previously been sampled by the extraction procedure. However, each output of the extraction oracle leaks at most 1 bit of entropy on the fresh x_i's it sampled during its execution. Therefore the adversary can only succeed in outputting such an x_i with negligible probability.

Lemma 8. *Assuming* $\mathcal{E}^{\mathsf{out}}$ *is a tag-CCA2 encryption scheme with pseudorandom ciphertexts, then* **Hybrid 2** $\overset{c}{\approx}$ **Hybrid 3**.

We replace the right part $\mathsf{Enc}_x^{\mathsf{out}}(c^{\mathsf{in}}; R^{(1)}(x))$ of the outputs to every PRF queries with $R^{(3)}(x)$ for some random $R^{(3)}$, one by one, using a hybrid argument.

To change the output to some query x^*, we reduce any distinguisher using our assumption on $\mathcal{E}^{\mathsf{out}}$. The reduction answers extraction queries using the decryption oracle provided by the tag-CCA2 game, and sends as a challenge message $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(f_s(z), z\,; r)$ and challenge tag x^*, and uses the challenge ciphertext from the tag-CCA2 game as a right part of the output to the PRF query on x^*. As we make our experiment abort if an extraction call uses some tag that is queried at any point by the distinguisher, we never have to decrypt any ciphertext with tag x^*, so that we can faithfully answer all the extraction queries by using the decryption oracle from the tag-CCA2 game. Note that we have to change the output of all the PRF queries on x^* in this hybrid.

If the challenge ciphertext from the tag-CCA2 game is a proper encryption of c^{in} under tag x^*, then the view of the distinguisher is as in Hybrid 2; and if it is random, then its view is as in Hybrid 3.

Lemma 9. *Assuming the security of* pPRF*, we have* **Hybrid 3** $\overset{c}{\approx}$ **Hybrid 4**.

We reduce any distinguisher to an attacker for the PRF security game. Our reduction, on PRF query x from the distinguisher, forwards it as a query in the PRF game. If it receives PRF values $f_s(x)$, the view of the distinguisher is as in Hybrid 3; if it receives a random $R^{(4)}(x)$, the view of the distinguisher is as in Hybrid 4.

Lemma 10. *We have* **Hybrid 4** $\overset{s}{\approx}$ **Hybrid 5**.

The same argument as to prove Hybrid 1 $\overset{s}{\approx}$ Hybrid 2 applies here.

4 Watermarking PRFs with Message-Embedding

In this section we describe our construction of a watermaking scheme that supports message embedding. Our construction is very similar to the non message embedding version: the main difference is that we now use a constraint-hiding constrained PRF as a base PRF.

4.1 Definitions

Let $\lambda \in \mathbb{N}$ be the security parameter, $\varepsilon \in [0,1]$ and $\ell = \ell(\lambda)$ be parameters. We make a few syntactical changes to the notions introduced in Sect. 3.1 when considering message-embedding watermarking schemes:

- $\mathsf{Mark}(k, \mathsf{msg}) \to \widetilde{k}$: On input a key k and a message $\mathsf{msg} \in \{0,1\}^\ell$, outputs a marked $\widetilde{k}$;
- $\mathsf{Extract}(\mathsf{ek}, C) \to \mathsf{msg}$: On input an extraction key ek and an arbitrary circuit C, outputs a message $\mathsf{msg} \in \{0,1\}^\ell \cup \{\mathsf{unmarked}\}$.
- Strong correctness: The adversary can now adaptively choose which message to mark.
  For all PPT $\mathcal{A}$ we require:

$$\Pr\left[F_k(x) \neq F_{\widetilde{k}}(x) \;\middle|\; \begin{array}{l}(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda) \\ k \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp}) \\ \mathsf{msg}^* \leftarrow \mathcal{A}^{F_k(\cdot), \mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp}) \\ \widetilde{k} \leftarrow \mathsf{Mark}(k, \mathsf{msg}^*) \\ x \leftarrow \mathcal{A}^{F_k(\cdot), F_{\widetilde{k}}(\cdot), \mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp})\end{array}\right] \leq \mathrm{negl}(\lambda).$$

- ε-unremovability: the adversary now additionally chooses some message msg^* given oracle access to the extraction procedure, and wins if he produces a circuit C^* that is ε-close to the marked challenge circuit such that $\mathsf{Extract}(\mathsf{ek}, C^*) \neq \mathsf{msg}^*$, as described by the following experiment $\mathsf{Exp}^{\mathsf{remov}-\mathsf{msg}}(1^\lambda)$:
  1. The challenger generates $(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$. It also samples a random $k \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp})$, and gives the public parameters pp to the adversary.
  2. The adversary computes a challenge message $\mathsf{msg}^* \in \{0,1\}^\ell \leftarrow \mathcal{A}^{\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp})$, given access to an extraction oracle, which on input a circuit C, outputs $\mathsf{Extract}(\mathsf{ek}, C)$.
  3. The challenger computes $\widetilde{C} \leftarrow \mathsf{Mark}(k, \mathsf{msg}^*)$ and sends it to the adversary.
  4. The adversary $\mathcal{A}^{\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp}, \widetilde{C})$ can make further extraction oracle queries.
  5. The adversary $\mathcal{A}^{\mathsf{Extract}(\mathsf{ek},\cdot)}(\mathsf{pp}, \widetilde{C})$ outputs a circuit C^*. The output of the experiment is 1 if $\mathsf{Extract}(\mathsf{ek}, C^*) \neq \mathsf{msg}^*$; and the output of the experiment is 0 otherwise.

We now say that an adversary $\mathcal{A}$ is ε-*admissible* if its output C^* in phase 5. satisfies $C^* \cong_\varepsilon \widetilde{C}$.

We say that a watermarking scheme achieves ε-unremovability if for all ε-admissible PPT adversaries $\mathcal{A}$ we have:

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{remov-msg}}(1^\lambda) = 1] \leq \mathrm{negl}(\lambda).$$

– Extraction correctness: we could now require that for all message $\mathsf{msg} \in \{0,1\}^\ell$:

$$\Pr\left[\mathsf{Extract}(\mathsf{ek}, \mathsf{Mark}(k, \mathsf{msg})) = \mathsf{msg} \;\middle|\; \begin{matrix}(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda) \\ k \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp})\end{matrix}\right] \geq 1 - \mathrm{negl}(\lambda),$$

but again, this property follows from ε-Unremovability.

4.2 Construction

Let $\lambda \in \mathbb{N}$ be the security parameter, let $\rho = 1/\mathrm{poly}(\lambda)$, and $\ell = \mathrm{poly}(\lambda)$ be parameters. Let $\varepsilon = 1/2 + \rho$. We describe our construction of a watermarkable family $\mathcal{F}_{\mathsf{pp}}$ and its associated ε-unremovable watermarking scheme supporting the embedding of messages of length ℓ.

We'll use the following primitives in our construction:

– $\mathcal{E}^{\mathsf{in}} = (\mathcal{E}^{\mathsf{in}}.\mathsf{KeyGen}, \mathsf{Enc}^{\mathsf{in}}, \mathsf{Dec}^{\mathsf{in}})$, a CCA2 secure public-key encryption scheme
– $\mathcal{E}^{\mathsf{out}} = (\mathcal{E}^{\mathsf{out}}.\mathsf{KeyGen}, \mathsf{Enc}^{\mathsf{out}}, \mathsf{Dec}^{\mathsf{out}})$, a sparse tag-CCA2 encryption scheme with pseudorandom ciphertexts
– $\mathsf{chcPRF} = (\mathsf{chcPRF.KeyGen}, \mathsf{chcPRF.Eval}, \mathsf{Constrain}, \mathsf{ConstrainEval})$, a constraint-hiding constrained PRF
– $\mathsf{PRF} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$, a PRF family
– $\mathsf{PRF}' = (\mathsf{PRF}'.\mathsf{KeyGen}, \mathsf{PRF}'.\mathsf{Eval})$, another PRF family.

We will use the following notations:

– $r^{\mathsf{in}} = r^{\mathsf{in}}(\lambda)$ and $r^{\mathsf{out}} = r^{\mathsf{out}}(\lambda)$ are the number of random bits used by $\mathsf{Enc}^{\mathsf{in}}$ and $\mathsf{Enc}^{\mathsf{out}}$, respectively;
– $(\mathcal{X}, \mathcal{Y}^{(1)}) = (\mathcal{X}_{\mathsf{pp}}, \mathcal{Y}^{(1)}_{\mathsf{pp}})$ are the input and output spaces of chcPRF; where we assume that $\mathcal{X}$ and $\mathcal{Y}^{(1)}$ are of size super-polynomial in λ;
– We'll suppose that PRF has input and output spaces $(\mathcal{X}, \{0,1\}^{r^{\mathsf{out}}}) = (\mathcal{X}_{\mathsf{pp}}, \{0,1\}^{r^{\mathsf{out}}})$;
– $\mathcal{CT} = \mathcal{CT}_{\mathsf{pp}}$ is the ciphertext space of $\mathcal{E}^{\mathsf{out}}$.
– We set the input space of our watermarkable PRF to be $\mathcal{X}$, and its output space to be $\mathcal{Y} = \mathcal{Y}^{(1)} \times \mathcal{CT}$. For $y \in \mathcal{Y}$, we will write $y = (y_1, y_2)$, where $y_1 \in \mathcal{Y}^{(1)}$ and $y_2 \in \mathcal{CT}$.
– We'll suppose that PRF' as input space $\mathcal{X}^{(1)}$ and output space $\mathcal{X}^{(2)}$ such that $\mathcal{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)}$, where we will suppose that both $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$ have super-polynomial size. In particular, for $x \in \mathcal{X}^{(1)}$, and $t \leftarrow \mathsf{PRF}'.\mathsf{KeyGen}$, we have $(x, \mathsf{PRF}'.\mathsf{Eval}(t, x)) \in \mathcal{X}$.

– For t a key for PRF', define $V_t := \{(x, \mathsf{PRF}'.\mathsf{Eval}(t,x)) \,|\, x \in \mathcal{X}^{(1)}\}$. Let C_t the circuit, which, on input $x \in \mathcal{X}$, parses x as $(x_1, x_2) \in \mathcal{X}^{(1)} \times \mathcal{X}^{(2)}$, and outputs 1 if $x_2 = \mathsf{PRF}'.\mathsf{Eval}(x_1)$, and outputs 0 otherwise; in other words, C_j tests membership in V_j. If the key t_j is indexed by some j, we will write V_j and C_j instead of the more cumbersome V_{t_j} and C_{t_j}.

We now describe our construction of a watermarking scheme, with its associated watermarkable PRF family:

– $\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, sample $(\mathsf{pk}^{\mathsf{in}}, \mathsf{sk}^{\mathsf{in}}) \leftarrow \mathcal{E}^{\mathsf{in}}.\mathsf{KeyGen}(1^\lambda)$ and $(\mathsf{pk}^{\mathsf{out}}, \mathsf{sk}^{\mathsf{out}}) \leftarrow \mathcal{E}^{\mathsf{out}}.\mathsf{KeyGen}(1^\lambda)$. Output:

$$\mathsf{pp} = (\mathsf{pk}^{\mathsf{in}}, \mathsf{pk}^{\mathsf{out}});$$

$$\mathsf{ek} = (\mathsf{sk}^{\mathsf{in}}, \mathsf{sk}^{\mathsf{out}}).$$

– $\mathsf{KeyGen}(1^\lambda, \mathsf{pp})$: On input the security parameter 1^λ, and the public parameters pp, sample $s \leftarrow \mathsf{chcPRF}.\mathsf{KeyGen}(1^\lambda)$, $s' \leftarrow \mathsf{PRF}.\mathsf{KeyGen}(1^\lambda)$ and $r \leftarrow \{0,1\}^{r^{\mathsf{in}}}$. Sample for $j \in \{0, \ldots, \ell\}$: $t_j \leftarrow \mathsf{PRF}'.\mathsf{KeyGen}(1^\lambda)$. The key of the watermarkable PRF is:

$$k = (s, (t_0, t_1 \ldots, t_\ell), r, s', \mathsf{pp}).$$

For ease of notation, we will simply write $k = (s, (t_0, t_1 \ldots, t_\ell), r, s')$ when the public parameters pp are clear from the context.

– $F_k(x)$: On input a key k and input x, output

$$F_k(x) = \Big(\; f_s(x),\; \mathsf{Enc}^{\mathsf{out}}_x(\mathsf{pk}_{\mathsf{out}}, \mathsf{Enc}^{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, (s, t_0, \ldots, t_\ell)\,;\, r)\,;\, f'_{s'}(x)) \; \Big)$$

where $f_s(\cdot) = \mathsf{pPRF}.\mathsf{Eval}(s, \cdot)$, $f'_{s'}(\cdot) = \mathsf{PRF}.\mathsf{Eval}(s', \cdot)$ and $\mathsf{Enc}^{\mathsf{out}}$ encrypts $\mathsf{Enc}^{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, (s, t_1, \ldots, t_\ell)\,;\, r)$ using tag x and randomness $f'_{s'}(x)$.

– $\mathsf{Mark}(k, \mathsf{msg})$: On input a key $k = (s, (t_0, t_1 \ldots, t_\ell), r, s')$, and a message $\mathsf{msg} \in \{0,1\}^\ell$, do the following:
  - Compute the circuit

$$C_{\mathsf{msg}} = C_0 \vee \bigvee_{\substack{j=1 \\ \mathsf{msg}_j = 1}}^{\ell} C_j,$$

    which on input $x \in \mathcal{X}$ outputs 1 if and only if $x \in V_0$ or if there exists some $j \in [\ell]$ such that $\mathsf{msg}_j = 1$ and $x \in V_j$, and 0 otherwise, where $V_j = \{(x_1, \mathsf{PRF}'.\mathsf{Eval}(t_j, x_1))\}_{x_1 \in \mathcal{X}^{(1)}}$.
  - Constrain the key s with respect to C_{msg}: $s_{\mathsf{msg}} \leftarrow \mathsf{chcPRF}.\mathsf{Constrain}(s, C_{\mathsf{msg}})$.
  - Compute $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, (s, t_0, \ldots, t_\ell)\,;\, r)$.
  - Output the marked key:

$$\widetilde{k} = (s_{\mathsf{msg}}, c^{\mathsf{in}}, s'),$$

    where the associated circuit computes:

$$F_{\widetilde{k}}(x) = \big(\; \mathsf{ConstrainEval}(s_{\mathsf{msg}}, x)\,,\, \mathsf{Enc}^{\mathsf{out}}_x(\mathsf{pk}_{\mathsf{out}}, c^{\mathsf{in}}\,;\, f'_{s'}(x)) \; \big)\,.$$

– $\mathsf{Extract}(\mathsf{ek}, C)$: Let $w = \lambda/\rho^2 = \mathrm{poly}(\lambda)$. On input the extraction key ek and a circuit C do the following:
  - If the input or output length of C do not match $\mathcal{X}$ and $\mathcal{Y}^{(1)} \times \mathcal{CT}$ respectively, output $\mathsf{unmarked}$.
  - For all $i \in [w]$ sample uniformly at random $x_i \leftarrow \mathcal{X}$, and do the following:
    * Parse $C(x_i) = (C_1(x_i), C_2(x_i))$ where $C_1(x_i) \in \mathcal{Y}^{(1)}$ and $C_2(x_i) \in \mathcal{CT}$.
    * Compute $c_i^{\mathsf{in}} = \mathsf{Dec}^{\mathsf{out}}_{\mathsf{sk}^{\mathsf{out}}, x_i}(C_2(x_i))$ (using secret key $\mathsf{sk}^{\mathsf{out}}$ and tag x_i);
    * If $c_i^{\mathsf{in}} \neq \perp$, compute $(s_i, t_{0,i}, \ldots, t_{\ell,i}) = \mathsf{Dec}^{\mathsf{in}}_{\mathsf{sk}^{\mathsf{in}}}(c_i^{\mathsf{in}})$.
  - Let $(s, t_0, \ldots, t_\ell)$ the majority of the w values $(s_i, t_{0,i}, \ldots, t_{\ell,i})$, where $i \in [w]$ (that is if $\mathsf{Dec}^{\mathsf{in}}_{\mathsf{sk}_{\mathsf{in}}}$ outputs some $(s, t_0, \ldots, t_\ell)$ more than $w/2$ times in the loop above). If such a majority does not exist, stop here and output $\mathsf{unmarked}$.
  - For $i \in [w]$, do the following:
    * Sample $z_{0,i} \leftarrow V_0$ where $V_0 = \{(x, \mathsf{PRF}'.\mathsf{Eval}(t_0, x)) \,|\, x \in \mathcal{X}^{(1)}\}$.
      This is done by picking a random $z_1 \leftarrow \mathcal{X}^{(1)}$ and setting $z = (z_1, \mathsf{PRF}'.\mathsf{Eval}(t_0, z_1))$.
    * Test $C_1(z_{0,i}) \neq f_s(z_{0,i})$.
    * If a majority are equal, stop here and output $\mathsf{unmarked}$.
  - For $j \in [\ell]$, do the following:
    * For $i \in [w]$ sample $z_{j,i} \leftarrow V_j$ where $V_j = \{(x, \mathsf{PRF}'.\mathsf{Eval}(t_j, x)) \,|\, x \in \mathcal{X}^{(1)}\}$.
    * Test for $i \in [w]$: $C_1(z_{j,i}) \neq f_s(z_{j,i})$.
    * If a majority are different (for $i \in [w]$), set $\mathsf{msg}_j = 1$, otherwise set $\mathsf{msg}_j = 0$.
  - Output $\mathsf{msg} = (\mathsf{msg}_1, \ldots, \mathsf{msg}_\ell)$.

Note that when it is clear from the context, we will omit writing $\mathsf{pk}_{\mathsf{out}}, \mathsf{pk}_{\mathsf{in}}$.

4.3 Correctness Properties of the Watermarking Scheme

Claim. Assuming $\mathcal{E}^{\mathsf{in}}$ and $\mathcal{E}^{\mathsf{out}}$ are perfectly correct and $\mathcal{E}^{\mathsf{in}}$ is sparse, the scheme above satisfies the non-triviality properties.

Proof. 1. For $(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $k = (s, (t_0, t_1 \ldots, t_\ell), r, s') \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp})$, we have that $\mathsf{Extract}$, on input F_k, gets $(s, t_0, \ldots, t_\ell)$ as the majority, by perfect correctness of $\mathcal{E}^{\mathsf{in}}$ and $\mathcal{E}^{\mathsf{out}}$. Therefore, the first check (corresponding to $j = 0$) makes $\mathsf{Extract}(\mathsf{ek}, F_k)$ output $\mathsf{unmarked}$ (as $(F_k)_1(z) = f_s(z)$ for all $z \in \mathcal{X}$).

2. Let C be a fixed circuit, and let $(\mathsf{pp}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$. By sparsity of $\mathcal{E}^{\mathsf{out}}$, the probability that any of the w values $C_1(x_i)$ decrypts, for $i \in [w]$, is negligible. Therefore, $\mathsf{Extract}(\mathsf{ek}, C)$ outputs $\mathsf{unmarked}$.

Claim. Assuming PRF and PRF' are secure, chcPRF preserves functionality on unconstrained inputs, and $\mathcal{E}^{\mathsf{out}}$ is tag-CCA2 with pseudorandom ciphertexts, then the scheme above satisfies strong correctness.

Proof. We use the exact same hybrids as in the non-message embedding case, after which the view of the adversary is independent of $(t_0, \ldots, t_\ell)$. We then conclude in two steps. First, the probability that the adversary outputs a constrained point is negligible. This is by PRF security of PRF'. Actually, the adversary cannot even find a point in any V_j, $j \in \{0, \ldots, \ell\}$ (where the set of constrained points the union of a subset of the V_j's defined by msg^*), as it would be indistinguishable from predicting one of $\ell + 1$ random values (in $\mathcal{X}^{(2)}$, the output space of PRF').

Second, the probability that the adversary finds an unconstrained point on which F_k and $F_{\widetilde{k}}$ differ is also negligible; this is by functionality preserving of chcPRF on unconstrained inputs (which is implied by our definition of constraint-hiding).

4.4 Security Properties of the Watermarking Scheme

Extended Pseudorandomness. We show here that our scheme satisfies Extended Pseudorandomness.

Claim (Extended Pseudorandomness). Suppose chcPRF and PRF are secure, and that $\mathcal{E}^{\mathsf{out}}$ is tag-CCA2 with pseudorandom ciphertexts. Then the watermarking scheme satisfies extended pseudorandomness.

Proof. The proof is similar to the one for Claim 3.4. The only difference is that we now also keep track of the points $z_{j,i}$ sampled from the sets V_j during the calls to the extraction oracle, and we abort in Hybrids 2 to 4 if any of the points $z_{j,i}$ is queried to the PRF oracle. This event only occurs with negligible probability as the sets V_j are of super-polynomial size.

Unremovability. We prove that our construction is ε-unremovable (where $\varepsilon = 1/2 + \rho$ where $\rho = 1/\mathrm{poly}(\lambda)$ is a parameter of our scheme).

Claim. Suppose that $\mathcal{E}^{\mathsf{in}}$ is CCA2-secure, chcPRF is a constraint-hiding constrained PRF, and PRF' is a PRF. Then the watermarking scheme is ε-unremovable.

Proof. We prove the claim via a sequence of hybrids.

Hybrid 0. This is the ε-Unremovability game $\mathsf{Exp}^{\mathsf{remov-msg}}_{\mathcal{A}}(1^\lambda)$.

Hybrid 1. We now change how extraction calls are answered (including the one used to determine the output of the experiment). Let $k = (s, (t_0, \ldots, t_\ell), r, s') \leftarrow \mathsf{PRF}_{\mathsf{pp}}.\mathsf{KeyGen}(1^\lambda, \mathsf{pp})$ be the (unmarked) PRF key sampled to produce the challenge marked circuit, and $c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(s, t_0, \ldots, t_\ell\,; r)$ be the associated ciphertext (which is used to produce the challenge marked circuit $\widetilde{C}$). On extraction query C from the adversary, the extraction procedure samples x_i's for $i \in [w]$ as before. Let denote by E the event that $\mathsf{Dec}^{\mathsf{out}}_{\mathsf{sk}^{\mathsf{out}}, x_i}(C_2(x_i)) = c^{\mathsf{in}}$, i.e. the second part $C_2(x_i)$ decrypts to c^{in} when decrypting using tag x_i. If E occurs, then instead

of decrypting this inner ciphertext c^{in} in the extraction procedure, we directly consider it as outputting $(s, t_0, \ldots, t_\ell)$ (used to pick the majority of decryption outputs); in particular c^{in}, s and $(t_0, \ldots, t_\ell)$ are now hard-coded in the modified extraction procedure.

Hybrid 2. We change how extraction calls are answered and how the challenge marked circuit is generated. Let $0_{\mathcal{K}}$ and $0_{\mathcal{K}'}$ be arbitrary fixed keys for chcPRF and PRF' respectively. We now use:

$$c^{\mathsf{in}} = \mathsf{Enc}^{\mathsf{in}}(0_{\mathcal{K}}, 0_{\mathcal{K}'}^{\ell+1}),$$

which is used as the ciphertext hard-coded in the extraction oracle (used to handle event E), and used to produce the challenge marked circuit $\widetilde{C}$. Furthermore, we abort the experiment if the adversary makes any extraction query before submitting his challenge message such that $C_2(x_i)$ gets decrypted to c^{in} for any $i \in [w]$ (where c^{in} is defined before giving the adversary oracle access to the extraction oracle).

Hybrid 3. We change how we produce the challenge marked circuit $\widetilde{C}$ and how we answer extraction queries (including the one determining the output of the experiment). First, to generate the challenge marked circuit, we use the simulator from the constraint-hiding experiment to generate a simulated key $\widehat{s}_{\mathsf{msg}^*} \leftarrow \mathsf{Sim}^{\mathsf{key}}(1^{|C_{\mathsf{msg}^*}|})$.

On extraction query C, we now abort if it considers any $c_i^{\mathsf{in}} \neq c^{\mathsf{in}}$ such that $\mathsf{Dec}^{\mathsf{in}}_{\mathsf{sk}^{\mathsf{in}}}(c_i^{\mathsf{in}}) = (s, *, \ldots, *)$. Furthermore, if we have $\mathsf{Dec}^{\mathsf{out}}_{\mathsf{sk}^{\mathsf{out}}, x_i}(C_2(x_i)) = c^{\mathsf{in}}$ for more than $w/2$ samples $i \in [w]$ in the same execution of the extraction procedure, we use the constraint-hiding simulator and check $C_1(z_{j,i}) \neq \mathsf{Sim}^{\mathsf{ch}}(z_{j,i}, C_{\mathsf{msg}^*}(z_{j,i}))$ where $z_{j,i} \leftarrow V_j$ for $i \in [w]$ and $j \in \{0, \ldots, \ell\}$ (instead of checking $C_1(z_{j,i}) \neq f_s(z_{j,i})$).

If c^{in} appears in less than $w/2$ samples, we ignore it in the majority election.

Hybrid 4. We modify how we answer extraction queries (including the one determining the output of the experiment). We now pick $\ell + 1$ random functions $R_j : \mathcal{X}^{(1)} \to \mathcal{X}^{(2)}$ for $j \in \{0, \ldots, \ell\}$. Define $W_j := \{(x, R_j(x)) \,|\, x \in \mathcal{X}^{(1)}\}$. If c^{in} appears in more than $w/2$ samples $i \in [w]$, we now sample $z_{j,i} \leftarrow W_j$, and check $C_1(z_{j,i}) \neq \mathsf{Sim}^{\mathsf{ch}}(z_{j,i}, d_{\mathsf{msg}^*}(z_{j,i}))$ instead, where $d_{\mathsf{msg}^*}(z) = 1$ if $z_2 = R_0(z_1)$ or if there exists some j such that $\mathsf{msg}_j^* = 1$ and $z_2 = R_j(z_1)$, where $z = (z_1, z_2)$.

Hybrid 5. Now if c^{in} appears in more than $w/2$ indices $i \in [w]$, for $j \in \{0, \ldots, \ell\}$ we sample $z_{j,i} \leftarrow W_j$, and check:

- $C_1(z_{0,i}) \neq \mathsf{Sim}^{\mathsf{ch}}(z_{0,i}, 1)$ for $j = 0$;
- $C_1(z_{j,i}) \neq \mathsf{Sim}^{\mathsf{ch}}(z_{j,i}, \mathsf{msg}_j^*)$ for $j \in [\ell]$.

We now argue that in Hybrid 5, the experiment outputs 0 with overwhelming probability.

Consider the execution of the extraction algorithm that determines the output of the experiment. We have $C^* \cong_{(1/2+\rho)} \widetilde{C}$ by admissibility of the adversary.

Hence, a Chernoff bound on the $w = \lambda/\rho^2$ random samples x_i picked by the extraction call gives that with probability at least $(1 - e^{-2\lambda})$, the majority of the x_i satisfy $C^*(x_i) = \widetilde{C}(x_i)$. In particular, by perfect correctness of $\mathcal{E}^{\mathsf{out}}$, we have $c_i^{\mathsf{in}} = c^{\mathsf{in}}$ for a majority of indices $i \in [w]$.

Therefore, for all $j \in \{0, \ldots, \ell\}$, the extraction algorithm now samples $z_{j,i} \leftarrow W_j$ for $i \in [w]$, and tests $C_1(z_{0,i}) \neq \mathsf{Sim}^{\mathsf{ch}}(z_{0,i}, 1)$ for $j = 0$, and $C_1(z_{j,i}) \neq \mathsf{Sim}^{\mathsf{ch}}(z_{j,i}, \mathsf{msg}_j^*)$ for $j \in [\ell]$. But then, by randomness of R_j, the probability that a random $z_{j,i} \leftarrow W_j$ satisfies $C^*(z_{j,i}) = \widetilde{C}(z_{j,i})$ is at least $1/2 + \rho$ (up to some negligible statistical loss upper bounded by $wQ/|X^{(1)}|$ due to the previous $Q = \mathrm{poly}(\lambda)$ extraction queries). Therefore, another Chernoff bound states that with overwhelming probability, the majority of those $z_{j,i}$'s (over $i \in [w]$) satisfy $C^*(z_{j,i}) = \widetilde{C}(z_{j,i})$.

Now, if $\mathsf{msg}_j^* = 0$, we have $\widetilde{C}_1(z_{j,i}) = \mathsf{Sim}^{\mathsf{ch}}(z_{j,i}, \mathsf{msg}_j^*)$ with overwhelming probability by (computational) correctness of chcPRF.

If $\mathsf{msg}_j^* = 1$ we have $\mathsf{Sim}^{\mathsf{ch}}(z_{j,i}, \mathsf{msg}_j^*) = R(z_{j,i})$ for a random function $R : \mathcal{X} \to \mathcal{Y}^{(1)}$ (picked independently of $\widetilde{C}$), so the probability that some index i satisfies $\widetilde{C}(z_{j,i}) = R(z_{j,i})$ is negligible over the randomness of R (again, even conditioned on the polynomial number $(\ell+1)wQ$ of evaluations to R during the extraction queries, as $\mathcal{Y}^{(1)}$ has super-polynomial size). Overall, an union bound gives that the extraction procedure, on input C^*, does not output $\mathsf{unmarked}$ with overwhelming probability (corresponding to $j = 0$), and then outputs msg^* with overwhelming probability.

Indistinguishability of the Hybrids. We now show that the hybrids above are indistinguishable.

Lemma 11. *Assuming $\mathcal{E}^{\mathsf{in}}$ is perfectly correct, we have* **Hybrid 0 ≡ Hybrid 1**.

The view of the adversary is identical in Hybrid 0 and Hybrid 1 by perfect correctness of the inner encryption $\mathcal{E}^{\mathsf{in}}$: in the latter we simply hardcode the result of the decryption whenever we have to decrypt c^{in} during an extraction oracle call.

Lemma 12. *Assuming $\mathcal{E}^{\mathsf{in}}$ is CCA2-secure, we have* **Hybrid 1** $\overset{\mathrm{c}}{\approx}$ **Hybrid 2**.

The same argument as for Lemma 3 holds, by additionally noting that any adversary who queries, before receiving the marked circuit, some C such that the extraction call on C gets c^{in} with substantial probability can be directly used to break CCA securiy of $\mathcal{E}^{\mathsf{in}}$.

Lemma 13. *Assuming* chcPRF *is a constraint-hiding constrained PRF, we have* **Hybrid 2** $\overset{\mathrm{c}}{\approx}$ **Hybrid 3**.

First, any adversary who queries some C such that the extraction call on C gets some c_i^{in} such that $\mathsf{Dec}^{\mathsf{in}}_{\mathsf{sk}^{\mathsf{in}}}(c_i^{\mathsf{in}}) = (s, *, \ldots, *)$ can be directly used to break constraint-hiding (as the challenger has $\mathsf{sk}^{\mathsf{in}}$, he can extract the PRF key s using

such an adversary). Then, the reduction is very similar to the proof of Lemma 4, by receiving both the constrained key $\widetilde{k}$ and values y_i^* from the constraint-hiding experiment to answer the extraction queries where c^{in} form the majority. This is because c^{in} is now the only possible ciphertext that makes the extraction procedure use evaluations to the constrained PRF $f_s(\cdot)$.

Lemma 14. *Assuming the security of* PRF'*, we have* **Hybrid 3** $\overset{\mathrm{c}}{\approx}$ **Hybrid 4**.

As the challenge marked circuit does not depend on $(t_0, \ldots, t_\ell)$ anymore, all the steps involving PRF' in Hybrid 3 can be simulated given only oracle access to $\mathsf{PRF}'.\mathsf{Eval}$ (on different keys $t_0, \ldots, t_\ell$). More precisely, we have to sample random points in V_j and compute $C_{\mathsf{msg}^*}(z_{j,i})$ (given as input to $\mathsf{Sim}^{\mathsf{ch}}$ if c^{in} form the majority). This gives a simple reduction to the PRF security of PRF', using a standard hybrid argument over the $\ell + 1$ PRF keys $t_0, \ldots, t_1$.

Lemma 15. *We have* **Hybrid 4** $\overset{\mathrm{s}}{\approx}$ **Hybrid 5**.

Hybrids 4 and 5 differ exactly when there is some point $z_{j,i} \leftarrow W_j$ such that $d_{\mathsf{msg}^*}(z_{j,i}) \neq \mathsf{msg}_j^*$, which happens exactly when $\mathsf{msg}_j^* = 0$ but $d_{\mathsf{msg}_j^*}(z_{j,i}) = 1$, that is, when $\mathsf{msg}_j^* = 0$ but $z_{j,i} \in W_{j'}$ for some $j' \neq j$ such that $\mathsf{msg}_{j'}^* = 1$. By definition, this implies having $R_j(z_1) = R_{j'}(z_1)$ for some $j' \neq j$ for some independently chosen random functions R_j and $R_{j'}$; and the probability that this happens, even conditioned on the $\ell = \mathrm{poly}(\lambda)$ functions R'_j and the $Q \cdot w = \mathrm{poly}(\lambda)$ samples $z_{j,i}$ picked accross all the extraction queries made by the adversary (where Q denotes the number of extraction queries he makes), is negligible, as $\mathcal{X}^{(2)}$ has super-polynomial size.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
2. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
3. Barak, B., et al.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6 (2012). http://dblp.uni-trier.de/db/journals/jacm/jacm59.html#BarakGIRSVY12
4. Boneh, D., Kim, S., Montgomery, H.: Private puncturable PRFs from standard lattice assumptions. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 415–445. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_15
5. Boneh, D., Lewi, K., Wu, D.J.: Constraining pseudorandom functions privately. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 494–524. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_17
6. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_15

7. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_29
8. Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and more) from LWE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 264–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_10
9. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFs for NC^1 from LWE. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 446–476. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_16
10. Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking cryptographic capabilities. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC, pp. 1115–1127. ACM Press, June 2016
11. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
12. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS, pp. 40–49. IEEE Computer Society Press, October 2013
13. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
14. Hopper, N., Molnar, D., Wagner, D.: From weak to strong watermarking. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 362–382. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_20
15. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13, pp. 669–684. ACM Press, November 2013
16. Kim, S., Wu, D.J.: Watermarking cryptographic functionalities from standard lattice assumptions. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 503–536. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_17
17. Naccache, D., Shamir, A., Stern, J.P.: How to copyright a function? In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 188–196. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49162-7_14
18. Nishimaki, R.: How to watermark cryptographic functions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 111–125. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_7
19. Peikert, C., Shiehian, S.: Privately constraining and programming PRFs, the LWE Way. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 675–701. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_23
20. Yoshida, M., Fujiwara, T.: Toward digital watermarking for cryptographic data. IEICE Trans. **94–A**(1), 270–272 (2011). http://dblp.uni-trier.de/db/journals/ieicet/ieicet94a.html#YoshidaF11